Inertial Yaw-Independent Velocity and Attitude Estimation for High Speed Quadrotor Flight

James Svacha[1], Giuseppe Loianno[2], and Vijay Kumar[1]

***Abstract*—This work addresses the velocity and attitude estimation of a quadrotor using motor speeds and inertial measurement unit (IMU) data. This result is obtained through a novel filter that employs a first order drag model which allows the velocity to be observed through the drag forces acting on the quadrotor. These forces are measured with the IMU. Compared to our previous contribution [1], decoupling the attitude into a yaw-tilt convention results in the tilt and velocity in the body-fixed frame being independent of the yaw. Thus, the velocity and attitude estimates are no longer affected by yaw drifts obtained from dead-reckoning procedures. The camera available on the vehicle can optionally be employed to recover the yaw in a computationally efficient way without requiring RANSAC, instead using the estimated velocity and attitude from the filter. The IMU is concurrently used to estimate the thrust, linear drag coefficients, and accelerometer biases. Experimental results on a quadrotor platform show the feasibility of the proposed approach.**

***Index Terms*—Aerial Systems: Perception and Autonomy, Sensor Fusion**

I. Introduction

QUADROTORS are used in a variety of applications such as surveillance and monitoring [2], exploration [3], agricultural monitoring, and structural inspection [4]. The main advantage of using quadrotors is their vertical takeoff and landing (VTOL) capabilities, which allow them to hover in place and fly within narrow confinements.

Despite their simplicity, quadrotors are limited in payload by size, weight, and power constraints. These constraints reduce the weight of sensors and computational hardware that can be outfitted to small-scale platforms. Nearly all quadrotors are outfitted with cheap micro electro-mechanical systems inertial measurement units (IMUs). Many small-scale cameras may be outfitted to small platforms, but computer vision algorithms are prone to failures or often require many computational resources [5], including those required by iterative random sampling outlier rejection algorithms such as random sample consensus (RANSAC). An ongoing area of research is focused on improving the quality of information that can be extracted from small sensors such as the IMU and reducing the computational cost of extracting visual information from the environment.

Fig. 1: The vehicle used during the experiment, based on [6].

In this work, we investigate the problem of estimating the velocity and attitude of a quadrotor primarily using an IMU, and only using a camera for the heading, or yaw, of the quadrotor. The IMU-based state estimation is made possible by a first-order drag model which allows the velocity of the quadrotor to be measured through the drag force it induces. We introduce a computationally efficient method for outlier rejection of image features that does not require random sampling, reducing the computational cost of yaw estimation using vision. In addition, we demonstrate that the yaw is not required in order to estimate the tilt and body-frame linear velocity of the quadrotor.

Previous works on estimating the velocity and attitude of quadrotors using only the IMU include [7], [8], [9], and [1]. In [7], a known drag coefficient was used in an extended Kalman filter (EKF) to estimate the x-y velocity in the body-fixed frame and the roll and pitch of the quadrotor. In [8], the authors estimate the drag coefficient as well. In [9], authors use a linear drag model and a thrust model to estimate the full body-frame velocity as well as the roll and pitch, but they assume the model parameters are known. In our previous work [1], a first attempt to estimate the full 3D velocity, tilt component of the attitude, accelerometer biases, and thrust and drag coefficients through an unscented Kalman filter (UKF) framework was proposed. However, the main shortcoming was given by the assumption that the yaw was considered known. In this work, this problem is addressed choosing a proper orientation convention.

The contributions of this paper are threefold. First, we show a way to estimate the body velocity and tilt of the vehicle just using motor inputs and inertial data. The proposed orientation

Manuscript received: September, 10, 2018; Revised October, 30, 2018; Accepted January, 13, 2019.

This paper was recommended for publication by Editor J. Roberts upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Qualcomm Research, the ARL grant DCIST CRA W911NF-17-2-0181, ONR grants N00014-07-1-0829, N00014-14-1-0510, ARO grant W911NF-13-1-0350, NSF grants IIS-1426840, IIS-1138847.

[1]The authors are with the GRASP Lab, University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104, USA. email: {jsvacha, kumar}@seas.upenn.edu.

[2] The author is with the New York University, Tandon School of Engineering, Brooklyn, NY 11201, USA. email: {loiannog}@nyu.edu.

Digital Object Identifier (DOI): see top of this page.

convention makes the tilt and velocity independent of the yaw. Second, the filter information can be used in an efficient way to reject outlying image correspondences so that the yaw angle may be estimated. Third, an observability analysis is provided to identify what type of motions are more suited to obtain a reliable state estimate.

The paper is organized as follows. In Section II, we provide an overview of attitude conventions, showing why a yaw-tilt convention is preferred over a tilt-yaw convention. The modeling and implementation of the proposed estimator are also detailed. Section III explains how to implement a UKF on a non-Euclidean manifold to estimate the tilt and velocity of the quadrotor in the body-fixed frame. Section IV shows that the yaw can be estimated using vision, and that the UKF output can be used to reject outliers in a computationally efficient manner. In Section V an observability analysis is provided to verify what type of motions allow a consistent state identification, whereas Section VI presents the experiments using a quadrotor to validate the proposed approach. Finally, Section VII concludes the work and discusses future scenarios.

II. Modeling

The inertial (world) coordinate frame is denoted $\mathcal{W}$, and it has coordinate axes $\mathbf{e}_1$ pointing North, $\mathbf{e}_2$ pointing West, and $\mathbf{e}_3$ pointing upwards. The body-fixed frame is denoted $\mathcal{B}$, and it has coordinate axes $\mathbf{b}_1$ pointing forward, $\mathbf{b}_2$ pointing to the quadrotor's left, and $\mathbf{b}_3$ pointing in the direction of thrust. The attitude of the quadrotor is given by $R = \begin{pmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3\end{pmatrix} \in SO(3)$. The camera coordinate frame is denoted $\mathcal{C}$. In our case this camera faces downwards, but this is not required. The frames $\mathcal{W}$, $\mathcal{B}$, and $\mathcal{C}$ are visualized in Figure 2.

A. Attitude Convention

In [1], we used the convention $R = R_\phi R_\psi$, where

$$R_\psi = \begin{pmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{pmatrix}, \text{ and} \tag{1}$$

$$R_\phi = \begin{pmatrix}\frac{z_2^2+z_1^2z_3}{z_1^2+z_2^2} & \frac{z_1z_2(z_3-1)}{z_1^2+z_2^2} & z_1\\ \frac{z_1z_2(z_3-1)}{z_1^2+z_2^2} & \frac{z_1^2+z_2^2z_3}{z_1^2+z_2^2} & z_2\\ -z_1 & -z_2 & z_3\end{pmatrix}. \tag{2}$$

In the above equations, $\mathbf{z} = \begin{pmatrix}z_1 & z_2 & z_3\end{pmatrix}^\top$ is the *tilt* of the robot, which is an element of the 2-sphere S^2. If $R = R_\phi R_\psi$, $\mathbf{z}$ is the direction of the $\mathbf{b}_3$ axis expressed in $\mathcal{W}$. The *yaw* of the robot ψ is the angle of rotation of the quadrotor about the $\mathbf{b}_3$ axis. R_ϕ can easily be obtained using Rodrigues' rotation formula.

After some analysis, we can analytically show a disadvantage of this attitude convention. First, we write $R_\phi = RR_\psi^\top$ and differentiate:

$$\begin{aligned}\dot{R}_\phi &= \dot{R}R_\psi^\top + R\dot{R}_\psi^\top\\ &= R[\boldsymbol{\omega}]_\times R_\psi^\top - R[\dot\psi\mathbf{e}_3]_\times R_\psi^\top\\ &= R[\boldsymbol{\omega} - \dot\psi\mathbf{e}_3]_\times R_\psi^\top,\end{aligned} \tag{3}$$

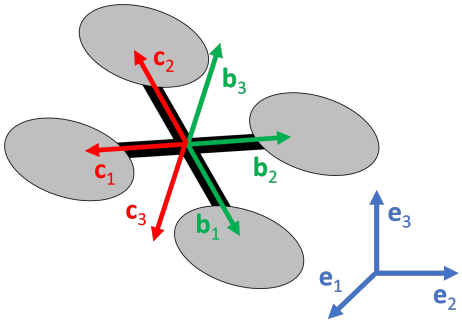


Fig. 2: The body-frame $\mathcal{B}$ (green), camera frame $\mathcal{C}$ (red), and world frame $\mathcal{W}$ (blue).

where $\boldsymbol{\omega} = \begin{pmatrix}\omega_1 & \omega_2 & \omega_3\end{pmatrix}^\top$ is the angular velocity of the quadrotor expressed in $\mathcal{B}$, and $[\mathbf{v}]_\times$ denotes the skew-symmetric matrix such that, for all $\mathbf{v}, \mathbf{w} \in \mathbb{R}^3$, $\mathbf{v}\times\mathbf{w} = [\mathbf{v}]_\times\mathbf{w}$. As seen in (3), the rate of change of tilt is dependent on ψ. This is a problem because if ψ is unknown, we cannot stabilize the tilt. However, if, instead of using the convention $R = R_\phi R_\psi$, we use $R = R_\psi R_\phi$, we get $R_\phi = R_\psi^\top R$. Differentiating gives us:

$$\begin{aligned}\dot{R}_\phi &= \dot{R}_\psi^\top R + R_\psi^\top\dot{R}\\ &= -[\dot\psi\mathbf{e}_3]_\times R_\psi^\top R + R_\psi^\top R[\boldsymbol{\omega}]_\times\\ &= -[\dot\psi\mathbf{e}_3]_\times R_\phi + R_\phi[\boldsymbol{\omega}]_\times.\end{aligned} \tag{4}$$

If we equate the third rows of each side of (4), we get:

$$\dot z_1 = \omega_3 z_2 + \omega_2 z_3, \tag{5}$$

$$\dot z_2 = -\omega_3 z_1 - \omega_1 z_3, \tag{6}$$

$$\dot z_3 = -\omega_2 z_1 + \omega_1 z_2. \tag{7}$$

Note that equations (5) through (7) show that the rate of change of tilt only depends on the tilt itself and the angular velocity in the body frame. These equations may be rewritten:

$$\dot{\mathbf{z}} = -R_z(\pi)[\boldsymbol{\omega}]_\times R_z(\pi)\mathbf{z}, \tag{8}$$

where $R_z(\pi) = \text{diag}\begin{pmatrix}-1 & -1 & 1\end{pmatrix}^\top$ represents an elementary rotation about the z axis by π radians. When $R = R_\psi R_\phi$, $\mathbf{z}$ is the direction of $\mathbf{b}_3$ not in the world frame $\mathcal{W}$, but a frame obtained by rotating $\mathcal{W}$ about the $\mathbf{e}_3$ axis by the angle ψ.

The independence of the rate of change of the tilt on the yaw makes intuitive sense because stabilizing the tilt in a local yawed frame should not require knowledge of that yaw. Thus we choose to use $R = R_\psi R_\phi$ in this work.

Notice from (4) that $\dot R_\phi$ is dependent on $\dot\psi$, but not ψ itself. Consequently, $\dot\psi$ is independent of ψ. We may derive an expression for $\dot\psi$ that depends only on the tilt $\mathbf{z}$ and the angular velocity in the body frame $\boldsymbol{\omega}$. First, we equate the upper right entry of (4) to get:

$$\dot z_1 = \dot\psi z_2 - \frac{\omega_1 z_1 z_2(z_3-1)}{z_1^2+z_2^2} + \frac{\omega_2(z_2^2+z_1^2z_3)}{z_1^2+z_2^2}. \tag{9}$$

Substituting (5) and solving (9) for $\dot\psi$ and simplifying using the fact that $\|\mathbf{z}\| = 1$ gives:

$$\dot\psi = \omega_3 - \frac{\omega_1 z_1 + \omega_2 z_2}{1+z_3}. \tag{10}$$

B. Process Model

The state vector is given by:

$$\mathbf{x} = \begin{pmatrix} \psi & \mathbf{z}^\top & \mathbf{v}^\top & \mathbf{b}^\top & \mathbf{k}^\top \end{pmatrix}^\top, \tag{11}$$

where $\mathbf{v} \in \mathbb{R}^3$ is the velocity of the quadrotor expressed in the components of $\mathcal{B}$, $\mathbf{b} \in \mathbb{R}^3$ is the accelerometer bias vector, and $\mathbf{k} = \begin{pmatrix} k_f & k_d & k_z \end{pmatrix}^\top$ where k_f is the thrust coefficient and k_d and k_z are the linear x-y and z drag coefficients, respectively.

We use the Newton equations with first-order drag effects and the previously derived rotation kinematics to model the quadrotor system. The dynamics are

$$\dot{\psi} = \omega_3 - \frac{\omega_1 z_1 + \omega_2 z_2}{1 + z_3} + \eta_\psi, \tag{12}$$

$$\dot{\mathbf{z}} = -R_z(\pi)[\boldsymbol{\omega}]_\times R_z(\pi)\mathbf{z} + \boldsymbol{\eta}_z, \tag{13}$$

$$\dot{\mathbf{v}} = \frac{k_f}{m} u_{ss} \mathbf{e}_3 - \frac{u_s}{m} D\mathbf{v} - g R_z(\pi)\mathbf{z} - \boldsymbol{\omega} \times \mathbf{v} + \boldsymbol{\eta}_v, \tag{14}$$

where m is the mass of the quadrotor, u_{ss} is the sum of the squared motor speeds u_i for $i \in \{1, \ldots, 4\}$, u_s is the sum of the motor speeds, $D = \operatorname{diag}\begin{pmatrix} k_d & k_d & k_z \end{pmatrix}$ is the matrix of drag coefficients, g is the acceleration due to gravity, and the η terms represent additive Gaussian white noise. In addition, we have $\dot{\mathbf{b}} = \boldsymbol{\eta}_b$ and $\dot{\mathbf{k}} = \mathbf{0}$. The inputs to the process model are u_{ss}, u_s, and $\boldsymbol{\omega}$. We make the following clarifications:

1) The yaw actually belongs to the 1-sphere S^1, and $\psi \in [0, 2\pi)$ is merely a coordinate for this space. When discretizing the dynamics in (12), we add the increment in yaw and then wrap the resulting angle so it is in the interval $[0, 2\pi)$.
2) Although $\mathbf{x}$ appears to be 13 dimensional, it is actually 12 dimensional since $\mathbf{z}$ lies on S^2, which is two-dimensional. To update $\mathbf{z}$ according to (13), we must ensure that $\mathbf{z}$ remains of unit length. We do this using the exponential map on S^2, which we discuss later.
3) Part of the state evolves on the manifold S^2. This requires an extension of the notions of Gaussian distributions and their mean and covariance to this space. Specifically, the noise $\boldsymbol{\eta}_z$ exists on the tangent space of S^2 at the point $\mathbf{z}$, denoted $T_\mathbf{z}S^2$, which is the space of the velocities of all curves in S^2 passing though the point $\mathbf{z}$. This procedure is advantageous because it minimally parameterizes the covariance. For more details, the reader may refer to [10].

C. Measurement Model

The measurements are considered to be the yaw, which is obtained from tracked features using a camera, and the specific acceleration of the quadrotor, obtained from the accelerometer. These are shown below:

$$y_1 = \psi + \nu_\psi \tag{15}$$

$$\mathbf{y}_2 = \frac{k_f}{m} u_{ss} \mathbf{e}_3 - \frac{u_s}{m} D\mathbf{v} + \mathbf{b} + \boldsymbol{\nu}_a, \tag{16}$$

where the ν terms represent additive Gaussian noise.

III. UKFs on Riemannian Manifolds

The state and measurement spaces in question are non-Euclidean because they involve the tilt and the yaw, which are on S^2 and S^1, respectively. An overview of UKF implementation on Riemannian manifolds is given in [10], but the basics will be presented here as well.

In [1], we used stereographic coordinates for the tilt. This was arbitrary and the filter does not depend on the choice of coordinates. We can choose not to use coordinates by using unit vectors in $\mathbb{R}^2$ to represent the yaw in S^1 or unit vectors in $\mathbb{R}^3$ to represent the tilt in S^2.

In our implementation which is derived from [1], we use stereographic coordinates for the tilt and angles constrained to $[0, 2\pi)$ for the yaw. When necessary, we convert back and forth from unit vectors to do various tasks like computing sigma points. Hence, we will discuss the UKF procedures such as sigma point generation and parallel transport on manifolds without using coordinates.

A. Generating Sigma Points

Let $\mathbf{p}$ be a point on an m dimensional smooth manifold $\mathcal{M}$. In our case, $\mathcal{M}$ is either S^1 or S^2 and hence $\mathbf{p}$ is a unit vector. Let $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_m\}$ be an orthogonal basis for the tangent space of $\mathcal{M}$ at the point $\mathbf{p}$, which is denoted $T_\mathbf{p}(\mathcal{M})$. Like $\mathbf{p}$, the tangent vectors $\mathbf{t}_i$ are expressed in $\mathbb{R}^n$, the Euclidean space in which $\mathcal{M}$ is embedded.

Let $\exp_\mathbf{p} : T_\mathbf{p}\mathcal{M} \to \mathcal{M}$ be the exponential map that maps a tangent vector $\mathbf{v}$ at point $\mathbf{p} \in \mathcal{M}$ to the point on $\mathcal{M}$ obtained by traveling along the geodesic of length $\|\mathbf{v}\|$ in the direction of $\mathbf{v}$. The exponential map on S^2 is derived in [1]. For S^1, we obtain the exponential map from that on S^2 by restricting $\mathbf{p}$ to S^1 and $\mathbf{v}$ to $T_\mathbf{p}S^1$.

If $\boldsymbol{\delta}_i \in \mathbb{R}^m$ is the deviation of the sigma point $\mathcal{P}_i$ from the state $\mathbf{p}$, then

$$\mathcal{P}_i = \exp_\mathbf{p}\left(\sum_{j=1}^{m} \delta_{ij}\mathbf{t}_j\right), \tag{17}$$

where δ_{ij} is the jth component of $\boldsymbol{\delta}_i$.

B. Sigma Point Integration

If $\mathcal{P}_i \in \mathcal{M}$ is the ith sigma point on the manifold and $\mathbf{v}_i \in T_{\mathcal{P}_i}\mathcal{M}$ is the velocity of $\mathcal{P}_i$ obtained from the dynamics, then the updated sigma point $\mathcal{P}_i^+$ over the time interval Δt may be obtained by using Euler integration:

$$\mathcal{P}_i^+ = \exp_{\mathcal{P}_i}(\Delta t \mathbf{v}_i). \tag{18}$$

C. Averaging

The weighted average $\bar{\mathcal{P}}$ of points $\mathcal{P}_i \in \mathcal{M}$, $i \in \{1, \ldots, N\}$, with corresponding weights w_i, is the point $\bar{\mathcal{P}}$ on $\mathcal{M}$ that minimizes the weighted sum (using the same weights w_i) of the geodesics from the $\mathcal{P}_i$'s to $\bar{\mathcal{P}}$. Notice that $\bar{\mathcal{P}}$ may not be unique.

Assuming that the sigma points are not spread out, a good estimate for $\bar{\mathcal{P}}$ may be obtained by computing $\sum_{i=1}^{N} w_i \mathcal{P}_i$ and projecting this result back onto the manifold (by normalizing

the vector, in the case $\mathcal{M} = S^1$ or S^2). However, a better estimate may in general be obtained by using an iterative algorithm in which an initial guess for $\bar{\mathcal{P}}$ is made and refined over each step by computing the weighted average deviation from this guess to each sigma point $\mathcal{P}_i$ and taking a step in that direction. The logarithm map on $\mathcal{M}$ is used to compute the weighted average deviation, and the exponential map is used to travel along the manifold in the direction of the deviation. The log map $\log_{\mathbf{p}} : \mathcal{M} \to T_{\mathbf{p}}\mathcal{M}$ is the inverse of the exponential map $\exp_{\mathbf{p}}$ in the sense that $\exp_{\mathbf{p}}(\log_{\mathbf{p}}(\mathbf{q})) = \mathbf{q}$ for all $\mathbf{q} \in \mathcal{M}$. It computes a tangent vector $\mathbf{v}$ that points in the direction of the geodesic from $\mathbf{p}$ to $\mathbf{q}$, with magnitude equal to the distance along that geodesic. The log map on S^2 is given in [1] and can easily be computed for S^1 by restricting the domain to $S^1 \subset S^2$.

D. Parallel Transport

The tangent vectors $\mathbf{t}_i$ must be parallel transported as the state evolves on $\mathcal{M}$. When $\mathcal{M} = S^2$, the parallel transport of $\mathbf{v} \in T_{\mathbf{p}}\mathcal{M}$ from $\mathbf{p} \in \mathcal{M}$ to $\mathbf{q} \in \mathcal{M}$ depends on the path from $\mathbf{p}$ to $\mathbf{q}$. However, when $\mathcal{M} = S^1$, the parallel transport only depends on $\mathbf{q}$. The parallel transport on S^2 is discussed in [1], [11]. For S^1, there is only one possible tangent vector that can be obtained by transporting $\mathbf{v}$ through any path from $\mathbf{p}$ to $\mathbf{q}$.

E. Computing Covariances

Now that we have discussed the computation of the mean of sigma points $\mathcal{P}_i$, we can discuss the computation of the covariance. The covariance is interpreted with respect to the tangent basis $\{\mathbf{t}_1, \ldots, \mathbf{t}_m\}$. Specifically, it is:

$$\Sigma_{\mathcal{P}} = \sum_{i=1}^{N} w_i \boldsymbol{\delta}_i \boldsymbol{\delta}_i^\top, \tag{19}$$

where, if $T = \begin{pmatrix}\mathbf{t}_1, \ldots \mathbf{t}_m\end{pmatrix}$,

$$\boldsymbol{\delta}_i = T^\top \log_{\bar{\mathcal{P}}}(\mathcal{P}_i). \tag{20}$$

Equation (20) is just resolving $\log_{\bar{\mathcal{P}}}(\mathcal{P}_i)$ in the components of the tangent basis via taking inner products. It should be noted that if the covariance is being computed in the state space, the tangent basis is the one that has been parallel transported along the state trajectory, but if it is computed in the measurement space, the tangent basis may be arbitrarily chosen. However, one must be consistent, as the tangent basis will be used again for the measurement update.

F. Measurement Update on $\mathcal{M}$

If $\mathbf{y} \in \mathcal{M}$ is the measurement and $\mathbf{h}(\hat{\mathbf{x}}) \in \mathcal{M}$ is the predicted measurement corresponding to the state estimate $\hat{\mathbf{x}}$, then we can update the state to $\hat{\mathbf{x}}^+$ as follows:

$$\hat{\mathbf{x}}^+ = \exp_{\hat{\mathbf{x}}}(\Delta\mathbf{x}),\ \Delta\mathbf{x} = KT^\top \log_{\mathbf{h}(\hat{\mathbf{x}})}(\mathbf{y}), \tag{21}$$

where $K = \Sigma_{xy}\Sigma_{yy}^{-1}$ is the Kalman gain and T is the matrix of the same tangent basis vectors that were used to compute the measurement covariance Σ_{yy} and the cross covariance Σ_{xy}.

Note that the exponential map on the state space means that we must sum the component of the state vector that corresponds to Euclidean space (the usual notion of sum is the exponential map for Euclidean space) and that we must use the corresponding exponential map for any component of the state vector that lives on a non-Euclidean manifold.

IV. Vision Pipeline

This section details the image processing pipeline used for optional yaw estimation. The advantage of this pipeline is that the velocity and tilt estimates from our filter allow us to compute an estimate of the rigid body motion between the keyframe and the current frame. This hypothesis is not obtained using RANSAC, and can be immediately used to reject outliers, saving computational resources. Figure 3 shows an overview of the vision pipeline and its interaction with the UKF. In this figure, P_{curr} is a list of the image features in the current frame, P_{prev} contains the features in the previous frame, and P_{key} contains the features in the keyframe.

A. Feature Tracking

We track FAST features [12] from the image in the previous camera frame, denoted $\mathcal{PC}$, to the image in the current camera frame, denoted $\mathcal{CC}$, using a KLT tracker [13] with pyramids. The initial estimates for the tracked features in the current frame are obtained by multiplying the normalized and undistorted homogeneous coordinates of the features in the previous frame by the rotation matrix $R_{\mathcal{PC}}^{\mathcal{CC}} = R_{\mathcal{CC}}^{\mathcal{W}\top} R_{\mathcal{PC}}^{\mathcal{W}}$, which can be computed from the filter outputs at the previous and current frames. The resulting feature coordinates are renormalized, distorted, and multiplied by the matrix of camera intrinsics to obtain the image coordinates.

B. Yaw Estimation

To estimate the yaw, we used the linear five point algorithm with known tilt, which was originally proposed in [14]. This algorithm uses point correspondences in two images, as well as the known tilts in the two angles, to estimate the translation, up to a scale, and the full orientation of the camera between the two images. This is done by decomposing the essential matrix as the product of a skew-symmetric matrix and an elementary rotation matrix and taking advantage of the resulting structure to simplify what would be the eight-point linear algorithm [15].

Although we track images from the previous frame to the current frame, we use the correspondences between the current frame and a fixed keyframe to estimate the motion. This helps to minimize the drift in the yaw estimate. The keyframe camera frame is denoted $\mathcal{KC}$. The keyframe is only updated when the number of tracked features falls below a specified threshold. This is common practice.

Without loss of generality, we will assume that the origins of the camera and body frames are coincident, but that the orientation of the camera frame relative to the body frame is given by the matrix R_{cam}. The epipolar constraint is given:

$$\mathbf{x}_{i\mathcal{KC}}^\top [\mathbf{t}_{\mathcal{CC}}^{\mathcal{KC}}]_\times R_{\mathcal{CC}}^{\mathcal{KC}} \mathbf{x}_{i\mathcal{CC}} = 0, \tag{22}$$

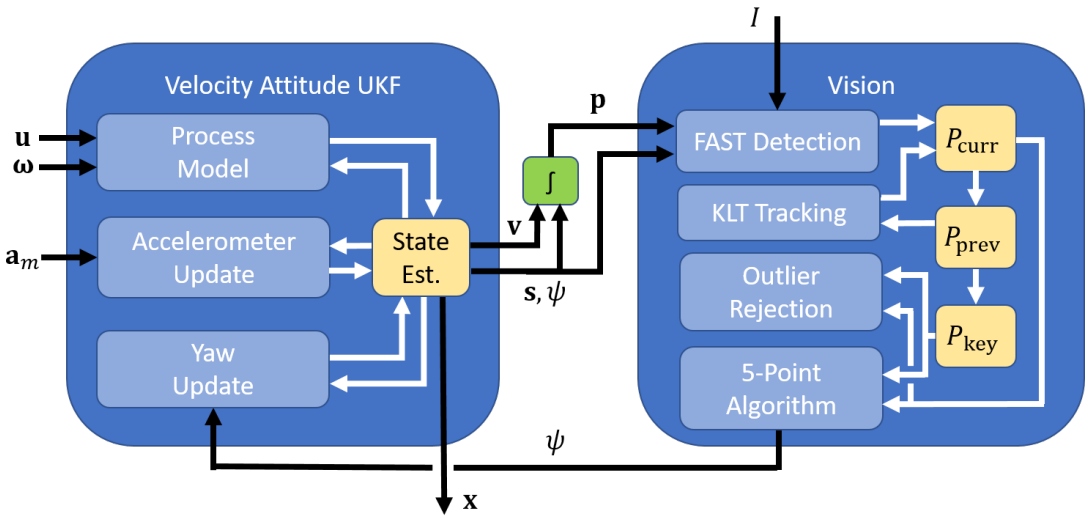


Fig. 3: Overview of the architecture of our state estimator.

where $\mathbf{x}_{i\mathcal{KC}}$ and $\mathbf{x}_{i\mathcal{CC}}$ are the homogeneous normalized coordinates of the ith tracked feature in the camera frame for the keyframe and the current frame, respectively. Now, we have:

$$R_{\mathcal{CC}}^{\mathcal{KC}} = R_{\mathcal{C}}^{\mathcal{B}\top} R_{\mathcal{KB}}^{\mathcal{W}\top} R_{\mathcal{CB}}^{\mathcal{W}} R_{\mathcal{C}}^{\mathcal{B}}, \tag{23}$$

where $\mathcal{KB}$ is the body frame at the time of the keyframe image and $\mathcal{CB}$ is the body frame at the time of the current image. But $R_{\mathcal{CC}}^{\mathcal{KC}}$ may be decomposed into a tilt rotation matrix and a yaw rotation matrix:

$$R_{\mathcal{CC}}^{\mathcal{KC}} = R_{\mathcal{CC},\psi}^{\mathcal{KC}} R_{\mathcal{CC},\phi}^{\mathcal{KC}}. \tag{24}$$

This results in a modified epipolar constraint:

$$\mathbf{x}_{i\mathcal{KC}}^{\top} E \tilde{\mathbf{x}}_{i\mathcal{CC}}^{\top} = 0, \tag{25}$$

where $E = [\mathbf{t}_{\mathcal{CC}}^{\mathcal{KC}}]_{\times} R_{\mathcal{CC},\psi}^{\mathcal{KC}}$ and $\tilde{\mathbf{x}}_{i\mathcal{CC}}^{\top} = R_{\mathcal{CC},\phi}^{\mathcal{KC}} \mathbf{x}_{i\mathcal{CC}}$. We can analyze the structure of the essential matrix:

$$\begin{aligned} E &= \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \cdot \begin{pmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} -t_z s\psi & -t_z c\psi & t_y \\ t_z c\psi & -t_z s\psi & -t_x \\ -t_y c\psi + t_x s\psi & t_x c\psi + t_y s\psi & 0 \end{pmatrix}, \end{aligned} \tag{26}$$

where $s\psi$ and $c\psi$ are abbreviations for $\sin\psi$ and $\cos\psi$, respectively. The structure of the essential matrix is:

$$E = \begin{pmatrix} E_1 & E_2 & E_3 \\ -E_2 & E_1 & E_4 \\ E_5 & E_6 & 0 \end{pmatrix}. \tag{27}$$

This observation does not take advantage of the full structure of the essential matrix, but it is enough to devise a simple linear algorithm to back out what each entry should be. This algorithm rewrites (25) as a linear equation of the form $A\mathbf{e} = 0$, where $\mathbf{e} = \begin{pmatrix} E_1 & E_2 & \dots & E_6 \end{pmatrix}^{\top}$, and uses a singular value decomposition to determine the entries of E up to a scale. Thus, the algorithm only requires 5 points, but more points may be used. For more information, refer to [14].

Upon determining the entries of E, we are not guaranteed that the resulting matrix is the product of a skew-symmetric matrix with an elementary yaw rotation matrix. In fact, we are not even guaranteed that the result is an essential matrix. Hence, we project the matrix onto essential matrix space and then decompose it into the product of a rotation matrix with a skew symmetric matrix [15]. There are two possible rotations. To determine the correct rotation, we pick the rotation matrix that minimizes $\text{trace}(I - \hat{R}_{\mathcal{CC}}^{\mathcal{KC}\top} R_{ci})$, where I is the 3×3 identity matrix, $\hat{R}_{\mathcal{CC}}^{\mathcal{KC}}$ is an estimate of $R_{\mathcal{CC}}^{\mathcal{KC}}$ obtained from the output of the UKF, and R_{ci} is the ith candidate solution for the rotation, where $i \in \{1, 2\}$. Finally, we decompose the rotation matrix into the product of an elementary yaw matrix with a tilt matrix (the tilt should be close to the identity). At this point, we have recovered $R_{\mathcal{CC},\psi}^{\mathcal{KC}}$, so we may determine the yaw of the body frame relative to the world frame by reconstructing $R_{\mathcal{CC}}^{\mathcal{KC}}$ using (24) and then solving for $R_{\mathcal{CB}}^{\mathcal{W}}$ in (23).

C. Outlier Rejection

The filter publishes a position estimate in the world frame, which is obtained by rotating the velocity into the world frame and applying a first-order Euler integration. The position estimate is therefore prone to drift. However, when computing the estimate $\hat{\mathbf{t}}_{\mathcal{CC}}^{\mathcal{KC}}$ of the position of the camera in the current frame relative to the keyframe, it is the relative position that matters. From our experiments, the position output of our filter is good enough to compute the motion hypothesis between the current frame and the keyframe. In addition, the filter outputs the attitude at each frame, so we can compute the attitude estimate $\hat{R}_{\mathcal{CC}}^{\mathcal{KC}}$ of the current frame relative to the keyframe. Using the estimate of the rigid body transform, we may compute the epipoles:

$$\mathbf{e}_{\mathcal{KC}} = \hat{\mathbf{t}}_{\mathcal{CC}}^{\mathcal{KC}}, \tag{28}$$

$$\mathbf{e}_{\mathcal{CC}} = -R_{\mathcal{KC}}^{\mathcal{CC}} \hat{\mathbf{t}}_{\mathcal{CC}}^{\mathcal{KC}}, \tag{29}$$

where $\mathbf{e}_{\mathcal{KC}}$ is the camera center in the current frame projected into the keyframe and $\mathbf{e}_{\mathcal{CC}}$ is the camera center in the keyframe

projected into the current frame. Assuming the camera doesn't undergo purely lateral translation, these points will not be at infinity and we can compute the distance from the epipoles to the epipolar lines. First, we compute the epipolar line coefficients ℓ_{ij} for the ith correspondence in the jth frame, where $j \in \mathcal{J} = \{\mathcal{KC}, \mathcal{CC}\}$:

$$\ell_{ij} = \mathbf{e}_j \times R^j_{j'} \mathbf{x}_{ij'}, \tag{30}$$

where $j' = k \in \mathcal{J}$ such that $j \neq k$. Finally, we compute the distance d_{ij} from the epipolar line to the point $\mathbf{x}_{ij}$ $\forall i$:

$$d_{ij} = \frac{|\ell_{ij}^\top \mathbf{x}_{ij}|}{\sqrt{a_{ij}^2 + b_{ij}^2}}, \tag{31}$$

where $\ell_{ij} = \begin{pmatrix} a_{ij} & b_{ij} & c_{ij} \end{pmatrix}^\top$. Finally, we threshold $d_{i\mathcal{KC}} + d_{i\mathcal{CC}}$ to determine whether the ith point correspondence is an outlier. This outlier rejection scheme is very fast and allows us to completely avoid time consuming random sampling methods such as RANSAC. This is a key contribution.

V. Observability Analysis

We now outline the analysis used to determine whether the nonlinear system given by equations (13) through (14), $\dot{\mathbf{b}} = \boldsymbol{\eta}_b$ and $\dot{\mathbf{k}} = \mathbf{0}$, and (16) is observable. We have factored out the yaw from the system, since the system has been shown to be independent of the yaw. The state vector of this system is

$$\mathbf{x}_{\text{obs}} = \begin{pmatrix} \mathbf{s}^\top & \mathbf{v}^\top & \mathbf{b}^\top & \bar{\mathbf{k}}^\top \end{pmatrix}^\top, \tag{32}$$

where $\bar{\mathbf{k}} = \begin{pmatrix} \bar{k}_f & \bar{k}_d & \bar{k}_z \end{pmatrix}^\top$ is the vector of drag coefficients normalized by the mass. That is, $\bar{k}_i = k_i/m$ for $i \in \{f, d, z\}$. Also, $\mathbf{s} = \begin{pmatrix} s_1 & s_2 \end{pmatrix}^\top$ is a vector of the stereographic coordinates corresponding to the vector $\mathbf{z} \in S^2$. These coordinates are used to give a minimal parameterization of the tilt for the analysis. The stereographic coordinates are expressed, as a function of $\mathbf{z}$:

$$s_1 = \frac{z_1}{1 + z_3}, \quad s_2 = \frac{z_2}{1 + z_3}. \tag{33}$$

We perform the observability analysis by checking the observability rank criterion, a technique developed in [16] and exemplified in [17]. This method involves writing the system in control affine form:

$$\dot{\mathbf{x}}_{\text{obs}} = \mathbf{f}_0(\mathbf{x}_{\text{obs}}) + \sum_{i=1}^{m} \mathbf{f}_i(\mathbf{x}_{\text{obs}}) u_i. \tag{34}$$

Once this is done, the unforced vector field $\mathbf{f}_0$ and the control input vector fields $\mathbf{f}_i$ are used to take Lie derivatives of the measurement function $\mathbf{h}(\mathbf{x}_{\text{obs}}, u_{i\in\{1,\dots,m\}})$. The observability matrix $\nabla_{\mathbf{x}_{\text{obs}}} \mathcal{O}$ is obtained by stacking these Lie derivatives and taking the gradient with respect to the state vector. Finally, the rank of the observability matrix is computed. A criterion for the observability of the system is that the rank of $\mathcal{O}$ must be the dimension of the state space, which is 11 in our case. We outline the procedure taken for the observability analysis and demonstrate the computations of some of the Lie derivatives, but we do not present all Lie derivatives or the observability matrix because the expressions are unwieldy. To see these calculations, refer to the Mathematica notebook [1].

We choose only one control input, that being u_{ss} because it is proportional to the thrust. The moments are not selected as inputs because they do not appear in the model. In control affine form, our system looks like:

$$\dot{\mathbf{x}}_{\text{obs}} = \mathbf{f}_0(\mathbf{x}_{\text{obs}}) + \mathbf{f}_1(\mathbf{x}_{\text{obs}}) u_{ss}, \tag{35}$$

$$\mathbf{f}_0(\mathbf{x}_{\text{obs}}) = \begin{pmatrix} \frac{1}{2}(\omega_2(1 + s_1^2 - s_2^2) + 2\omega_3 s_2 - 2\omega_1 s_1 s_2) \\ \frac{1}{2}(\omega_1(s_1^2 - s_2^2 - 1) - 2\omega_3 s_1 + 2\omega_2 s_1 s_2) \\ \frac{2gs_1}{1+s_1^2+s_2^2} - \bar{k}_d u_s v_1 + \omega_3 v_2 - \omega_2 v_3 \\ \frac{2gs_2}{1+s_1^2+s_2^2} - \bar{k}_d u_s v_2 - \omega_3 v_1 + \omega_1 v_3 \\ \frac{(s_1^2+s_2^2-1)g}{1+s_1^2+x_y^2} - \bar{k}_z u_s v_3 + \omega_2 v_1 - \omega_1 v_2 \\ \mathbf{0}_{6\times 1} \end{pmatrix}, \tag{36}$$

$$\mathbf{f}_1 = \begin{pmatrix} \mathbf{0}_{1\times 4} & \bar{k}_f & \mathbf{0}_{1\times 6} \end{pmatrix}^\top, \tag{37}$$

$$\mathbf{h}(\mathbf{x}_{\text{obs}}, u_{ss}) = \bar{k}_f u_{ss} \mathbf{e}_3 - u_s \bar{D} \mathbf{v} + \mathbf{b}, \tag{38}$$

where $\bar{D} = \text{diag}\begin{pmatrix} \bar{k}_d & \bar{k}_d & \bar{k}_z \end{pmatrix}$. The first order Lie derivative of $\mathbf{h}$ with respect to $\mathbf{f}_0$ is:

$$L_{\mathbf{f}_0} \mathbf{h} = \nabla_{\mathbf{x}_{\text{obs}}} \mathbf{h} \cdot \mathbf{f}_0 \tag{39}$$

$$= \begin{pmatrix} -\bar{k}_d u_s \left(\frac{2gs_1}{1+s_1^2+s_2^2} - \bar{k}_d u_s v_1 + \omega_3 v_2 - \omega_2 v_3 \right) \\ -\bar{k}_d u_s \left(\frac{2gs_2}{1+s_1^2+s_2^2} - \bar{k}_d u_s v_2 - \omega_3 v_1 + \omega_1 v_3 \right) \\ -\bar{k}_z u_s \left(\frac{(s_1^2+s_2^2-1)g}{1+s_1^2+x_y^2} - \bar{k}_z u_s v_3 + \omega_2 v_1 - \omega_1 v_2 \right) \end{pmatrix}, \tag{40}$$

and the first order Lie derivative of $\mathbf{h}$ with respect to $\mathbf{f}_1$ is:

$$L_{\mathbf{f}_1} \mathbf{h} = \nabla_{\mathbf{x}_{\text{obs}}} \mathbf{h} \cdot \mathbf{f}_1 = \begin{pmatrix} 0 & 0 & -k_z k_f u_s \end{pmatrix}^\top. \tag{41}$$

The second order Lie derivatives are computed:

$$L_{\mathbf{f}_0 \mathbf{f}_0} \mathbf{h} = (\nabla_{\mathbf{x}_{\text{obs}}} L_{\mathbf{f}_0} \mathbf{h}) \cdot \mathbf{f}_0, \tag{42}$$

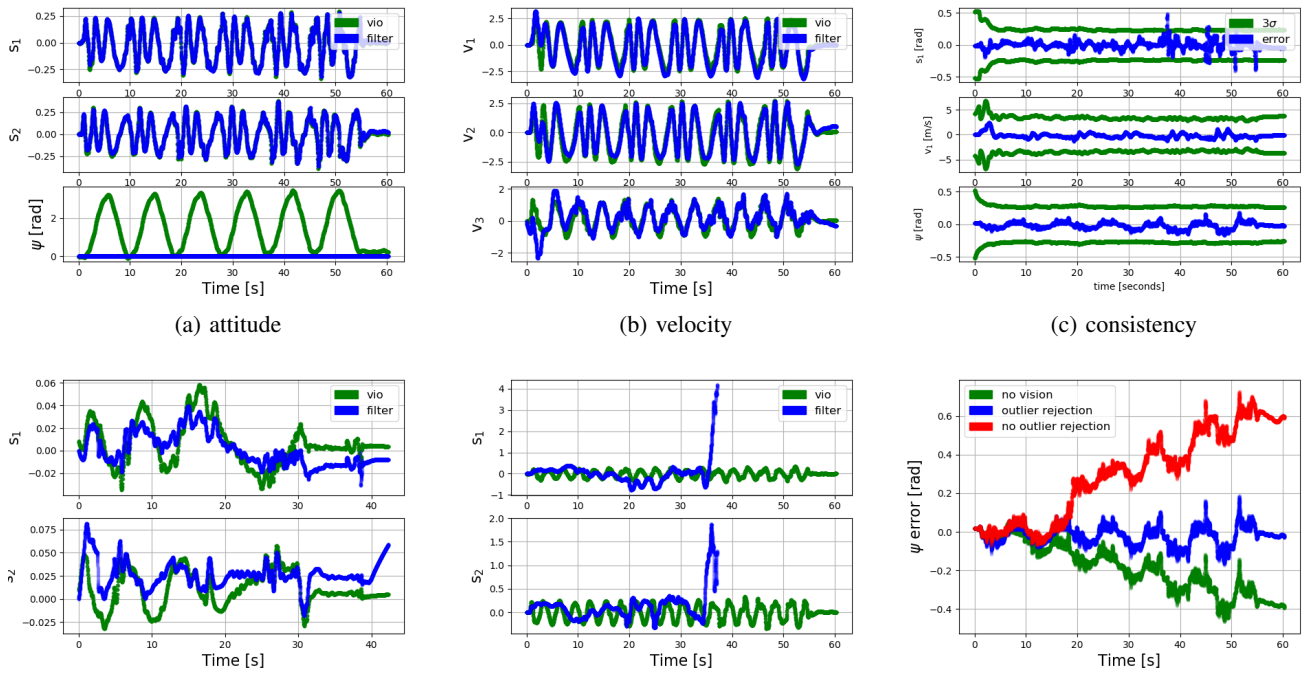
$$L_{\mathbf{f}_0 \mathbf{f}_1} \mathbf{h} = (\nabla_{\mathbf{x}_{\text{obs}}} L_{\mathbf{f}_0} \mathbf{h}) \cdot \mathbf{f}_1. \tag{43}$$

These lie derivatives, along with $\mathbf{h}$ itself (which is the zeroth order Lie derivative of itself) are stacked vertically into a vector $\mathcal{O}$. Finally, the gradient $\nabla_{\mathbf{x}_{\text{obs}}} \mathcal{O}$ is computed. If this matrix is rank 11, then the observability rank criterion is satisfied and the system is, in general, observable. This is the case. However, under certain conditions the matrix has a rank deficiency. Some of these cases are when $\omega_1 = \omega_2 = 0$ and $s_1 = s_2 = \omega_3 = 0$. This could explain why the system seems to be unobservable in hover. In this case, both of the above conditions are satisfied, meaning that the system may not be observable. However, further analysis is required since the observability rank criterion is only a sufficient condition.

VI. Experimental Results

In this section, we report on the experiments performed at the PERCH lab (Penn Engineering Research Collaborative Hub) at the University of Pennsylvania indoor testbed. The total flying area volume is $20 \times 6 \times 4$ m^3. Experiments were performed using a platform based on our previous contribution [6], as shown in Figure 1. The high-level control was

[1] https://github.com/jsvacha/observability_analysis



(a) attitude (b) velocity (c) consistency

(d) slow trajectory (e) tilt-yaw convention (f) yaw error

Fig. 4: (a) Attitude and (b) velocity of our filter compared to visual odometry for a 60 second run without using vision and forcing the yaw estimate to zero. (c) s_1, v_1 and ψ errors and 3σ bounds without using vision. Tilt estimate of our filter when (d) the speed is too low and (e) the tilt-yaw convention is used with unknown yaw. (f) The yaw error without using vision, using vision with outlier rejection, and using vision without outlier rejection.

TABLE I: Root mean square and standard deviation of velocity and attitude errors with and without using vision.

	v_1	v_2	v_3	$\|\mathbf{v}\|$	s_1	s_2	$\|\mathbf{s}\|$	yaw
	without vision							
RMS	0.465	0.454	0.299	0.715	0.0664	0.0625	0.0911	0.241
STD	0.442	0.444	0.284	–	0.0664	0.0623	–	0.0850
	with vision							
RMS	0.457	0.454	0.298	0.710	0.0658	0.0624	0.0907	0.0621
STD	0.423	0.442	0.285	–	0.0658	0.0624	–	0.0534

TABLE II: Average, median, and maximum times for each major step in the UKF and vision pipeline, in milliseconds.

	UKF			**vision**					
	process	acc meas	yaw meas	detection	pyramid	tracking	outlier	5 point	total
avg (i7)	0.0558	0.0333	0.0370	1.14	2.26	0.410	0.0110	0.136	3.44
med (i7)	0.0566	0.0330	0.0370	1.04	2.22	0.396	0.0110	0.142	2.94
avg (arm)	0.265	0.155	0.128	4.81	13.1	2.16	0.0209	0.408	18.3
med (arm)	0.166	0.0949	0.0927	4.45	12.0	2.09	0.0181	0.403	16.8

performed onboard. Each motor speed is measured through the available SDK[2], and is obtained by dividing the voltage signal switching frequency by the number of poles. The results are compared to the on board visual inertial odometry used in [6], which has been shown to have similar accuracy to a motion capture system and is therefore considered ground truth.

Figures 4a and 4b show the tilt and velocity estimates for our filter over a 60 second run of a lissajous trajectory without using vision for yaw estimation. To demonstrate that this filter works without yaw estimation, we forced the yaw estimate to zero. During the trajectory the quadrotor reached speeds of up to 3.2 m/s. Velocity is expressed in the body-fixed frame.

Figure 4c shows the estimation errors for s_1, v_1 and ψ, as well as bounds that represent three standard deviations as determined by the filter when vision was not used to estimate the yaw. This plot shows that the estimation error remains mostly within the bounds and that the estimation errors are zero-mean, indicating the consistency of the filter.

Figures 4d and 4e show two cases when the tilt estimation does not work well. The first case is when the trajectory is too slow, and hence when the system is too close to hover. In this case, the previous trajectory was slowed down so that the maximum velocity was around 1.2 m/s. The second case

[2]https://developer.qualcomm.com/hardware/qualcomm-flight/tools

is when the tilt-yaw convention is used instead of the yaw-tilt convention, and the yaw is unknown. In this case, the yaw estimate was forced to zero. This run demonstrates that the filter using the tilt-yaw convention cannot estimate the tilt if the yaw is unknown, as the filter diverges before 40 seconds.

Figure 4f shows the yaw error, in radians, of the filter for three cases: when not using vision, when using vision but no outlier rejection, and when using vision with outlier rejection to estimate the yaw for the same dataset. When vision wasn't used, the IMU was dead-reckoned to estimate the yaw. Notice that using vision to estimate the yaw can result in a significant reduction of the yaw error drift. Furthermore, choosing not to reject outliers can cause the state estimate to drift.

The root mean squares and standard deviations of the errors for the velocity and attitude from $t = 20$ seconds to $t = 50$ seconds are listed in Table I. The yaw and tilt errors were computed in the tangent space, so they may be interpreted in radians. Table I shows that using vision to estimate the yaw affects the estimation of the tilt and velocity as well. This is to be expected since (12) shows that the derivative of the yaw is a function of the tilt, so there is still a coupling and the yaw does provide information about the other states. This information is incorporated only during the yaw measurement update, so if vision is not used, the yaw state does not affect any of the other states. For these reasons, we used one filter to estimate everything instead of estimating the yaw with a separate filter.

The UKF and vision pipeline were used to postprocess the data on a laptop with an Intel Core i7 processor. The UKF was also run on the quadrotor platform, which has an ARM processor. The average and median times required to complete each major step are shown in Table II. For the UKF, the steps are the process model, accelerometer measurement, and yaw measurement. For the vision pipeline, the steps are FAST feature detection, creating the image pyramid, KLT tracking, outlier rejection, and the 5-point algorithm. Since the IMU rate is at 100 Hz and the camera is 30 frames per second, we have 10 ms to perform the process update and 33 ms to perform the measurement update making the approach suitable for small scale, computationally limited platforms.

We performed trials at other speeds, giving similar results. Please refer to the video [3] to observe these experiments.

VII. Conclusion

In this work, we developed a novel UKF to estimate the velocity and attitude of a quadrotor during high speed flight. In addition, this pipeline estimates accelerometer biases and drag and thrust coefficients for the system model. We also developed a novel vision pipeline to estimate the yaw of the quadrotor in a way that significantly reduces the drift obtained by dead reckoning based on the gyros. This pipeline uses the output of the UKF to estimate the hypothesis for the rigid body motion of the camera between frames, bypassing the classic RANSAC algorithm for outlier rejection. We demonstrated the performance of the filter by running it on experimental data obtained from a real quadrotor platform. We think that this work can benefit nano-scale quadrotors.

Future work will involve developing more robust Lyapunov observers to estimate these quantities. These observers should have better convergence guarantees than the UKF. In addition, work will be done to make some use out of the vision pipeline's translation direction estimate, which is currently discarded. Finally, we wish to improve the fidelity of the observer by adding Eulerian dynamics for the rotation.

[3]https://www.youtube.com/watch?v=TRDcZS5fb9I

References

[1] J. Svacha, K. Mohta, M. Watterson, G. Loianno, and V. Kumar, "Inertial velocity and attitude estimation for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–9.

[2] J. Hodgson, S. Baylis, R. Mott, A. Herrod, and R. Clarke, "Precision wildlife monitoring using unmanned aerial vehicles," *Nature - Scientific Reports*, Mar. 2016.

[3] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 46–56, Sept 2012.

[4] T. Ozaslan, G. Loianno, J. Keller, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, "Autonomous navigation and mapping for inspection of penstocks and tunnels with MAVs," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1740–1747, July 2017.

[5] S. Shen, N. Michael, and V. Kumar, "Tightly-Coupled Monocular Visual-Inertial Fusion for Autonomous Flight of Rotorcraft MAVs," in *IEEE International Conference on Robotics and Automation*, Seattle, USA, 2015.

[6] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, April 2017.

[7] D. Abeywardena, S. Kodagoda, G. Dissanayake, and R. Munasinghe, "Improved state estimation in quadrotor mavs: A novel drift-free velocity estimator," *IEEE Robotics & Automation Magazine*, vol. 20, Jun. 2013.

[8] R. Leishman, J. C. Macdonald, Jr., R. Beard, and T. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems Magazine*, vol. 34, Feb. 2014.

[9] G. Allibert, D. Abeywardena, M. Bangura, and R. Mahony, "Estimating body-fixed frame velocity and attitude from inertial measurements for a quadrotor vehicle," in *2014 IEEE Conference on Control Applications (CCA)*, Oct 2014, pp. 978–983.

[10] C. Hertzberg, R. Wagner, U. Frese, and L. Schrder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 41, no. 1, pp. 57–77, 2013.

[11] J. Svacha, K. Mohta, M. Watterson, G. Loianno, and V. Kumar, "Inertial velocity and attitude estimation for quadrotors: Supplementary material," Feb. 2018. [Online]. Available: http://repository.upenn.edu/ese_papers/834

[12] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, vol. 1, May 2006, pp. 430–443.

[13] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, Jun 1994, pp. 593–600.

[14] F. Fraundorfer, P. Tanskanen, and M. Pollefeys, "A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 269–282.

[15] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

[16] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, Oct. 1977.

[17] A. Martinelli, "State estimation based on the concept of continuous symmetry and observability analysis: the case of calibration," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 239–255, April 2011.