Pose and Motion Estimation of Free-Flying Objects: Aerodynamics, Constrained Filtering, and Graph-Based Feature Tracking

Matthew Gardner, Member, IEEE and Yan-Bin Jia, Senior Member, IEEE

Abstract—We investigate the problem of dynamically estimating the instantaneous position, orientation, velocity, and angular velocity of an arbitrarily shaped object during its free flight based on image frames taken simultaneously by two high-speed cameras. Aerodynamic effects including drag, lift, and Magnus forces are modeled to describe the object’s flight. Observables are derived from combining dynamics with a camera projection model assuming two-view geometry, via the use of multiple quaternions. The state of the composed system can then be estimated via constrained Kalman filtering, to which a solution is presented for the case of multiple quadratic constraints. To keep track of appearing and disappearing visual features during flight, the estimation algorithm employs a graph matching-based technique to maintain a set of evolving hypotheses through evaluation, pruning, and addition. Experiments conducted over various objects have either provided validation against motions independently estimated using multiple accelerometers, or formed verification by matching flight images against projections based on the state estimates.

Index Terms—Estimation, aerodynamics, Kalman filtering, constrained optimization, image graph, feature tracking.

I. INTRODUCTION

Motion estimation of free-flying objects is a challenging problem in multiple areas of robotics. Pose and motion estimates are necessary for sensing obstacles, planning actions, and navigating autonomously in dynamically changing environments. In space, tumbling objects such as debris and asteroids are tracked using vision, with motions estimated via Kalman filtering [1] and optimization [2]. Flight trajectory estimation can help space robots better collect space debris [3], [4], service orbiting satellites [5], and make maneuvers during space exploration [6], [7]. For an object flying in the air, the ability to obtain accurate estimates of its pose and velocity over a brief duration allows for a robot to perform skillful maneuvers such as catching [8] and batting [9], [10], [11]. In sports, such estimation is also used to track a ball for the purpose of human training [12]. In manufacturing, a potential application of motion estimation could be enabling robotic arms to pass objects down an assembly line through catching and throwing motions. Other applications include projectile tracking, robotic underwater exploration, and virtual reality.

Measurement of position and linear velocity has been used for a range of tasks. The position trajectory of a flying object was estimated from a sequence of monocular images for object tracking [13], by stereo cameras for a high-speed ping-pong robot [11], [14], and by two high-speed cameras on pan-tilt mechanisms for batting a ball [15]. As the complexity of robot applications has grown, estimation of angular velocity has become increasingly necessary. Measurements by linear accelerometers placed inside a body along some orthogonal axes were used to calculate the body’s angular velocity via solution of a system of kinematic equations [16] or through optimization [17], [18]. A magnetometer, accelerometer, and angular rate sensor were employed to obtain quaternion-based measurements, which were then supplied to a Kalman filter for polishing [19]. These techniques, however, were invasive as they relied on sensors mounted inside the object’s body. In addition, none of them addressed how to estimate the object’s linear velocity. Meanwhile, vision-based estimation of both linear and angular velocities was investigated for object tracking [20], [21], but without the temporal and spatial constraints of the object being in free flight.

Vision sensors such as RGB cameras are widely accessible and easy to work with, making them ideal for the estimation problem. They can be made small enough to serve as eyes for a humanoid robot. Can fast and accurate motion estimation of a flying object be achieved using cameras alone? Here we put an emphasis on the object in a free flight with no actuation. Several challenges immediately arise:

- images of the object lack depth information,
- the object moves a large distance over a small duration,
- images are susceptible to noise due to the fast movement.

Some vision systems have partially addressed these challenges. Among the best known are motion capture systems composed of multiple high-speed cameras surrounding an object with attached markers [22], or actuated cameras that control their pitch and yaw for tracking [15]. Such systems are complex and often infeasible due to limitations of cost, space, power consumption, etc. Moreover, they depart from a basic preference for estimation without attaching markers or sensors to alter the object’s physical appearance.

A. Contributions and Outline of the Paper

In this work, we present a scheme for accurately estimating the poses and motions of free-flying objects while remaining
robust to the effects of short flight periods, large aerodynamic forces experienced in the case of low mass density, and image noise due to fast movements. We will use a system which consists of two cameras with wide angle lenses and operating at 120 frames per second (fps) or higher, and is inexpensive relative to larger motion capture systems such as OptiTrack and Vicon. Besides achieving estimation, we make several other contributions:

a) We introduce a graph-based feature tracking algorithm using hypotheses to deal with uncertainties and switching of observable features.

b) We construct for the first time a solution, via equation derivation and numerical root finding, to Kalman filtering under multiple quadratic constraints in the general form.

c) We model aerodynamics for free flying objects in general shapes and integrate it with vision-based estimation.

Fig. 1 illustrates the interactions among three components of the scheme: aerodynamics, constrained Kalman filtering, and graph-based feature tracking. Extensive experiments are conducted with a relatively low-cost vision system, which past efforts have not considered.

The paper is organized as follows. Section II overviews work related to estimation that comes from multiple areas. Section III discusses the system dynamics of a free-flying object and computation of the aerodynamic forces affecting its motion. Section IV introduces the camera projection model and geometric constraints enforced by stereo cameras to produce observables from the object. Section V gives the Kalman filter algorithm that enforces multiple quadratic constraints on the estimated state while making use of the models from Sections III and IV. Section VI describes a graph-based feature tracking algorithm for identifying image-to-model correspondences that are required to drive multiple Kalman filters in a competition for the best estimate. Section VII presents results from three experiments of increasing complexity, conducted with a wooden frame, rugby ball, and foam polyhedron. Section VIII concludes the paper with discussions on improvements and future directions for dynamic estimation.

B. Notation

A vector (or a point) is by default a column vector, written by a lowercase bold letter, e.g., \( \mathbf{v} \). The left superscript of a point (or a vector), if exists, denotes the frame in which it is expressed. For example, \( \mathbf{p} \) gives the coordinates of a point \( p \) in the world frame (denoted by \( \mathbf{w} \)), while \( \mathbf{v} \) gives its coordinates in the body frame (denoted by \( \mathbf{b} \)). A scalar (or vertex in Section VI) is written as a lowercase non-bold letter, and a matrix (or graph) as an uppercase non-bold letter. A vector function is written in bold, while a scalar function is non-bold. The subscripts \( x, y, \) and \( z \) of a letter (non-bold) represent the respective \( x, y, \) and \( z \)-coordinates (or components) of a point (or a vector) named by the same letter (bold). For example, \( p_x \) denotes the \( x \)-coordinate of the point \( p \). Multiplication of two quaternions uses the operator \( \otimes \). In a slight abuse of notation, multiplication of a quaternion \( q \) and a vector \( \mathbf{v} \), i.e. \( q \otimes \mathbf{v} \), implies that \( \mathbf{v} \) is represented as the pure quaternion \( (0, \mathbf{v}^\top) \). In addition, the superscripts ‘−’ and ‘+’ are used to refer to the prior and posterior estimates produced by a Kalman filter. An estimate of a state \( x \) is denoted as \( \hat{x} \).

II. RELATED WORK

Our effort draws upon works from a diverse set of related fields: aerodynamics, feature tracking, Kalman filtering, and simultaneous localization and mapping (SLAM).

A. Aerodynamics

Aerodynamics are known to play a key role in affecting a flight motion. Calculation of forces such as air drag and lift results in more accurate estimates, as done for an aircraft’s motion using a downward-facing camera [23]. In robot table tennis, the effects of drag and Magnus were incorporated into estimation of a ping-pong ball’s motion with varying degrees of spin [24], [25]. Unfortunately, closed forms for lift and drag forces so far exist only for well studied shapes such as a ball and airfoil [26, pp. 258–262, 349], and they do not generalize to irregular shapes such as a polyhedron. The estimation scheme in this paper is described for polyhedral objects whose topological structures can be represented by planar graphs, achieving a degree of generality as many objects can be approximated in this way. Note that machine learning techniques were used to produce models of arbitrarily shaped objects in flight under the effects of aerodynamics and other non-linear dynamics [22], based on training over numerous flight trajectories from laborious hand throws.

B. Feature Tracking

Coupled with motion estimation is the problem of tracking an object’s features in its changing images in order to generate observables for setting up an estimation algorithm. Disparities between features from observation and those from prediction are used to estimate the object’s pose via weighted least squares [27], [28], non-linear minimization [29], or statistical modeling [30], [31], [32]. Model-based pose tracking was applied to estimate the pose of a spacecraft with different ways of initialization [33]. In general, these approaches, by minimizing measurement errors, led to frame rates (up to 30 Hz) that were too low for high-speed tracking, and also suffered from erratic performance in case of large disparities between observed and predicted features. Some approaches [31], [34] have achieved high processing rates to deal with large motions. For tracking a free-flying object, these approaches fail for two main reasons. First, the object remains at a distance (about 3 m from camera...
sensors) and appears in images with low pixel count (about 100 px area). In contrast, previous works are often validated by experiments with one or more objects occupying large regions of the image. Second, instead of undergoing a slow translation, the object is both translating and rotating at high speeds (up to 6 m/s translation and 20 rad/s rotation) throughout the image. Consequently, motion blur in images poses a challenge.

To make use of the object’s topological structure, matching was conducted between a “projection” graph constructed from images with a model graph, where edges and vertices contained attributes pertaining to features [35]. Approximate graph matching algorithms were developed to produce sub-optimal solutions to object recognition, while remaining tolerant to measurement errors and image noise [36], [37]. Recently, deep neural networks were trained on synthetic data to track stationary objects with occlusions [38], [39].

C. Kalman Filtering

For motion estimation, the Kalman filter [40] remains the golden standard due to its effectiveness at smoothing out estimates in the presence of noisy measurements, as well as its computational efficiency. An extended Kalman filter (EKF) using dual quaternions closely calculated the constant velocity of an object from its images taken by a single camera [21]. A modified EKF was able to estimate the rigid body dynamics of a moving object by exploiting the epipolar constraint [23], [41], [42]. For localization and navigation of air and ground vehicles, visual inertial navigation systems (VINS) also made use of the epipolar constraint in the measurement models of a linear Kalman filter [43], an unscented Kalman filter (UKF) [44], and an EKF [45]. In [46], an alternative to the use of epipolar constraints was proposed. This approach derived a measurement model for every observed static feature to express geometric constraints over its observing cameras. State correction in the EKF was then based on a range space projection of measurement residuals over all features, which were independent of errors in the feature coordinates.

Unit quaternions, used for representing orientation (or attitude), cannot be updated through vector addition by a conventional Kalman filter because they are only closed under operations (i.e., multiplications) of the Lie group SO(3) with its elements represented by quaternions. To maintain the invariance of the rotation representation, the multiplicative extended Kalman filter (MEKF) from [47] first performed an EKF style update on a state that included the position and velocity components, as well as some unconstrained parameters. Then, on orientation update, these parameters were used to generate through normalization a unit error quaternion that measured the difference between the true orientation and the estimated one. The second-order version of MEKF was given in [48] for three types of unconstrained orientation parameters including Euler angles, Rodriguez parameters, and a quaternion. The MEKF has been used widely in applications such as inertial integrated navigation [49], attitude estimation [50], and integration with GPS [51]. Generally relying on on-board inertial sensors, the MEKF is difficult to tune and known to have divergence issue due to its ad hoc normalization to maintain the unit quaternion constraint.

The invariant extended Kalman filter (IEKF), first introduced in [52] for attitude estimation, leveraged the left invariance of the vector field describing the orientation-only kinematics of a flying object under the multiplication action of SO(3) with unit quaternion elements. Extended later for velocity estimation and to right invariance on a Lie group [53], the IEKF preserves the unit norm of the quaternion estimate, as well as “symmetry” in the sense of not altering the dynamics of error, whose component for orientation is in the multiplicative form as inspired by the MEKF. With a larger expected domain of convergence than EKF and guaranteed convergence for invariant observations [54], the IEKF has been popularized in applications such as spacecraft attitude estimation [53] and unmanned aerial vehicle navigation [55]. Such applications typically employ on-board sensors such as gyroscopes, accelerometers, and magnetometers that output state variable values directly (or after a Lie group action such as rotation) so the IEKF becomes applicable. In our estimation problem, an IEKF is inappropriate for three reasons. First, the rigid body dynamics now includes aerodynamic effects such as drag, which is quadratic in the velocity’s magnitude and also linear in its normalization, and lift, which is quadratic in the angular velocity. Consequently, the system dynamics no longer preserves invariance under multiplication by members of a Lie group such as SE(3). Second, since observations are performed by cameras under perspective projection and radial distortion, the equivariant property [53] for the output function does not hold either. Third, the object under estimation is meant to be natural or man-made with a small size, and it would be unrealistic and also defeat the purpose of this estimation to embed a gyroscope, accelerometer, or magnetometer inside the object. In fact, the constrained EKF to be introduced in Section IV will effectively keep the orientation estimate within SO(3). Interestingly, a recent finding [57] has shown that, when the state space is a Lie group, a version of the EKF with an equality constraint is equivalent to the IEKF.

A third approach to orientation estimation is constrained Kalman filtering which in principle can effectively deal with a general equality constraint. In this approach, the posterior estimate is still obtained from minimizing the variance of the estimated error but now with the constraint incorporated via the use of a Lagrange multiplier. This forces the state estimate to stay on the constraint surface, e.g., SO(3) for an orientation estimate. To our best knowledge, all existing methods have considered either single constraint that is linear, quadratic [58] or special quadratic (e.g., unit norm [59], generalized norm [60]), or multiple constraints that are linear [61] or nonlinear after linearization [62]. In [59] and [60], second order sufficient conditions were used to choose from multiple values of the Lagrange multiplier satisfying the first order necessary condition equation.

The gain constrained Kalman filter (GCKF) [62] was derived from constrained recursive least squares minimization of the Kalman gain matrix, which constrained weights used to correct the state estimate. The GCKF was generalized to a

---

1 In visual inertial SLAM where a monocular camera was used as one of the sensors, the inventor of the IEKF and his colleagues opted for an UKF. [56]
UKF with nonlinear models under linear constraints [63], and to a norm-constrained Kalman filter (NCKF) dealing with one nonlinear constraint on the state’s norm [59]. The NCKF was then extended to handle a single quadratic form constraint [60]. Other approaches included a smoothly constrained Kalman filter [64] and a two-step UKF that constrained the state’s statistics [65].

As far as we know, none of existing works have dealt with multiple nonlinear constraints directly without linearizing them. In this paper, we need to handle three unit quaternion constraints that correspond to the orientations of the object and the two cameras. We will present a solution to the constrained filtering that works for a set of general quadratic constraints.

D. Simultaneous Localization and Mapping

SLAM algorithms solve a similar problem of estimating the pose and motion of a robot relative to its surroundings by producing a map of features in the three-dimensional (3D) world space. MonoSLAM used a single camera with an EKF to estimate a robot’s pose, velocity, and angular velocity, along with feature points in 3D [66]. Other approaches tracked the image coordinates of features in the estimated state, and described a projection from 3D to two dimensions (2D) in the measurement model of an EKF [67] or UKF [68]. VINS uses vision and inertia sensors for real-time estimation required for robot navigation. A UKF was used to estimate the pose of an autonomous rotorcraft with an inertial measurement unit (IMU) and cameras [69, 70]. Stereo vision was employed to improve angular velocity and position estimation over traditional approaches via sensor fusion in a UKF [71].

III. System Dynamics

Consider a flying object with mass $m$ as shown in Fig. 2. The object, with known geometry and physical properties, has a body frame $\mathcal{F}_b$ located at its center of mass $o$ and defined by its principal axes. In the frame, the inertia tensor $\mathbf{Q} = \int_{B} \rho_b(z) \left( (z - o) \otimes (z - o)^T \right) \, dV$ is a diagonal matrix. Here, $I_3$ is the $3 \times 3$ identity matrix and the function $\rho_b(z)$ gives the object’s density about a point $z$. The rotation of $\mathcal{F}_b$ from some world frame $\mathcal{F}_w$ is described by a unit quaternion $r$. Let $b \mathbf{v}$ and $b \mathbf{w}$ be the object’s velocity and angular velocity expressed in $\mathcal{F}_b$, respectively. The object’s velocity in $\mathcal{F}_w$ is given by a quaternion product: $^w \mathbf{v} = r \otimes (b \mathbf{v}) \otimes r^*$, where $r^*$ is the conjugate of $r$.

Denote by $^w \mathbf{f}$ the non-gravitational force exerted on the object in the world frame $\mathcal{F}_w$. Newton’s equation takes on the form

$$^w \mathbf{F} = \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} + \frac{1}{m} ^w \mathbf{f},$$

where $g \approx 9.80665 \text{ m/s}^2$ is the gravitational acceleration constant. Euler’s equation assumes the form

$$Q^b \mathbf{w} + b \mathbf{w} \times (Q^b \mathbf{w}) = b \tau,$$

where $b \tau$ is the external torque in the body frame. We have

$$b \mathbf{w} = Q^{-1} (b \tau - b \mathbf{w} \times (Q^b \mathbf{w})).$$

The object is subject to aerodynamic forces whose effects are non-negligible if its mass density is low. Without accounting for these forces, it can be difficult to accurately track the object’s state, or predict a future state for the purpose of planning. To describe the aerodynamic forces, consider a snapshot at a time instant during flight. The object, viewed “stationary”, is experiencing a flow of air with velocity $-b \mathbf{v}$ expressed in its body frame $\mathcal{F}_b$. There are two primary aerodynamic forces: lift, denoted $b \mathbf{f}_l$, and drag, denoted $b \mathbf{f}_d$.

Lift force acts on the object due to air particles traveling smoothly over its surface $\partial B$ and producing pressure forces normal to the surface. Bernoulli’s equation [72, pp. 20–21] gives the following relationship between air pressure $p$ and air speed $u$:

$$p = p_0 - \frac{1}{2} \rho u^2,$$

where $p_0$ is the constant total pressure and $\rho$ is air density. At a surface area element $e$, denote by $n(e)$ the unit outward normal and by $b \mathbf{u}(e)$ the air velocity (whose exact form is yet to be determined). The total lift in the frame $\mathcal{F}_b$ is then calculated by integrating the pressure $p$ over $\partial B$ as follows:

$$b \mathbf{f}_l = \int_{\partial B} - p n(e) \, de = -p_0 \int_{\partial B} n(e) \, de + \frac{1}{2} \rho \int_{\partial B} (b \mathbf{u}(e) \cdot b \mathbf{u}(e)) \, n(e) \, de = \frac{1}{2} \rho \int_{\partial B} (b \mathbf{u}(e) \cdot b \mathbf{u}(e)) \, n(e) \, de.$$

In the second step above, we used the fact $\int_{\partial B} n(e) \, de = 0$, which easily follows from the divergence theorem. Meanwhile, the air pressure yields a torque about the center of mass $o$:

$$b \tau_l = \frac{1}{2} \rho \int_{\partial B} (b \mathbf{u}(e) \cdot b \mathbf{u}(e)) \, e \times n(e) \, de.$$

Consider a neighborhood $N$ of the object in which the air flow is affected by its presence at the same time instant. Under the assumption that the air flow is incompressible and irrotational [73, p. 100], the air velocity at a point $e \in N$ is $\nabla \phi(e)$, where $\phi$, known as the velocity potential, satisfies the Laplace equation below:

$$\nabla^2 \phi(e) = 0.$$

Here, $\nabla^2$ is the Laplacian operator. Denote by $\partial N$ the outer boundary surface of the neighborhood $N$. The Laplace equation (6) is subjected to two boundary conditions:

$$\nabla \phi(e) = -b \mathbf{v}, \text{ for all } e \in \partial N,$$

$$\nabla \phi(e) \cdot n(e) = 0, \text{ for all } e \in \partial B.$$

Calculation of $\phi$ involves numerically solving (6) for all the points within the neighborhood $N$ containing the object. This is detailed in the supplemental material [74].
Meanwhile, within a thin boundary layer next to the object’s surface, the viscosity of air exists to produce friction with the surface. This leads to two effects inside the thin region [72, p. 42]: 1) air rotating with the surface and 2) drag force. In the former, air molecules with zero velocity relative to the body contribute a rotational component to air velocity. This creates the Magnus effect [72, pp. 16–33]. Combining the component with the air velocity due to the potential $\phi$ in the larger neighborhood $N$, we have

$$b\mathbf{u}(e) \approx \nabla\phi(e) + b\mathbf{\omega} \times b\mathbf{e},$$  \hspace{1cm} (9)

for an element $e$ within the thin layer with body coordinates $b\mathbf{e}$.

Moreover, the boundary layer contributes two types of drag force: pressure drag from the difference in pressure between the front and back side of the object, and skin friction from tangential stress on the surface [26, pp. 201–202]. Due to the mathematical difficulty in modeling these effects, the following approximation is adopted [73, pp. 211–215, 339]:

$$b\mathbf{f}_d = -\frac{1}{2}\rho C_d A (b\mathbf{v} \cdot b\mathbf{v}) b\mathbf{v},$$  \hspace{1cm} (10)

where $C_d$ is the drag coefficient and $A$ is the cross-sectional area. The approximated drag force acts through the object’s center of mass, yielding no torque. The coefficient $C_d$, dependent on the object’s shape, is determined from studies on the aerodynamics of general shapes [75], [76]. To compute the area $A$, all the points on the object’s surface are projected orthographically onto a plane with normal $b\hat{\mathbf{e}}$. The convex hull of their image points in the plane is constructed with its area assigned to $A$.

Finally, we apply the calculated forces and torques in Newton’s and Euler’s equations (1)–(2), where

$$w\mathbf{f} = \mathbf{r} \times (b\mathbf{f}_l + b\mathbf{f}_d) \otimes \mathbf{r}^*,$$  \hspace{1cm} (11)

$$b\mathbf{\tau} = b\mathbf{\tau}_l.$$  \hspace{1cm} (12)

Fig. 3 gives a diagram of the steps to calculate aerodynamic forces. The Laplace equation is solved given the object’s current velocity estimate $b\mathbf{v}$. The resulting potential field $\phi$ is used along with $b\mathbf{\omega}$ to calculate the air velocity $b\mathbf{u}(e)$ via equation (9). Surface integration (4) and (5), based on Bernoulli’s equation (3), are carried out to yield the lift force $b\mathbf{f}_l$ and torque $b\mathbf{\tau}_l$. The drag force $b\mathbf{f}_d$ is simultaneously computed from the velocity estimate according to (10). Newton’s and Euler’s equations of dynamics are then integrated to yield new velocity estimates for the next round of calculations.

The state of the flying object is described by the 13-vector

$$\xi = \left( o^T, \mathbf{r}^T, w^T \mathbf{v}^T, b\mathbf{\omega}^T \right)^T.$$

Its quaternion component $\mathbf{r}$, subject to the unit length constraint $|\mathbf{r}| = 1$, has the following derivative given in Appendix C of [77]:

$$\dot{\mathbf{r}} = \frac{1}{2} \mathbf{r} \otimes b\mathbf{\omega}.$$  \hspace{1cm} (14)

This, together with $\dot{\mathbf{o}} = w\mathbf{v}$, (1), and (2), forms the following system of nonlinear differential equations:

$$\dot{\xi} = a(\xi) = \begin{pmatrix} w\mathbf{v} \\ \frac{1}{2} \mathbf{r} \otimes b\mathbf{\omega} \\ g + \frac{w\mathbf{f}}{m} \\ Q^{-1} (b\mathbf{\tau} - b\mathbf{\omega} \times (Q b\mathbf{\omega})) \end{pmatrix}.$$  \hspace{1cm} (15)

### IV. Observables from Vision

The object’s state $\xi$ is estimated based on measurements extracted out of its images taken simultaneously by two cameras. One of the cameras observing the flying object is shown on the left in Fig. 4. At the camera’s focal point $w\mathbf{c}$ is a frame $F_c$ whose z-axis is perpendicular to the image plane $\Pi$. The rotation of $F_c$ from the world frame $F_w$ is described by a quaternion $r_c$. The plane $\Pi$ has a local $w\mathbf{v}$ frame at the upper left corner of the image. Let $p$ be a point on the object. Its coordinates $\mathbf{p}$, and $w\mathbf{p}$ in the frames $b\mathbf{F}_c$, $w\mathbf{F}_c$, and $w\mathbf{F}_w$ respectively, are described by the following mappings:

$$w\mathbf{p} = \mathbf{o} + \mathbf{r} \otimes (w\mathbf{p} - w\mathbf{c}) \otimes \mathbf{r}^*,$$  \hspace{1cm} (16)

$$w\mathbf{p} = \mathbf{r}_c \otimes (w\mathbf{p} - w\mathbf{c}) \otimes \mathbf{r}_c.$$  \hspace{1cm} (17)

As described by Forsyth and Ponce [78, pp. 16–18], the pinhole camera model maps $w\mathbf{p}$ to the normalized image coordinates $\mathbf{p}$, which is distorted under Brown’s model [79] to become $\mathbf{\tilde{p}}$ before transformed into image coordinates $\mathbf{p}$ [78, p. 17]. The above sequence of transformations is best summarized as follows:

$$w\mathbf{p} \rightarrow w\mathbf{p} \rightarrow \mathbf{p} \rightarrow \mathbf{\tilde{p}} \rightarrow \mathbf{p}.$$  \hspace{1cm} (18)

Since $w\mathbf{p}$ is known beforehand (as the position of a feature on the object), $w\mathbf{p}$, $\mathbf{p}$, and $\mathbf{\tilde{p}}$ are functions of the object’s position $\mathbf{o}$ and orientation $\mathbf{r}$, as well as the camera’s position $\mathbf{c}$ and orientation $\mathbf{r}_c$ (as a unit quaternion).

Now we consider the second camera as shown on the right of Fig. 4. It has focal point $c'$, and its body frame $F_{c'}$ has a relative orientation described by the quaternion $r_{c'}$ to the world frame $F_w$. This camera generates the image coordinates $\mathbf{\tilde{p}}$ of the same point $\mathbf{p}$. Knowing $w\mathbf{p}$ and $\mathbf{\tilde{p}}$, we can respectively recover the normalized image coordinates $\mathbf{p}$ and $\mathbf{\tilde{p}}$ using the Levenberg-Marquardt algorithm for nonlinear least-squares minimization [80], [81].

As illustrated in Fig. 4, two rays, originating at $\mathbf{p}$ on the object and passing through the focal points $\mathbf{c}$ and $\mathbf{c}'$, intersect...
the two image planes at the points \( q \) and \( q' \), respectively. The epipolar constraint, stating the co-planar-ity of these five points, has the compact form \((rp^T, 1)E(rp^T, 1)^T = 0\) [78, pp. 200], where \( E \) is a 3 × 3 skew-symmetric matrix with rank two [82], [83].

During the object’s flight, the epipolar constraint is not exactly satisfied due to noise. For every feature point \( p_j \), \( j = 1, \ldots, n \), we denote \( n_j \) as the normal to the epipolar plane defined by \( p_j \) and the two focal points \( c \) and \( c' \).\(^2\) Let \( V \) and \( V' \) be the two sets of point features currently visible in the respective images produced by the two cameras. A vector of observables is defined as follows:

\[
y = \begin{pmatrix} \ldots, p_j^T, \ldots, p_k^T, \ldots, n_l^T, \ldots \end{pmatrix}^T,
\]

for all \( b p_j \in V, b p_k \in V' \), and \( b p_l \in V \cap V' \).

The cameras positions \( c, c' \) and orientations \( r_c, r_{c'} \), known as the extrinsic camera parameters, are approximated off-line via a calibration procedure which enforces the epipolar constraint. Their measurement errors may compromise the epipolar constraint and consequently the estimation accuracy during the object’s flight. Motivated by SLAM and VINS algorithms [45], [46], [66], we conduct online estimation of parameters to find an agreement between a pose estimate of the object and those of the cameras.

The state vector \( \xi \) in (13) is now augmented to include the pose of each camera:

\[
x = (\mathbf{o}^T, r^T, w_v^T, b\omega^T, c^T, r_c^T, c'^T, r_{c'}^T)^T,
\]

(20)

The state \( \dot{x} \) has 27 variables in total. From equation (15), the system dynamics are subsequently augmented:

\[
\dot{x} = b(x),
\]

(21)

where the derivatives of \( c, r_c, c' \) and \( r_{c'} \) are all zero. We also write the vector of observables as \( y = h(x) \) despite that it does not depend on state variables \( v \) and \( \omega \).

V. KALMAN FILTERING WITH QUADRATIC CONSTRAINTS

The dynamics model (21) propagates the state of the object with time, while the measurement model described in Section IV can be employed to correct the state based on visual inputs acquired at discrete time instants. Due to the need to cope with nonlinearities present in both models, we here use a hybrid extended Kalman filter (EKF) [84, pp. 403–407].

Denote by \( x_k \) the value of the state \( x \), and \( y_k \) the value of the vector \( y \) of observables, all at the \( k \)-th time instant. The size \( m \) of \( y_k \) varies with the number of observable features. The measurement model can be written as

\[
y_k = h_k(x_k) + \nu_k,
\]

(22)

where the \( m \)-vector \( \nu_k \) is the zero-mean Gaussian white noise whose covariance is given by an \( m \times m \) matrix \( R_k \).

The EKF estimates the distribution of the state \( x \). It first forward integrates (21) with \( b(x) \) evaluated at the posterior estimate \( \hat{x}_{k-1} \) from time instant \( k-1 \). The result is a prior estimate \( \hat{x}_k \) at time instant \( k \). Meanwhile, integrate the differential Riccati equation [84, p. 405] to update the \( 27 \times 27 \) covariance matrix from its posterior value \( P_{k-1}^+ \) at instant \( k-1 \) to its prior value \( P_k^- \) at instant \( k \). The estimate \( \hat{x}_k \) is then corrected to yield a posterior estimate \( \hat{x}_k^+ = \hat{x}_k + K_k\epsilon_k \), where \( K_k \) is a gain matrix determined through optimization and \( \epsilon_k = y_k - h_k(\hat{x}_k) \) is the residual. The posterior covariance matrix \( P_k^+ \) is updated using \( K_k \) to conclude the current step of estimation. We refer the reader to [84, pp. 124-129] for more details of the above updates.

A. Constrained Kalman Filtering

When the system state is subjected to some constraints, the unconstrained EKF described above performs corrections that would cause the estimate to drift away from the constraint surface. To resolve this, the constraints need to be incorporated into the optimization problem for determining the Kalman gain \( K_k \). The state \( x \) in equation (20) contains three unit quaternions \( r, r_c, r_{c'} \) representing rotations of the object and two cameras, respectively. These quaternions are each seen as a unit 4-vector so they induce constraints

\[
\eta_i(x) \equiv x^TA_i x - 1 = 0,
\]

(23)

for \( i = 1, 2, 3 \), where

\[
A_1 = \text{diag}(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
A_2 = \text{diag}(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
A_3 = \text{diag}(0, 0, 0, 0, 0, 0, 0, 0, 0, 0).
\]

In the above, \( \text{diag}(\cdot) \) specifies a diagonal matrix whose main diagonal consists of the listed arguments with \( 0_d \) and \( 1_d \) being \( d \)-vectors of zeros and ones, respectively.

The aim is to modify the Kalman filter equations to include the three quadratic constraints given in (23). The gain matrix \( K_k \) is now obtained through constrained minimization:

\[
\min_{\hat{x}_k} \text{tr}(P_k^+ x_k a_k) \quad \text{subject to} \quad \eta_i(\hat{x}_k) = 0, \quad i = 1, 2, 3.
\]

(24)
Below we will derive a solution to (24) that can be easily extended to handle multiple quadratic constraints in the general form.

B. Solution to the Constrained Minimization Problem

Continue to let the time be fixed at instant \( k \). With no ambiguity, we shall temporarily omit the subscripts in \( x_k^+, x_k^-, K_k, e_k, P_k^+, P_k^- \), and \( R_k \). Letting \( x = \hat{x}^+ = \hat{x}^+ + K \epsilon \), the constraints in (23) are expanded into, for \( i = 1, 2, 3 \),

\[
\eta_i(\hat{x}^+) = (x^+ + K\epsilon)^\top A_i(x^+ + K\epsilon) - 1 = 0. \tag{25}
\]

We write \( \eta = (\eta_1, \eta_2, \eta_3)^\top \). Substituting in (25) and the equation \( P^+ = (I_{27} - KH)P^- (I_{27} - KH)^\top + KHK^\top \) [84, p. 85], the Lagrangian for the constrained minimization problem (24) is

\[
L = \text{tr} \left( (I_{27} - KH)P^- (I_{27} - KH)^\top + KHK^\top \right) + \lambda^\top \eta, \tag{26}
\]

where the vector \( \lambda = (\lambda_1, \lambda_2, \lambda_3)^\top \) contains three Lagrange multipliers, one for each constraint in (25). The optimal values of \( K \) and \( \lambda \) are attained at a stationary point of \( L \).

The partial derivative of \( L \) with respect to \( \lambda_i \) simply recovers the constraint (23), for \( i = 1, 2, 3 \). To derive \( \partial L / \partial K \), we first obtain

\[
\frac{\partial}{\partial K} \text{tr} \left( KHK^\top \right) = 2KR, \tag{27}
\]

by making use of the derivative \( \partial \text{tr}(AX^\top) / \partial X = A \) for matrices \( A \) and \( X \), as well as the symmetry of \( R \). Next we have, by making use of the symmetry of \( P^- \),

\[
\begin{align*}
\frac{\partial}{\partial K} \text{tr} \left( (I_{27} - KH)P^- (I_{27} - KH)^\top \right) &= \frac{\partial}{\partial K} \text{tr} \left( KHP^- (KH)^\top \right) - 2 \frac{\partial}{\partial K} \text{tr} \left( P^- H^\top K^\top \right) \\
&= 2KHP^-H^\top - 2P^- H^\top. \tag{28}
\end{align*}
\]

Meanwhile, differentiation of (25) yields

\[
\frac{\partial}{\partial K} (\eta_i(\hat{x}^+)) = 2A_i \hat{x}^+ \epsilon^\top + 2A_i K \epsilon \epsilon^\top. \tag{29}
\]

In deriving (29), we made use of \( \partial(x^\top Ay) / \partial A = xy^\top \) for vectors \( x, y \), and matrix \( A \), as well as the symmetry of \( A_i \). With (27)–(29) we can write out the partial derivative \( \partial L / \partial K \):

\[
\frac{\partial L}{\partial K} = 2 \left( KS - N K \epsilon \epsilon^\top - M \right). \tag{30}
\]

where

\[
S = HP^-H^\top + R,
\]

\[
N(\lambda) = \sum_{i=1}^{3} \lambda_i A_i,
\]

\[
M(\lambda) = P^- H^\top - \sum_{i=1}^{3} \lambda_i A_i \hat{x}^+ \epsilon^\top.
\]

Vanishing of \( \partial L / \partial K \) is expressed by the equation

\[
KS + N K \epsilon \epsilon^\top = M. \tag{31}
\]

The remaining task is to solve (25) and (31) for \( K \) and \( \lambda \). First, obtain \( K \epsilon \) via right multiplication of (31) by \( S^{-1} \epsilon \):

\[
K \epsilon = J^{-1} m, \tag{32}
\]

where, denoting \( \epsilon = \epsilon^\top S^{-1} \epsilon \),

\[
J(\lambda) = I_{27} + \epsilon N(\lambda) \quad \text{and} \quad m(\lambda) = M(\lambda) S^{-1} \epsilon.
\]

Substitute (32) into (25) to eliminate \( K \) and obtain

\[
\alpha_i \equiv \eta_i(\hat{x}^+) + 2(\hat{x}^+)^\top A_i J^{-1} m + (J^{-1} m)^\top A_i J^{-1} m = 0,
\]

for \( i = 1, 2, 3 \). Rewrite the three equations (33) as

\[
\alpha(\lambda) = 0, \tag{34}
\]

where \( \alpha = (\alpha_1, \alpha_2, \alpha_3)^\top \). The system (34) has 3 equations in 3 unknowns, and can be solved via root finding with a combination of the (local) Newton’s method and the (global) homotopy continuation method [85]. These solvers make use of the Jacobian \( \partial \alpha / \partial \lambda \), which requires the following partial derivatives, \( i, j = 1, 2, 3 \):

\[
\frac{\partial m}{\partial \lambda_j} = -\epsilon A_i \hat{x}^+ \quad \text{and} \quad \frac{\partial J^{-1}}{\partial \lambda_j} = -\epsilon J^{-1} A_i J^{-1}. \tag{35}
\]

In the above, \( \partial J^{-1} / \partial \lambda_j \) was derived from differentiating \( J^{-1} J = I_{27} \) while making use of the symmetry of \( J \).

When \( k = 1 \), the Lagrange multipliers \( \lambda \) are first initialized by solving (34) via homotopy continuation. When \( k > 1 \), the multipliers’ values are expected to change slightly from those at time instant \( k - 1 \). Hence, these old values can be used as a starting point by Newton’s method to achieve quick updates. In the case of divergence, we will fall back on homotopy continuation. The gain matrix \( K \) is next solved from equation (31) after substituting equation (32) on the left hand side. The matrix is used in the correction step of the EKF to produce state estimates that satisfy the set of quadratic constraints.

The optimization procedure is easily adapted to work with quadratic constraints in the general form of \( x^\top A_i x + b_i^\top x + c_i = 0, 1 \leq i \leq m \), where \( A_i \) is not necessarily symmetric. We need only make several changes described below. In equations starting at (29), every appearance of \( A_i \) will be replaced with \( (A_i + A_i^\top) / 2 \). The partial derivative in (29) will include an extra summand \( b_i \epsilon^\top \), and this change will propagate to (30) and (31) and, subsequently, to the expression for \( M(\lambda) \).

VI. Graph-Based Feature Tracking

Motion estimation has so far assumed a fixed set of known correspondences between point features in images and those on the moving object. So the measurement model (22) allows the working of a single Kalman filter. The property cannot be maintained, however, in the course of the object’s flight as features emerge and disappear, which introduces discontinuities in observations. We now address this issue of identifying and tracking features of the object in images with respect to those on its shape model. To avoid an unnecessary diversion into handling of geometric complexities, we assume the object to be a polyhedron \( P \) whose vertex connectivity can be represented by a planar graph. The range of \( P \) includes
A. Image Graph and Hypothesis

The polyhedron \( P \) with \( n \) vertices \( p_0, \ldots, p_{n-1} \). Its underlying planar graph is referred to as the model graph \( M \), which is represented by a doubly-connected edge list (DCEL) \([86, pp. 29–33]\). The process turns every facet of \( P \) into a bounded region in \( M \), except for one facet, which is “opened up” to become the unique unbounded region. As shown in Fig. 5(a) and (c) for an octahedron, every vertex \( p_i \) of \( P \) corresponds with the vertex \( v_i \) in \( M \).

A frame generated by either camera is a light intensity image of \( P \) with its faces illuminated differently and showing discontinuities at edges and vertices. These edge and vertex features are used to construct a planar graph \( G \), referred to as the image graph, as illustrated in Fig. 5(b). The features are detected from the image using a Canny edge detector [87] and Shi-Tomasi corner detector [88]. We denote \( \langle u_i, v_j \rangle \) if the vertex \( u_i \) in \( G \) corresponds to the vertex \( v_j \) in the model graph \( M \). The tracking algorithm hypothesizes a set of vertex correspondences \( C(G, M) = \{ \langle u_i, v_j \rangle \} \). This set represents a bijective mapping from a vertex subset of \( G \) to one of \( M \). The mean square error of \( C(G, M) \) given a state estimate \( \hat{x} \) is defined as follows:

\[
d(C, \hat{x}) = \frac{1}{|C|} \sum_{(u_i, v_j) \in C} \| u_i - v_j \|^2,
\]

where \( u_i \) represents the image coordinates of \( u_i \) in \( G \), and \( v_j \) represents those of \( v_j \) when the polyhedron is projected on the image plane under the transformation defined in \( \hat{x} \).

B. Overview of Work at One Time Instant

The two cameras are synchronized to produce images (or “frames”) of the polyhedron at equally spaced time instants. Let us focus on the \( k \)-th instant when two image frames \( I_k \) and \( I_k' \) are generated simultaneously. The image graphs \( G_k \) and \( G_k' \) constructed from the frames are associated with two sets of vertex correspondences \( C(G_k, M) \) and \( C(G_k', M) \), respectively, which form a hypothesis \( H \). The two correspondence sets are first updated as detailed in Section VI-C. Any change to the two sets is propagated to the vector \( y_k \) of observables and its measurement model (22), using the image points of vertices. Next, the Kalman filter \( K \) associated with the hypothesis \( H \) corrects its prior state estimate \( \hat{x}_k \) to yield the posterior estimate \( \hat{x}_k^+ \).

The updated estimate \( \hat{x}_k^+ \) is now applied to the object. With the estimate we evaluate the error of the hypothesis:

\[
e(H) = d(C(G_k, M), \hat{x}_k^+) + d(C(G_k', M), \hat{x}_k^+).
\]

to cope with noise and provide some tolerance, the hypothesis \( H \) is rejected if \( e(H) \geq \Gamma \) in each of the \( l \) most recent time instants. Proper tuning of \( \Gamma \) and \( l \) helps the algorithm settle on a small set of low-error hypotheses.

At the \( k \)-th time instant, a collection \( H \) of hypotheses (including \( H \)) are active, each equipped with a Kalman filter responsible for estimation under the hypothesis. If no features are detected in either image, then the active Kalman filters update their estimates without correction for observables.

In the case that all active hypotheses have been eliminated, new ones will be generated before moving on to the next time instant. This is described in Section VI-D.

C. Updating a Hypothesis

Let us go back to \( H \) as one of the hypotheses active at time instant \( k \). Its correspondence sets \( C(G_k, M) \) and \( C(G_k', M) \) were passed on from time instant \( k - 1 \), or, newly created with \( H \) at the end of processing for that time instant. They need to be updated because, in the images \( I_k \) and \( I_k' \), some vertices, and some edges of the polyhedron may have appeared or disappeared or changed. Below we focus on how \( C(G_k, M) \) is updated from \( C(G_{k-1}, M) \), as \( C(G_k', M) \) can be similarly updated.

The first stage of the update begins with computing the maximal correspondence between \( G_k \) and \( G_{k-1} \). Their initial vertex correspondences can be obtained during image processing. Using the computed \( C(G_{k-1}, G_k) \) and the inherited \( C(G_{k-1}, M) \), we initialize \( C(G_k, M) \) such that \( \langle u_k, v_k \rangle \in C(G_k, M) \) whenever \( \langle u_k, v_k \rangle \in C(G_{k-1}, G_k) \) and \( \langle u_k, v_k \rangle \in C(G_{k-1}, M) \) for some \( u_k \). In the second stage, \( C(G_k, M) \) is extended to become a maximal correspondence set.

Both stages involve computing the correspondence between two planar graphs, first \( G_{k-1} \) and \( G_k \), and then \( G_k \) and \( M \). This correspondence problem is one of subgraph isomorphism, which can be done in linear time, but becomes NP-complete in the presence of noise in images where some tolerance to erroneous edges and vertices must be allowed [89]. Approaches to error-tolerant graph matching include tree search algorithms in exponential time [35], [90], [91], and sub-optimal algorithms in polynomial time [37], [92].

We here propose a tree search subroutine \( A(G_k, M, C_0) \), where \( C_0 \) is the initial value of \( C(G_k, M) \). We can then call \( A(G_{k-1}, G_k, D_0) \) to extend the correspondence \( C(G_{k-1}, G_k) \) from an initial value \( D_0 \). To also address generation of a new hypothesis, we here let the algorithm start with \( C(G_k, M) = 0 \). A depth-first search is performed by matching edges of \( G_k \) with different edges of \( M \). Fig. 6 demonstrates a portion of the search performed on an example pair of \( G_k \) and \( M \).
Hypothesis update
Dynamics
Kalman filtering

D. Generating New Hypotheses

If \( k = 1 \) or all hypotheses have been rejected based on the posterior estimate \( \hat{x}_k^- \), the algorithm attempts to generate a new set of hypotheses. Start with a correspondence set \( C_0 = \{\langle u_1, v_1 \rangle, \langle u_2, v_2 \rangle\} \), which involves adjacent vertices \( u_1 \) and \( u_2 \) in \( G_k \) and \( v_1 \) and \( v_2 \) in \( G_k' \). A maximum correspondence set \( C(G_k, G_k', C_0) \) is constructed using the tree search subroutine \( A(G_k, G_k', C_0) \). Next, by matching the edge \( \langle u_1, u_2 \rangle \) with an edge of \( M \), the same algorithm finds all maximal correspondence sets \( C_1, \ldots, C_N \) between \( G \) and \( M \). For each \( C_i, i = 1, \ldots, N \), we construct a correspondence set \( C_i' \) between \( G_k' \) and \( M \) such that \( \langle u', v' \rangle \in C_i' \) whenever \( \langle u, v \rangle \in C_i \) for some vertex \( u \). A hypothesis \( H_{\text{new}} = (C_i, C_i') \) is thus obtained. To evaluate its error \( \epsilon(H_{\text{new}}) \), we need to initialize a new state estimate \( \hat{x}_{k+1}^+ \). Under the hypothesis \( H_{\text{new}} \), initialization of the object’s position \( \mathbf{o} \) and orientation \( \mathbf{r} \) as components of \( \hat{x}_{k+1}^+ \) is described in Appendix A. Its initial velocity can be estimated via finite differencing over position during the first few frames. Estimate of the initial angular velocity is set to zero to avoid a large error from differencing. If \( \epsilon(H_{\text{new}}) < \Gamma \), the hypothesis is added along with its Kalman filter. The above procedure repeats by matching \( \langle u_1, u_2 \rangle \) with other edges of \( M \) at its start, returning any hypotheses with low errors.

VII. Experiments

To demonstrate the accuracy and robustness of the presented estimation scheme, experiments were conducted with three objects: a wooden frame, rugby ball, and foam polyhedron. The experiment with the wooden frame was conceived to have a source of ground truth available and, for this purpose, by “invasive” means of attaching accelerometers to the frame to alter its structure. The rugby ball experiment addressed motion estimation in sports where objects tend to have smooth surfaces and experience large Magnus forces. The polyhedron experiment aimed at a generally shaped object with the objective of demonstrating estimation using non-fiducial features.

As summarized in Table I, various compositions of the three basic modules (aerodynamics, Kalman filtering, and graph-based feature tracking), with increasing complexity, were validated. Five different estimators were used: an extended Kalman filter (EKF), unscented Kalman filter (UKF) [93], quadratically constrained Kalman filter (QCKF) described in Section V-B, a QCKF with aerodynamics approximated for a sphere (QCKF-S), and a QCKF with aerodynamics fully modeled (QCKF-A). All five estimators used the dynamics model (15), while the first three had both the lift force \( f_L \) and drag force \( f_D \) set to zero. All five used the projection model of Section IV, and except for the first estimator, incorporated the quaternion constraints (25). For the QCKF-A, lift force was calculated per the procedure in [74]. The QCKF-S instead approximated the object by a volume-equivalent sphere, and applied empirical equations of drag and lift for a spherical shape. Both the rugby ball and the polyhedron experiments

3In the case of the UKF, the quaternion constraints were enforced using a projection method [63].

Fig. 6. Search tree explored by the maximal correspondence algorithm on the image graph \( G_k \) and model graph \( M \) from Fig. 5. Each non-root node represents a correspondence of two vertices from \( G_k \) and \( M \), or \( \langle u, \emptyset \rangle \) in the case that no correspondence is made, effectively ignoring vertex \( u \). The pair of bolded edges in \( G_k \) and \( M \) identify the initial edge correspondence. Each leaf is labeled with a set \( C \), produced by collecting all vertex correspondences along the path descending from the root to the leaf. The algorithm explores a total of 20 sets of correspondences, of which 6 are shown in the expanded subtree.

Fig. 7. Overview of the steps to update a hypothesis, estimate its corresponding state, and compute its error.

Matching the ordered edges \( \langle u_0, u_1 \rangle \in G_k \) and \( \langle v_4, v_1 \rangle \in M \). To find the correspondences for the vertex \( u_2 \) adjacent to \( u_1 \), it scans the edges incident on \( v_1 \) clockwise, yielding the branches at the third level of the tree that represent the correspondences \( \langle u_2, v_3 \rangle, \langle u_2, v_0 \rangle, \) and \( \langle u_2, v_4 \rangle \). A fourth branch representing \( \langle u_2, \emptyset \rangle \) is added to include the possibility of \( u_2 \) not matched. This traversal continues until all vertices have been visited. As a result, every path from the root down to a leaf represents a possible correspondence set. For example, in Fig. 6, the leaf node \( \langle u_3, v_5 \rangle \) denotes the correspondence set \( C_1 = \{\langle u_0, v_4 \rangle, \langle u_1, v_1 \rangle, \langle u_2, v_3 \rangle, \langle u_4, v_0 \rangle, \langle u_3, v_5 \rangle\} \).

Multiple correspondence sets are often generated by the tree search algorithm. A metric is thus needed for comparing them. As this is before the Kalman filter makes correction to the prior estimate \( \hat{x}_k^- \), a correspondence set \( C_1 \) is larger than another correspondence set \( C_2 \), denoted \( C_1 \succ C_2 \), if

\[
\begin{align*}
&(|C_1| = |C_2| \land d(C_1, \hat{x}_k^-) < d(C_2, \hat{x}_k^-)) \lor \\
&(|C_1| > |C_2| \land d(C_1, \hat{x}_k^-) - d(C_2, \hat{x}_k^-) < \delta),
\end{align*}
\]

for some small threshold \( \delta > 0 \). If neither \( C_1 \succ C_2 \) nor \( C_1 \prec C_2 \), then \( C_1 = C_2 \). A maximum correspondence set \( C^* \) is chosen, i.e., we have \( C(G_k, M) = C^* \).

Similarly, we update the correspondence set \( C(G_k', M) \) based on the image graph \( G_k' \) generated by the second camera. Update of the hypothesis \( \hat{H} \) is then completed for the time instant. The Kalman filter is applied as described in Section V to generate the posterior estimate \( \hat{x}_{k+1}^+ \). Figure 7 summarizes the steps carried out so far.
made use of the graph-based feature tracking algorithm from Section VI.

In each experiment, two Ximea MQ02CG-CM high-speed, color cameras in stereo vision configuration were used to capture images simultaneously of an object in flight at more than 200 fps. Both cameras were equipped with 81.9 degree field of view Navitar NMV-6 lenses. They were carefully calibrated with point data obtained by images of a 3D calibration object via bundle adjustment [94] and individually initialized camera parameters [95]. The two cameras’ parameters were estimated and poses approximated together via optimization under the epipolar constraint for minimum reprojection error.

Estimation ran until the object left both cameras’ fields of view. The initial state estimate was conducted as described in Section VI-D. The initial angular velocity had its estimate set close to zero to avoid a large error from differencing. Initial error covariances $P_0^i$ of the state estimate and $R_0$ of the vector of observables were determined from tuning the Kalman filter over more than 100 experiment trials.\(^5\)

For metric on pose estimation error, we choose the root-mean-square error (RMSE) formulated as $\sqrt{d(C, \hat{x})}$, where $d$ is defined in (36), $C$ is the winning hypothesis, and $\hat{x}$ is its state estimate.

### A. Wooden Frame with Accelerometers

In the first experiment, a wooden frame object made up of three orthogonal links was thrown by hand across the two cameras’ fields of view approximately 2.3 m away. Shown in Fig. 8(a), the object contained three links with four vertices. Each link was easily detected by its color during image processing. Mounted near each vertex was a tri-axis ADXL335 accelerometer from Analog Devices, with range $\pm29.4$ m/s\(^2\), to measure linear acceleration at the point.\(^6\) Fig. 8(b) shows the configuration of one axis of the frame. The data acquired by each accelerometer were transmitted wirelessly using a separate Digi XBee Series 1 modules.\(^7\)

With a total mass of 0.31 kg, the wooden frame is subjected to negligible effects of aerodynamic forces. The frame was estimated by the QCKF, EKF, and UKF for 0.62 s at a rate of 202 fps, and over a distance of 2.13 m within the images of its flight as shown in Fig. 9(a)–(b). The poses estimated by the EKF and QCKF at a few moments are also drawn. In the images, pairs of poses estimated by the EKF do not appear to align well with the object, primarily due to absence of the epipolar constraint and the unit quaternion constraints. Meanwhile, poses estimated by the QCKF align better as a result of including the epipolar normals as observables. Consequently, the EKF yielded higher reprojection errors. The UKF (not shown) obtained similar errors to the QCKF due to its enforcement of the constraints.

For validation, we compare the estimated pose and velocities with those measured by accelerometers. The pose and velocity $b\dot{v}$ are initialized as described earlier. The $i$-th accelerometer produces readings $a_i$ that satisfy the kinematics equation $a_i = b\dot{v} + b\omega \times b\tau_i + b\omega \times (b\omega \times b\tau_i)$, where $b\tau_i$ locates the accelerometer in the body frame. Using four accelerometers, a system of twelve linear equations can be formulated in twelve (dependent) unknowns [18]: three for $b\dot{v}$, and nine for a $3 \times 3$ matrix $W$ composed from $b\omega$ and $b\dot{w}$. At each time instant, the resulting $b\omega$ updates the object’s rotation, which is used to convert $b\dot{v}$ back to the world frame after its update from integration of $b\dot{v}$.

The method described above of integrating accelerations produce the smooth curves plotted by solid lines in lighter colors in Fig. 9(c)–(d). For comparisons, these two diagrams also plot the estimated values of velocity and angular velocity by the UKF and QCKF. Velocity estimates of the two filters mostly overlap, while small gaps can be seen among $\dot{w}_x$ and $\dot{w}_y$. At the ending time 0.62 s, the estimation errors of the QCKF and UKF were respectively $(-0.056, -0.153, -0.091)$ and $(-0.185, -0.284, -0.107)$ m/s for velocity, and

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>EKF</td>
</tr>
<tr>
<td>Wood Frame</td>
<td>EKF</td>
</tr>
<tr>
<td>Rugby Ball</td>
<td>QC   ✓</td>
</tr>
<tr>
<td>Foam Polyhedron</td>
<td>QC   ✓</td>
</tr>
</tbody>
</table>

---

\(^5\)Composed of checkerboard patterns pasted on the faces of a cube

\(^6\)Covariances were 0.1 for the three components of the object’s position, 0.01 for four components of rotation and three components of velocity, and 1 for three components of angular velocity, 0.005 for six components of each camera’s position, and 0.001 for eight components of rotation. Measurement covariances were 25 for the two coordinates of image points and 0.003 for the three coordinates of the epipolar normals in equation (19).

\(^7\)The object’s inertia tensor was also adjusted to account for the accelerometers as point masses, yielding the value $Q = \text{diag}(5.57, 5.57, 5.60) \times 10^{-9}$ kg\cdot m\(^2\).
quickly improve. Increased pose errors are also seen due to the occlusion in Fig. 9(e)–(f). Here, the ground truth poses were approximated using the initial pose and integration of the calculated velocities. To reduce drift due to measurement error of the accelerometers, least squares minimization of reprojection errors from point features in the images is performed to correct poses. Errors of the UKF are also plotted by dashed lines and are generally slightly larger.

B. Rugby Ball

In the second experiment, estimation took place with aerodynamics on a thrown rugby ball. The hollow ball was primarily composed of rubber with a mass of 0.42 kg, and took on the shape of a prolate spheroid with semi-axis lengths 0.145, 0.095, and 0.095 m. Since spherical objects lack features to contribute measurements during estimation, five markers were attached to the rugby ball’s surface: a green marker at one of the two intersections with its major axis, and four blue markers at all intersections with its two minor axes. The markers form a topology based on their color and proximity to each other, allowing use of the feature tracking algorithm in Section VI. The model graph \( M \) induced by these markers is drawn in Fig. 10(a) while the image graph \( G \) at a few moments of flight is drawn in (b). The ball was thrown with a spin about its major axis (i.e. a “spin pass”), such that one green marker remained visible and the blue markers rotated into and out of view. The markers were detected by their color and provided to two estimators: QCKF-A (with aerodynamics modeled) and QCKF (without). To measure the cross sectional area needed for approximating drag force, 400 surface points were computed as “vertices” to be projected along the velocity direction. The ball’s flight spanned 0.704 s, at which the object achieved estimated velocity \( \mathbf{v} = (-2.148, 1.991, -3.547)^\top \) m/s and angular velocity \( \mathbf{\omega} = (18.280, -0.1576, 1.772)^\top \) rad/s.

How does the modeling of aerodynamics improve prediction of the object’s trajectory? Fig. 10(a) shows the position trajectories starting with estimates by the QCKF-A and QCKF at time 0.12 s and forward propagated to time 0.58 s near the end of flight. To evaluate these two estimates fairly, both trajectories are produced with inclusion of aerodynamics, and then compared to the “true” trajectory constructed over estimates continually acquired over the remainder of the flight. Hence, errors from the comparisons indicate the accuracies of the starting estimates. The position trajectory (green) starting from the estimate of the QCKF-A is seen to yield a much closer prediction. The position trajectory (red) of the QCKF falls faster than that of the QCKF-A due to the absence of an upward lift force from the Magnus effect produced by the

\[
(-0.004, 0.032, -0.026)^\top \quad \text{and} \quad (-0.023, -0.008, -0.009)^\top \text{ rad/s for angular velocity. The difference in these velocity errors, while seemingly small, yield larger errors (i.e. drift) in pose estimates through integration. Moreover, in (d) higher errors are seen to accumulate in } b_\omega y \text{ and } b_\omega z \text{ up to time } 0.22 \text{ s. This is due to the } x\text{-axis not being detected, effectively reducing the visual information available for estimating rotation about the two adjacent orthogonal axes. Once the } x\text{-axis becomes visible again, estimation errors in } b_\omega y \text{ and } b_\omega z \text{ TABLE II

<table>
<thead>
<tr>
<th>Object</th>
<th>( f_d ) (via equation (10))</th>
<th>( f_y ) (via equation (4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rugby Ball</td>
<td>[0.034, 0.063]</td>
<td>[0.327, 0.486]</td>
</tr>
<tr>
<td>Polyhedron</td>
<td>[0.100, 0.361]</td>
<td>[0.142, 0.394]</td>
</tr>
</tbody>
</table>
ball’s back-spin. Fig. 10(b) shows the estimated trajectory of the ellipsoid by the QCKF-A with the estimated velocities and resulting lift forces drawn by arrows. Moreover, the row for the rugby ball in Table II gives ratios of the magnitude of drag and lift forces to that of gravity, where \( f_l \) in particular is a large contributing force.

When compared to estimates at the end of flight, prediction errors of orientation as \( zyx \) Euler angles were \((0.62, 0.42, 0.24)\) rad for the QCKF, and \((0.47, 0.35, 0.18)\) rad for the QCKF-A. While the QCKF-A yields lower prediction errors, in both cases the errors about the \( z \)-axis were greater due to the high rate of rotation. Inclusion of aerodynamics also yielded slightly lower reprojection errors for QCKF-A shown in Fig. 10(c)–(d). As more image frames were acquired, integration of dynamics propagated errors to the object’s pose, resulting in an improvement of 2.2 px for the QCKF-A by the end of flight, or approximately 1 cm with the object at a distance of 2.9 m from the cameras.

### C. Irregular Polyhedron

In the final experiment, a hollow, polyhedral foam object was thrown by hand to demonstrate motion estimation alongside feature tracking in the absence of fiducial markers. The polyhedron was constructed by combining eight triangular facets cut from foam boards.\(^8\) It had a mass of 0.069 kg and inertia tensor of \( Q = \text{diag}(3.20, 1.96, 2.45) \times 10^{-4} \text{ kg \cdot m}^2 \).

Due to its large size and low weight, the flying polyhedron was subjected to the effects of aerodynamics, to the extent that air flew over its faces produced noticeable lift force. Overall, the aerodynamic effect was more complex than on the rugby ball from the previous experiment given the polyhedron’s geometric asymmetry and singularities at its edges and vertices.

Detected corner and edge features on the polyhedron were inputs to the feature tracking algorithm in Section VI. The polyhedron was estimated by QCKF-A for 0.52 s at a rate of 207 fps, and over a distance of 1.73 m within the image. Fig. 11(a)–(b) shows its images during the flight. The object’s pose estimates at a few time instants (e.g., 0.1 s) were plotted with only those edges visible to each camera. In several frames, some visible edges are clearly missing due to poor lighting of the polyhedron’s faces. Regardless, the feature tracking and estimation algorithms were able to keep up with the flight. Magnitudes of the polyhedron’s aerodynamic forces from QCKF-A are shown in Table II.

To demonstrate the performance, Fig. 11(c) plots a timeline of hypothesis errors for the object’s flight. The algorithm initially generates two hypotheses labeled 1 and 2. Both maintain relatively low error as their estimates improve, until time 0.25 s when errors spike due to noise in the image. A few frames later, the hypotheses are able to recover with updated correspondences, yielding errors that reflect the object’s new pose. In particular, hypothesis 2, which is now presumably an incorrect hypothesis due to its higher error, persists for a short while just below the threshold for rejection. At approximately 0.35 s, errors again increase due to noise that occurs as a face of the object disappears. At this moment, seen in Fig. 11(a) and (b), image processing fails in subsequent iterations to distinguish between the vertices labeled \( v_1 \) and \( v_2 \). A few iterations of the tracking algorithm have passed before the two hypotheses are rejected and hypothesis generation is triggered, yielding five new hypotheses labeled 3–7 at 0.4 s. Of these, hypothesis 3 is tracked with low error as it appears to be correct.

\(^8\) The polyhedron has the following vertices in the frame \( F_b \), aligned with its principal axes: \((0.051, -0.008, 0.121)\)\(^T\), \((-0.04, -0.055, 0.101)\)\(^T\), \((0.073, 0.103, -0.068)\)\(^T\), \((-0.071, 0.156, -0.002)\)\(^T\), \((-0.044, -0.095, -0.102)\)\(^T\), and \((0.031, -0.101, -0.05)\)\(^T\).
correctly identify the objects’ pose, while 6 and 7 are later rejected, and 4 and 5 remain within the error threshold. The object’s state estimate at each frame is chosen from the hypothesis with the lowest error.

It is inevitable that image noise and imperfect processing will prevent a single hypothesis from spanning the whole flight. Regardless, the sequence of poses of the object is successfully tracked by the hypotheses 1 and 3 (both conveniently colored blue). Due to the smooth transitioning of the state estimate for the new set of hypotheses, minimal progress in estimation is lost during the time when hypothesis generation takes place.

D. Comparison with Closed-form Lift Force using Spherical Approximation

The polyhedron was simultaneously estimated using closed-form equations of drag and lift for a sphere centered at \( \mathbf{o} \) and with equal volume (i.e., \( 2.66 \times 10^{-3} \text{ m}^3 \)). More specifically, the drag equation in (10) is now calculated with a constant cross-sectional area \( A \) and coefficient \( C_D \approx 0.4 \) [72, p. 261].

The lift force, no longer calculated via solving the Laplace equation (6), has a form similar to (10):

\[
f_l = \frac{1}{2} \rho AC_l (b \mathbf{v} \cdot b \mathbf{v}) b \mathbf{ω} \times b \mathbf{v},
\]

where \( C_l \approx 0.22 \) [96]. Fig. 12(a) and (b) show reprojection errors from QCKF-A and QCKF-S for the first and second cameras, respectively. The errors are similar until 0.3 s when new vertices appearing in images cause measure inaccuracies of rotation (resulting in a jump of reprojection error for QCKF-S). Even after correction, the QCKF-S yielded a poorer estimate of angular velocity due to not accounting for lift force from air translating over the non-spherical surface. Consequently, the estimated orientation lags behind that of the QCKF-A, resulting in an increase of reprojection error of 6.2 and 4.6 px at the end of estimation for the respective cameras, or approximately 1.5 and 1.1 cm with the polyhedron at a distance of 2.4 m from the cameras.

E. Experiment Summary

The experiments described in this section were conducted with a 4-core 3.5 GHz Intel Xeon CPU and 8 GB of memory. Table III presents computation times for the five estimators to update and correct the state estimate, averaged over all experiments for which they were used. During the update step, two rounds of the fourth order Runge-Kutta method [97], [98] of integration were performed. In the case of QCKF-A, aerodynamic forces are repeatedly calculated by solving a linear system of approximately 3,500 rows and 5,000 columns. It is for this reason that the QCKF-A is most time-consuming.

Since no data sets of real images and corresponding motion data for a free-flying object are available to the best of our knowledge, we rely on the reprojection error \( \sqrt{d(C, x)} \) to represent pose error in the world. Below we argue that no two poses of the object could generate the same pair of images on the two cameras, unless some degeneracy of projection takes place or the object assumes certain symmetry. The image coordinates of three feature points generated by a single camera yield up to six independent equations, each composing the chain in (18), from which the pose can be solved. However, the lack of depth information and imaging noise could still result in pose ambiguities. With another image taken simultaneously, six more constraints are induced and satisfiable by only a finite number of poses. The remaining constraints, including the epipolar constraint discussed in Section IV, further reduce the possible poses to almost always a unique one. The probability
of two different initial states of the object producing the same sequence of image pairs is negligible, if not zero.

VIII. DISCUSSION AND FUTURE WORK

This work presents a scheme for motion estimation under conditions that have previously rendered acquisition of accurate pose and velocity estimates infeasible. During a fast free flight, the object is susceptible to large aerodynamic forces (in the case of low mass density), and observed for only a fraction of a second. Thorough nonlinear models of the object’s dynamics and vision-based observables are presented for use in robust Kalman filtering. Consideration of lift and drag forces yields improved trajectory estimates. Estimation error is further reduced by enforcing multiple quaternion-induced constraints on the state at minimal computational cost. Finally, planar graph matching effectively tracks features of a polyhedron with no fiducial markers, while dynamic updating of hypotheses robustly deals with large image noise. These contributions, though combined into one motion estimation scheme, can also be employed individually to improve accuracy, enforce multiple constraints, or deal with noise.

Our approach to motion estimation was developed in part with robotic manipulation in mind. Unlike in VINS and SLAM, a small margin of error is allowed for estimates to yield predicted trajectories that can lead to successful executions of operations such as batting, catching, etc. We have observed first hand how sensitive the outcome of robotic batting is to error in pose and motion estimates [10, p. 475]. Consider batting of an object with 1 cm of error in its pose estimate. Depending on the object’s geometry, the batting configuration at impact can differ significantly from expected, and the impact period of less than a tenth of a second leaves no room for the robot to make correction during contact, resulting in the object following a different trajectory, and ultimately, a failed batting outcome. Their often known contact, resulting in the object following a different trajectory, expected, and the impact period of less than a tenth of a second. Thorough consideration of lift and drag forces yields improved trajectory estimates. Estimation error is further reduced by enforcing multiple quaternion-induced constraints on the state at minimal computational cost.

Computational cost can hinder real time estimation. Table III shows that the QCKF operated at 110 Hz, and the QCKF-A with repeated calculations of aerodynamic forces operated at just below 10 Hz. An order of magnitude improvement could be achieved by a combination of modern computing hardware, tuning of parameters, utilization of GPU processing (e.g., in calculation of lift force), and optimizations of the QCKF to reduce the number of matrix calculations [45].

A few potential directions are in line for future work. First, generalized to irregular polyhedral objects, the approach can be further extended to curved objects and objects bounded by multiple curved patches. Features such as high curvature points, discontinuities between surface patches, textures, or fiducial markers can provide the observables required for estimation. Second, the work can be utilized in high-speed manipulations such as batting and catching or in robot sports such as ping pong, tennis, etc. For instance, robotic catching would benefit from the accuracy of model-based estimation. Existing methods fail to determine a catching configuration for objects thrown along trajectories that deviate from those used to build machine learned models [8].

Moreover, the use of stereo vision allows the motion estimation approach to be versatile. Two cameras can be mounted on a humanoid robot to perform motion estimation of a flying ball in sports for example. As long as features on the ball can be detected in its images, its trajectory can be perceived. Cameras may be mounted on other mobile robots or vehicles for estimation where additional parameters may need to be estimated for localization, or fixed to the world to perceive motion in a scene. The estimation scheme is applicable to tasks for space robots, where aerodynamic forces can be ignored due to scarcity or absence of air.

APPENDIX A
INITIALIZATION OF 6-DOF POSE

Once a new hypothesis is created, we estimate the object’s pose, composed of position \( o^{(k)} \) and rotation \( r^{(k)} \) at the \( k \)-th time instant. The estimates will be used by a Kalman filter created under the hypothesis.

Stereo cameras viewing a point \( p \) induce the epipolar geometry described in Section IV, which allows for a pair of image points \( i'p \) and \( i'p \), respectively generated by the first and second cameras, to be back-projected to some (but not necessarily the same) point in the world. The chain of equations in (18) are solved in reverse order to obtain the world coordinates \( \hat{w}p \) of their common source point. In particular, the second step of the solution involves undistortion of \( \hat{w}p \) and \( \hat{w}p \), yielding \( \hat{w}p \) and \( \hat{w}p \). To recover the point’s third coordinate, we perform “rectification” of the points \( \hat{w}p \) and \( \hat{w}p \) to transform their image planes to be coplanar [99, p. 159]. To that effect, the transformed points lie on a line parallel to the line through the camera centers, allowing for “triangulation” to determine their z-coordinate [100].

With a set of back-projected points and the set of object’s vertex coordinates in \( F_b \), the correspondence between them is known under the new hypothesis \( H_{\text{new}} \). We invoke Horn’s method [101] to compute the translation \( o^{(k)} \) and rotation \( r^{(k)} \) of the frame \( F_b \) from \( F_w \).

\[ \text{with a possible loss in accuracy due to errors in measurement of depth} \]

\[ \text{with a possible loss in accuracy due to errors in measurement of depth} \]

\[ \text{with a possible loss in accuracy due to errors in measurement of depth} \]

\[ \text{with a possible loss in accuracy due to errors in measurement of depth} \]