Abstract—We investigate the problem of dynamically estimating the 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 the dynamics with the camera projection model, via the use of multiple quaternions, under the constraint of two-view geometry. The state of the composed system can then be estimated via constrained Kalman filtering. To keep track of appearing and disappearing visual features during the 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 a stepper motor or multiple accelerometers, or formed verification by matching flight images against projections based on the state estimates.

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 quickly move 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 is sufficient 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. Can fast and accurate motion estimation of a flying object be achieved using low cost camera hardware? 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 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 properties. Consider instead two cameras equipped with wide angle lenses to capture a large field of view and arranged in stereo configuration to provide depth information. Since the flight duration can be less than one second, the cameras are desired to operate at 120 fps or higher in order to produce enough images of the object for estimation. This camera configuration is meanwhile rather inexpensive.

Support for this research was provided in part by the US National Science Foundation under Grant IIS-1421034. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Matthew Gardner is with Boston Dynamics Inc., Mountain View, CA 94043 (email: matthew@mgardner.me).

Yan-Bin Jia is with the Department of Computer Science, Iowa State University, Ames, IA 50010 USA (e-mail: jia@iastate.edu).

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

Yan-Bin Jia is with the Department of Computer Science, Iowa State University, Ames, IA 50010 USA (e-mail: jia@iastate.edu).

Matthew Gardner is with Boston Dynamics Inc., Mountain View, CA 94043 (email: matthew@mgardner.me).

Abstract—We investigate the problem of dynamically estimating the 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 the dynamics with the camera projection model, via the use of multiple quaternions, under the constraint of two-view geometry. The state of the composed system can then be estimated via constrained Kalman filtering. To keep track of appearing and disappearing visual features during the 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 a stepper motor or multiple accelerometers, or formed verification by matching flight images against projections based on the state estimates.

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 quickly move 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 is sufficient 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. Can fast and accurate motion estimation of a flying object be achieved using low cost camera hardware? 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 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 properties. Consider instead two cameras equipped with wide angle lenses to capture a large field of view and arranged in stereo configuration to provide depth information. Since the flight duration can be less than one second, the cameras are desired to operate at 120 fps or higher in order to produce enough images of the object for estimation. This camera configuration is meanwhile rather inexpensive.
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 motion estimates. Lift and drag forces were approximated for the dynamics of an aircraft in estimating its 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]. 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 do not generalize to irregular shapes such as a polyhedron. Machine learning approaches determine correspondences between features from observations and those on a known model of the object that constrained the state’s statistics [41].

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 the Kalman filter. Model-based approaches determine correspondences between features from observation and those on a known model of the object. Model points projected onto an image yield predicted features to be used in a local search for the observed features. The disparity between the two is used to estimate the object’s pose via weighted least squares [42], [43], non-linear minimization [44], or statistical modeling [45], [46], [47]. Model-based pose tracking was applied to estimate the pose of a spacecraft with different ways of initialization [48]. In general, these approaches, by minimizing measurement errors, led to frame rates that were too low for high-speed estimation (up to 30 Hz), and also suffered from erratic performance in case of large disparities between observed and predicted features. Graph matching was introduced to make use of the topological structure of the object’s shape. Subgraph isomorphism is employed to match a “projection” graph constructed from images with a model graph, where edges and vertices contained attributes pertaining to features [49]. Approximate graph matching algorithms were developed to produce sub-optimal solutions to object recognition, while remaining tolerant to measurement errors and image noise [50], [51]. Recently, deep neural networks were trained on synthetic data to track stationary objects with occlusions [52], [53].

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 by low-density objects, and image noise due to fast movements. Secondary contributions include:

- a graph-based feature tracking algorithm using hypotheses to deal with uncertainties,
- Kalman filtering under multiple constraints, and
- approximation of aerodynamic forces for general shapes.

Fig. 1 illustrates the interaction between three components of the scheme: constrained Kalman filtering, graph-based feature tracking, and aerodynamics. 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 discusses the system dynamics of a free-flying object and computation of aerodynamic forces affecting its motion. Section III introduces the camera projection model and geometric constraint enforced by stereo cameras to produce observables of the object. Section IV gives the Kalman filter algorithm that enforces multiple quadratic constraints on the estimated state while making use of the models from Sections II and III. Section V describes a graph-based feature tracking algorithm for identifying image-to-model correspondences that are required to carry out estimation. Section VI presents results from four experiments.
of increasing complexity, conducted with a plastic cuboid, wooden frame, rugby ball, and foam polyhedron. Section VII concludes the paper with discussions on improvements and future directions for dynamic estimation.

A. Other Related Work

Simultaneous localization and mapping (SLAM) algorithms solve a similar problem of estimating the pose and motion of a robot relative to its surroundings. Kalman filter-based approaches are common, making use of visual and inertial sensors mounted on board an autonomous aircraft or ground vehicle. SLAM algorithms produce a map of 3D features in the world, typically by augmenting the system state to include camera poses or coordinates of feature points. MonoSLAM used a single camera with an EKF to estimate a robot’s pose, velocity, and angular velocity, along with 3D feature points [54]. Other approaches tracked the image coordinates of features in the estimated state, and described 3D-to-2D projections in the measurement model of an EKF [55] or UKF [56]. The estimated state vector could be quite large, requiring high computation time to produce a map of the world.

Similarly, 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 [57], [58]. Measurements of position in the world were obtained via optimization from features observed by translating cameras, while orientation measurements were provided by IMU readings. Stereo vision was employed to improve angular velocity and position estimation over traditional approaches via sensor fusion in a UKF [59].

B. Notation

A vector (or a point) is by default a column vector, written by a lowercase bold letter, e.g., \( \mathbf{v} \). A unit vector has a hat, e.g., \( \hat{v} = v/\|v\| \). The left superscript of a point (or a vector), if exists, denotes the frame in which it is expressed. For example, \( \mathbf{w}p \) gives the coordinates of a point \( p \) in the world frame (denoted by \( \mathbf{w} \)), while \( \mathbf{b}p \) gives its coordinates in the body frame (denoted by \( \mathbf{b} \)). A scalar (or vertex in Section V) is written by a lowercase non-bold letter, and a matrix (or a vector) named by the same letter (bold). For example, \( p_x \) denotes the \( x \)-coordinate of the point \( p \). The subscript \( x \), \( y \), and \( z \) (if 3D) 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, \( p_x \) is written in bold, while a scalar function is non-bold. The superscripts ‘−’ and ‘+’ are used to refer to the prior and posterior estimates produced by a Kalman filter.

II. 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. Under the frame, its angular inertia matrix \( Q \) is diagonalized. The rotation of \( \mathcal{F}_b \) from some world frame \( \mathcal{F}_w \) is described by a unit quaternion \( r \). Let \( \mathbf{b}v \) and \( \mathbf{b}\omega \) 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: \( \mathbf{w}v = r(\mathbf{b}v)r^* \) where \( r^* \) is the conjugate of \( r \).

Denote by \( wf \) the non-gravitational external force on the object in the world frame \( \mathcal{F}_w \). Newton’s equation takes on the form

\[
\mathbf{w}\ddot{\mathbf{v}} = \mathbf{g} + \frac{w}{m} \quad \text{(1)}
\]

where \( \mathbf{g} \) is the gravitational acceleration vector. Euler’s equation assumes the form \( Q \mathbf{b}\ddot{\omega} + \mathbf{b}\omega \times (Q \mathbf{b}\omega) = \mathbf{b}\tau \), where \( \mathbf{b}\tau \) is the external torque in the body frame. We have

\[
\mathbf{b}\omega = Q^{-1}(\mathbf{b}\tau - \mathbf{b}\omega \times (Q \mathbf{b}\omega)) \quad \text{(2)}
\]

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 of the object at a time instant during its flight. The object, viewed “stationary”, is experiencing a flow of air with velocity \( -\mathbf{b}v \) expressed in its body frame \( \mathcal{F}_b \). There are two contributed forces: lift, denoted \( \mathbf{b}f_l \), and drag, denoted \( \mathbf{b}f_d \), both described in \( \mathcal{F}_b \).

Let \( S \) denote the object’s surface in the frame \( \mathcal{F}_b \). Lift force acts on the object due to air particles traveling smoothly over \( S \) and producing pressure forces normal to the surface. Bernoulli’s equation [60, pp. 20-21], after elimination of constant static pressure, gives the following relationship between air pressure \( p \) and air speed \( u \):

\[
p = -\frac{1}{2} \rho u^2,
\]

where \( \rho \) is air density. The total lift force is then calculated by integrating this pressure over \( S \) as follows:

\[
\mathbf{b}f_l = \frac{1}{2} \rho \int_S (\mathbf{b}u(s) \cdot \mathbf{b}u(s)) \mathbf{n}(s) \, ds,
\]

where \( \mathbf{b}u(s) \) is the air velocity at a surface area element \( s \) with its exact form yet to be determined, and \( \mathbf{n}(s) \) is the unit normal to the surface element. In addition, the accumulation of these forces about the center of mass \( o \) yields a torque:

\[
\mathbf{b}\tau_l = \frac{1}{2} \rho \int_S (\mathbf{b}u(s) \times \mathbf{b}u(s)) \, ds \quad \text{(5)}
\]

Let \( \mathbf{b}e \) denote a point within an instantaneous neighborhood \( N \) of the object in which the air flow is affected by its presence. Under the assumption that the air flow is incompressible and irrotational [61, p. 100] at the point, the air velocity is
\[ \nabla \phi (b^e) \text{, where } \phi \text{, known as the velocity potential, satisfies the Laplace equation below:} \]
\[ \nabla^2 \phi (b^e) = 0, \]  
(6)
where \( \nabla^2 \) is the Laplacian operator. Denote by \( \partial N \) the boundary surface of the neighborhood \( N \). The Laplace equation (6) has two boundary conditions:
\[ \phi (b^e) = -b v, \text{ for all } b^e \text{ on } \partial N, \]  
(7)
\[ \nabla \phi (b^e) \cdot \hat{n}(b^e) = 0, \text{ for all } b^e \text{ on } S. \]  
(8)
Calculation of \( \phi \) involves numerically solving the above differential equation for all the points within the neighborhood \( N \) containing the object. This is detailed in Appendix B.

Meanwhile, within a boundary layer near the object’s surface, the viscosity of air exists, producing friction with the body’s surface. This leads to two effects inside the thin region [60, 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. Combining the component with the air velocity outside of the boundary layer from equation (6), we have
\[ b u (b^e) \approx \nabla \phi (b^e) + b \omega \times b e, \]  
(9)
for a point \( b^e \) within the boundary layer. To determine the air pressure at the point, we apply Bernoulli’s equation (3) with \( u = \| b u (b^e) \| \).

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 [61, pp. 211–215, 339]:
\[ b f_d = -\frac{1}{2} \rho C_d A (b v \cdot b v) \hat{b} v, \]  
(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 \( C_d \) is dependent on the object’s shape and is determined from studies on the aerodynamics of general shapes [62], [63]. To compute the area \( A \), all the points on the object’s surface are projected orthographically onto a plane with normal \( -\hat{b} v \). 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
\[ w f = r (b f_1 + b f_d) r^*, \]  
(11)
\[ b \tau = b \tau_1. \]  
(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 \( b v \). The resulting potential field \( \phi \) is used along with \( b \omega \) to calculate the air velocity \( b u (b^e) \) via equation (9). Bernoulli’s equation calculates the pressure \( p \), and is integrated over the surface to yield the lift force \( b f_1 \) and torque \( b \tau_1 \). The drag force \( b f_d \) is simultaneously computed from the velocity estimate according to (10), Newton’s and Euler’s equations are then integrated to yield new velocity estimates for a new round of calculations.

The state of the flying object is described by the 13-vector
\[ \xi = \begin{pmatrix} o \\ w v \\ b \omega \end{pmatrix}. \]  
(13)
The state is subject to a constraint that its quaternion component \( r \) keeps unit length, i.e. \( |r| = 1 \). Moreover, \( r \) has the following derivative given in Appendix C of [64]:
\[ i \dot{r} = \frac{1}{2} r^*(b \omega). \]  
(14)
This, together with \( \dot{\phi} = w v \), (1), and (2), forms the following system of nonlinear differential equations:
\[ \dot{\xi} = a(\xi) = \begin{pmatrix} w v \\ \frac{1}{2} r^*(b \omega) \\ g + \frac{m}{f} \end{pmatrix} Q^{-1} (\tau - b \omega \times Q(b \omega)). \]  
(15)

III. OBSERVABLES FROM VISION

The object’s state \( \xi \) is estimated based on measurements extracted out of its images taken simultaneously by two cameras. Observables are produced by projection from a single camera, accommodating the geometry induced by the locations of the two cameras in stereo vision configuration. Parameters of camera poses can also be included as observables to help reduce estimation errors.

A. Single Camera Projection Model

A single camera observing the flying object is shown at the bottom left in Fig. 4. At the camera’s focal point \( ^w c \) is a frame \( F_c \) with its z-axis 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 coordinate system with the origin at the upper left corner of the image, the u-axis pointing rightward, and the v-axis pointing downward. We use the projection model outlined in detail by Forsyth and Ponce [65, pp. 16–17]. Let \( p \) be a point on the object, and denote \( ^0 p, ^w p \) and \( ^c p \) as its coordinates in the body, world, and camera frames, respectively. They assume the following mappings:
\[ w p = o + r (b p) r^*, \]  
(16)
\[ ^c p = r_c^* (w p - w c) r_c. \]  
(17)
Letting \( \hat{c}p = (\hat{c}p_x, \hat{c}p_y, \hat{c}p_z)^\top \), the pinhole camera model yields the normalized image coordinates

\[
r_p = (\frac{\hat{c}p_x}{\hat{c}p_z}, \frac{\hat{c}p_y}{\hat{c}p_z}) = \frac{1}{\hat{c}p_z} (\hat{c}p_x, \hat{c}p_y),
\]

(18)

Brown’s distortion model [66] is then used to map the undistorted point \( r_p \) to a distorted point \( \hat{r}_p \):

\[
\hat{r}_p = \delta_r (r_p) + \delta_t (r_p),
\]

(19)

where, denoting \( \theta = ||r_p|| \), the distortion functions are defined as

\[
\delta_r (r_p) = (1 + k_1 \theta^2 + k_2 \theta^4 + k_3 \theta^6) (\frac{1}{r_p}) \hat{r}_p, \quad (20)
\]

\[
\delta_t (r_p) = (h_1 (\theta^2 + 2 \theta ^2 \hat{r}_p^2) + h_2 (\theta^2 + 2 \theta^2 \hat{r}_p^2)),
\]

(21)

for some coefficients \( k_1, k_2, k_3 \) of radial distortion, and \( h_1, h_2 \) of tangential distortion.

To convert the point to image coordinates, several intrinsic parameters are needed: scale factors \( \alpha \) and \( \beta \), the skew angle \( \theta \), and the image center \((w_c, v_c)^\top\). The distorted point is then transformed into image coordinates:

\[
i_p = \begin{pmatrix} u \\ v \end{pmatrix} = \frac{\alpha - \alpha \cot \theta}{\beta \sin \theta} (\hat{r}_p) + \begin{pmatrix} w_c \\ v_c \end{pmatrix} \). \quad (22)
\]

Due to manufacturing inaccuracies of cameras, all five parameters in (22) have to be approximated to account for various sources of error.

Note that \( b_p, w_p \) and \( c_p \) are 3-vectors with \( b_p \) being determined beforehand (as the position of a feature on the object), while \( r_p, \hat{r}_p \), and \( i_p \) are 2-vectors. The sequence of transformations is best summarized as follows:

\[
\begin{array}{l}
\hat{b}_p \rightarrow w_p \rightarrow c_p \rightarrow r_p \rightarrow \hat{r}_p \rightarrow i_p. \\
\end{array} \quad (23)
\]

**B. System Observables**

Let us add a second camera so the two cameras are in a stereo vision configuration to produce different views of the flying scene. Shown at the bottom right in Fig. 4, the second camera has focal point \( c' \), and its body frame \( F_{c'} \) has a relative orientation described by the quaternion \( q_{c'} \) to the world frame \( F_w \). We denote by \( c' \) the coordinates of \( p \) in the frame of the second camera, \( \hat{c}p \) the corresponding normalized image coordinates, \( \hat{\hat{r}}_p \) the distorted point by this camera, and \( \hat{i}_p \) the final image point. The point \( p \) undergoes the following sequence of transformations due to the second camera:

\[
w_p \rightarrow c' \rightarrow r' \rightarrow \hat{r}' \rightarrow i_p.
\]

As illustrated in Fig. 4, two rays, originating at \( p \) on the object and passing through the two focal points \( c \) and \( c' \), intersect the image planes at two points \( q \) and \( q' \), respectively. We have

\[
wq - wc = \gamma (wp - wc)
\]

(24)

for some \( \gamma > 0 \), where

\[
t = r_c (\hat{r}_p, 1)^{\top} r_c^{\ast}.
\]

Similarly, it follows that

\[
wq' - wc' = \gamma' c_p z',
\]

(25)

for some \( \gamma' > 0 \), where

\[
t' = r_c (\hat{r}_p, 1)^{\top} r_c^{\ast}.
\]

The five points \( p, c, c', q, \) and \( q' \) are in a plane called the epipolar plane. We express this coplanarity using the latter four points which are either known for the cameras or obtainable from the images:

\[
(w c' - w c) \cdot ((wp - wc) \times (w q - wc)) = 0.
\]

(26)

The unit normal \( \hat{n} \) to this epipolar plane is determined as

\[
\hat{n} = \frac{t \times t'}{||t \times t'||}.
\]

(27)

Substitute (24) and (25) into (26) and then get rid of \( \gamma, \gamma', c_p z, \) and \( \gamma' c_p z \) on the left hand side of the resulting equation. After several steps of manipulation and with introduction of the rotation matrix \( U \) corresponding to \( r_c^{\ast} r_{c'} \), we arrive at the following equation [65, pp. 200] with \( E = MU \).

\[
\begin{array}{l}
wp \rightarrow c' \rightarrow r' \rightarrow \hat{r}' \rightarrow i_p.
\end{array} \quad (23)
\]

where \( E = MU \). Here, \( M \) is the anti-symmetric matrix which, when multiplied with any 3-vector \( u \), yields the cross product \( (w c' - wc) r_c \times u \). The \( 3 \times 3 \) matrix \( E \), referred to as the essential matrix, is skew-symmetric and has rank two [67], [68]. Equation (28) is known as the epipolar constraint, and is enforced over the camera parameters during calibration. To apply the constraint, image points are first back-projected and undistorted into the ideal pinhole model. More specifically, given \( \hat{r}_p \), we recover \( \hat{r}_p \) by solving (19) and then \( r_p \) by solving (18). The coordinates \( \hat{r}_p \) are similarly recovered. This two-step process is carried out using the Levenberg-Marquardt algorithm for nonlinear least-squares minimization [69], [70].

**Fig. 4.** A camera captures images of the object by projecting it onto the image plane \( \Pi \) about its center of projection \( c \) where the camera’s frame of reference \( F_c \) is located. The \( x \)- and \( y \)-axes of \( F_c \) are aligned with the \( u \)- and \( v \)-axes in the image plane, and the \( z \)-axis in the normal direction. The distance between \( c \) and \( \Pi \) is equal to the focal length of the camera. Additionally, a second camera (referred to by use of a single quote ‘) with center of projection \( c' \) and frame of reference \( F_{c'} \) is illustrated, producing the two-view geometry of a point \( p \) from which the rays to \( c \) and \( c' \) intersects the image planes at \( q = (wp^\top, 1)^\top \) and \( q' = (wp^\top, 1)^\top \). The points \( c, c', \) and \( p \) form the epipolar plane drawn with the normal \( \hat{n} \).
The camera projection model is used to calculate a vector $y$ of observables from point features on the object that are visible. Let $b_1, \ldots, b_n$ be all the feature points on the object. Clearly, not all of them will be observed as occlusion is inevitable. Let $V$ and $V'$ be the sets of features that are currently visible in the respective images produced by the two cameras. The vector $y_k$ gathers two types of observables. Those of the first type are point features forming the vector

$$y_a = (\ldots, b_i^T, \ldots, y_j^T, \ldots)^T,$$

for all $b_i \in V, b_j \in V'$.

Observables of the second type are derived as follows. From a feature point $w_i p_j$ come out two rays to the focal points $w_i c$ and $w_i c'$ that intersect the image planes at the points $w_i q_j$ and $w_i q_j'$, respectively (illustrated in Fig. 4 for $p$). The epipolar normal $n_j$ of the plane through these five points is then obtained using (27). Since the image points $b_i p_j$ and $b_i p_j'$ are subject to noise, the corresponding epipolar constraint is not exactly satisfied. Instead, we include the normal $n_j$ to indirectly enforce the constraint. The second vector of observables $y_b$ stacks together the normals for vertices visible in both images, that is,

$$y_b = (\ldots, n_j^T, \ldots)^T,$$

for all $b_j \in V \cap V'$. The full vector of observables is

$$y(\xi) = \begin{pmatrix} y_a \\ y_b \end{pmatrix}.$$

IV. Kalman Filtering with Quadratic Constraints

Kalman filter-based motion estimation relies on the dynamics model (33) to propagate the state of the object with time, and the measurement model described in Section III-B to correct the state for observables obtained from camera sensors. Visual inputs are acquired at discrete time instants. This, plus the need to cope with nonlinearities present in both models, leads to the use of the hybrid extended Kalman filter [34, pp. 403–407].

Denote by $x_k$ the value of the state $x$, and $y_k$ the vector of observables, all at the $k$-th time instant. The size $m$ of $y_k$ may vary with the features that are currently observable. The measurement model can be written as

$$y_k = h(x_k, \nu_k),$$

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 governed by equations (33) and (34) estimates the distribution of the state $x$. Starting with the update at the time instant, it forward integrates the linearized dynamics derived from (33) using the Jacobian of $b(x)$, which is 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 $k$. Obtain its $27 \times 27$ error covariance matrix $P_k^-$. The estimate $\hat{x}_k$ is then corrected to yield a posterior estimate $\hat{x}_k^+$ based on the measurement residual $\epsilon_k = y_k - h(\hat{x}_k)$ as follows:

$$\hat{x}_k^+ = \hat{x}_k + K_k \epsilon_k,$$

where $K_k$ is a $27 \times m$ gain matrix to be determined. Using (35), we write out the $27 \times 27$ posterior error covariance matrix $P_k^+$ for $\hat{x}_k^+$ in terms of $P_k^-:

$$P_k^+ = (I - K_k H_k) P_k^- (I - K_k H_k)^\top + K_k R_k K_k^\top ,$$

where $I$ is the $27 \times 27$ identity matrix, and the $m \times 27$ matrix $H_k$ is the linearized measurement model, computed as the Jacobian of $h(x)$ at $\hat{x}_k$. Minimization of $\text{tr}(P_k^+)$ yields the Kalman gain

$$K_k = P_k^- H_k^\top (H_k P_k^- H_k^\top + R_k)^{-1}.$$

With $K_k$ determined, $\hat{x}_k^+$ and $P_k^+$ are then evaluated according to equations (35) and (36) to conclude the updates at the $k$-th time instant.

A. Constrained Kalman Filtering

When the system state is subjected to some constraints, the unconstrained estimator (35)–(37) performs corrections that would cause the estimate to drift away from the constraint surface. To resolve this, the constraints are incorporated into the minimization problem for determining the Kalman gain $K_k$, yielding an update that produces a constrained state estimate. The augmented state $x$ in equation (32) 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 4-vector that has a unit inner product with itself, inducing constraints

$$\eta_h(x) \equiv x^\top A_h x - 1 = 0,$$
for \( i = 1, 2, 3 \), where
\[
A_1 = \text{diag}(0_3, 1_4, 0_3, 0_3, 0_4, 0_3, 0_3, 0_3), \\
A_2 = \text{diag}(0_3, 0_4, 0_3, 0_3, 1_4, 0_3, 0_3), \\
A_3 = \text{diag}(0_3, 0_4, 0_3, 0_4, 1_4, 0_3, 0_3).
\]

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 (38). This is done by computing the \( 27 \times m \) matrix \( K_k \) through the following constrained minimization:
\[
\min_{R_k} \text{tr} \left( P_k^+ \right) \quad \text{subject to} \quad \eta_i(\hat{x}_k^+) = 0, \ i = 1, 2, 3. \tag{39}
\]

Previous works [38], [39] addressed a single constraint only. Here, we will obtain a solution to (39) 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 omit the subscripts in \( x_k^+, x_k, K_k, \epsilon_k, P_k^+, P^- \), and \( R_k \). Substituting equation (35) in with \( x = \hat{x}^+ \), the constraints in (38) are expanded into
\[
\eta_i(\hat{x}_k^+) = (\hat{x}^- + K\epsilon)\top A_i(\hat{x}^- + K\epsilon) - 1 = 0. \tag{40}
\]

We write \( \eta = (\eta_1, \eta_2, \eta_3)\top. \) From equations (36) and (40), the Lagrangian for the constrained minimization problem (39) is
\[
L = \text{tr} \left( (I - KH)P^-(I - KH)\top + KRR\top \right) + \lambda\top \eta, \tag{41}
\]
where the vector \( \lambda = (\lambda_1, \lambda_2, \lambda_3)\top \) contains three Lagrange multipliers, one for each constraint in (38). 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 (38), for \( i = 1, 2, 3 \). To obtain \( \partial L/\partial K \), we first have
\[
\frac{\partial}{\partial K} \text{tr} \left( KRK\top \right) = 2KR, \tag{42}
\]
where we made 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 differentiate the trace of the first matrix product in (41), making use of the symmetry of \( P^- \),
\[
\frac{\partial}{\partial K} \text{tr} \left( (I - KH)P^-(I - KH)\top \right) = \frac{\partial}{\partial K} \text{tr} \left( KHPP^-KH\top \right) - 2 \frac{\partial}{\partial K} \text{tr} \left( P^-H\top K\top \right) = 2KHPP^-H\top - 2P^-H\top. \tag{43}
\]

In the meantime, differentiation of (40) yields
\[
\frac{\partial}{\partial K} (\eta_i(\hat{x}_k^+)) = 2A_i \hat{x}^- \epsilon\top + 2A_i K \epsilon \epsilon\top. \tag{44}
\]

In (44), 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 (42)–(44) we can write out the partial derivative \( \partial L/\partial K \):
\[
\frac{\partial L}{\partial K} = 2 \left( KHP^-H\top + KR - P^-H\top + \sum_{i=1}^{3} \lambda_i \left( A_i \hat{x}^- \epsilon\top + A_i K \epsilon \epsilon\top \right) \right). \tag{45}
\]

Vanishing of \( \partial L/\partial K \) is expressed by the equation
\[
KW + N(\lambda)K \epsilon \epsilon\top = M(\lambda), \tag{46}
\]
where
\[
W = HP^-H\top + R, \\
N(\lambda) = \sum_{i=1}^{3} \lambda_i A_i, \\
M(\lambda) = P^-H\top - \sum_{i=1}^{3} \lambda_i A_i \hat{x}^- \epsilon\top.
\]

The remaining task is to solve (40) and (46) for \( K \) and \( \lambda \). First, we right multiply equation (46) by \( W^{-1}\epsilon \). This gives us
\[
K \epsilon = J(\lambda)^{-1}m(\lambda), \tag{47}
\]
where \( \epsilon = \epsilon\top W^{-1}\epsilon \) and
\[
J(\lambda) = I + \epsilon N(\lambda), \\
m(\lambda) = M(\lambda)W^{-1}\epsilon.
\]

Substitute (47) into (40) to eliminate \( K \) and obtain
\[
f_i(\lambda) \equiv \eta_i(\hat{x}^-) + 2(\hat{x}^-)\top A_i J(\lambda)^{-1}m(\lambda) + (J(\lambda)^{-1}m(\lambda))\top A_i J(\lambda)^{-1}m(\lambda) = 0,
\]
for \( i = 1, 2, 3 \). Rewrite the three equations (48) as
\[
f(\lambda) = 0, \tag{49}
\]
where \( f = (f_1, f_2, f_3)\top \). The system (49) 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 [71]. These solvers make use of the Jacobian \( \partial f/\partial \lambda \), which requires the following partial derivatives, \( i, j = 1, 2, 3 \):
\[
\frac{\partial m}{\partial \lambda_j} = -\epsilon A_i \hat{x}^-,
\]
\[
\frac{\partial J^{-1}}{\partial \lambda_j} = -\epsilon J^{-1} A_j J^{-1}. \tag{51}
\]

Equation (51) was derived from differentiating \( J^{-1}J = I \) while making use of the symmetry of \( J \).

To summarize, at time instant \( k \), the Lagrange multipliers \( \lambda \) are first initialized by solving (49) via homotopy continuation. The gain matrix \( K \) can then be solved from equation (46) after substitution of equation (47) on the left hand side. The matrix is used in (35) and (36) to produce corrected state estimates that satisfy the set of quadratic constraints. At the next time instant, new observables are obtained, but the multipliers’ values are expected to change slightly. Hence, their values at time instant \( k - 1 \) can be used as a starting point by Newton’s
method for quick updates. In the case of divergence, we fall back to homotopy continuation.

The optimization procedure is easily adapted to work with quadratic constraints in the general form of $x^T A_i x + b_i^T x + c_i = 0$, $1 \leq i \leq m$, where $A_i$ is not necessarily symmetric. We need only make several changes described below. In equations starting at (44), every appearance of $A_i$ will be replaced with $(A_i + A_i^T)/2$. The partial derivative in (44) will include an extra summand $b_i e^T$, and this change will propagate to (45) and (46) and, subsequently, to the expression for $M(\lambda)$.

V. 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. The system dynamics and observation function have non-varying forms. Such continuity allows the working of a single Kalman filter. This is often not true, however, in the course of the object’s flight as features emerge and disappear, which introduces discontinuities in observations. This section addresses the problem of identifying and tracking features of the object in images with respect to those on its shape model. It is known that any shape can be approximated arbitrarily well by some polyhedron or polyhedral mesh (i.e., a triangular mesh). To avoid an unnecessary diversion into handling geometric complexities, in this section we focus on tracking when the object is a convex polyhedron $P$.

The polyhedron $P$ has $n$ vertices $p_0, \ldots, p_{n-1}$. Their connectivities are visualized by unfolding the surface of $P$ onto a planar graph, referred to as the model graph $M$, which is represented as a doubly-connected edge list (DCEL) [72, pp. 29–33]. The process turns every facet of $P$ into a bounded region in $M$, except for one facet, which has been “torn open” to become the unique unbounded region. As shown in Fig. 5(a) and (c) for the case of an octahedron, every vertex $v_i$ in $M$ corresponds with the vertex $p_i$ of $P$.

The two cameras are synchronized to produce images (or “frames”) of the polyhedron at equally spaced time instants. A frame generated at the $k$-th time instant, say, by the first camera, is a light intensity image of $P$ with its faces illuminated differentially and showing discontinuities at edges and vertices. These edge and vertex features are used to construct a planar graph $G_k$, referred to as the image graph, as illustrated in Fig. 5(b).

We denote $(u_i, v_j)$ if the vertex $u_i$ in $G_k$ corresponds to the vertex $v_j$ in the model graph $M$. The tracking algorithm hypothesizes a set of vertex correspondences $C(G_k, M) = \{\ldots, (u_i, v_j), \ldots\}$. This set represents a bijective mapping from a vertex subset of $G_k$ to one of $M$. Meanwhile, another set of vertex correspondences $C_0(G_k, M)$ is hypothesized between $G_k'$, the image graph produced by the second camera, and $M$. The two sets together form a hypothesis $H = (C(G_k, M), C(G_k', M))$.

The hypothesis $H$ at time instant $k$ is first inherited from one either active at time instant $k-1$, or created at the end of processing for that time instant. The correspondence sets $C(G_k, M)$ and $C(G_k', M)$ in the hypothesis are then updated based on two new image frames taken simultaneously by the two cameras. The vector $y_k$ of observables, constructed from the image points of vertices in $G_k$ and $G_k'$, is then updated under the changed correspondence sets. In the case where either of the two correspondence sets includes a new correspondence or removes an existing one, the structural change is propagated to $y_k$ as well as to the measurement model of the associated Kalman filter $K$. Next, the filter $K$ corrects its prior state estimate $x^+_k$, which has evolved from the posterior estimate $x^+_{k-1}$ at time instant $k-1$ through the system dynamics, to yield the posterior estimate $x^+_{k}$.

Section V-A gives details on how to update $H$.

The updated estimate $x^+_k$ is applied to the object. A graph $G_k^{'}$ is constructed from the vertices in $M$ that are represented in $G_k$, with each assigned image coordinates computed from the transformation specified within $x^+_k$ and then the projection by the first camera. Similarly, $x^+_{k}$ is used to construct an image graph $G_k'$ from $G_k'$. The error $\epsilon(H)$ of the hypothesis $H$ combines the differences between $G_k$ and $G_k^{'}$ and those between $G_k'$ and $G_k'$. The hypothesis is passed on to the $(k+1)$-st time instant only if the error is within some threshold. Hypothesis evaluation is described in Section V-B.

At the $k$-th time instant, a collection $H$ of hypotheses (including $H$) are active, each equipped with a Kalman filter responsible for estimating the object’s pose and motion. In the case that all active hypotheses have been eliminated at the time instant, new ones will be generated before moving on to the next time instant. This is described in Section V-C.

A. Updating a Hypothesis

Let us go back to $H$ as one of the hypotheses active at time instant $k$. It was passed on from time instant $k-1$. (The hypothesis could be newly generated at the end of processing for the previous time instant, or at the very beginning of tracking in case $k = 1$.) Image graphs $G_k$ and $G_k'$ are respectively constructed from the two new images taken by the cameras at time instant $k$ as described earlier. In such a new image, vertices and edges of the polyhedron may appear or disappear as its pose has changed from time instant $k-1$. Thus, there is a need for updating the correspondence sets $C(G_k, M)$ and $C(G_k', M)$ based on this new pair of image frames. Below we only describe how $C(G_k, M)$ is updated from $C(G_{k-1}, M)$, since $C(G_k', M)$ can be similarly updated.
The first stage of updating $C(G_{k-1},M)$ starts by computing the maximal correspondence between $G_k$ and the image graph $G_{k-1}$ at instant $k-1$. This requires some initial vertex correspondences between $G_{k-1}$ and $G_k$, which can be obtained during image processing. Using the set $C(G_{k-1},G_k)$ of these correspondences as well as the correspondence set $C(G_{k-1},M)$ inherited from time instant $k-1$, we initialize the correspondence set $C(G_k,M)$ for $G_k$ in the first stage. Fig. 6 illustrates the relationship between the three sets of correspondences. In the second stage, $C(G_k,M)$ is extended to a maximal correspondence set via mapping the image graph $G_k$ with the model graph $M$.

Both stages involve computing the correspondence between two planar graphs, first $G_{k-1}$ and $G_k$, and then $G_k$ and $M$. The problem of computing the correspondence between two planar graphs is one of subgraph isomorphism, which can be done in linear time [73]. However, due to the presence of noise in images, some tolerance to erroneous edges and vertices must be allowed. The matching problem becomes NP-complete, based on deciding whether an edge or vertex should be considered erroneous or not. Approaches to error-tolerant graph matching include tree search algorithms in exponential time [49, 74, 75], and sub-optimal algorithms in polynomial time [51, 76].

We propose a tree search algorithm to extend $C(G_k,M)$. To also address generation of a new hypothesis, the algorithm starts with $C(G_k,M) = \emptyset$. A depth-first search of the decision tree is performed by matching edges of $G_k$ with different edges of $M$. Fig. 7 demonstrates a portion of the search performed on an example pair of $G_k$ and $M$. The traversal starts with $\{(u_0,v_4),(u_1,v_1)\}$ derived from matching the ordered edges $(u_0,u_1) \in G_k$ and $(v_4,v_1) \in M$. 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 $(u_2,v_3), (u_2,v_0)$, and $(u_2,v_5)$. A fourth branch representing $(u_2,\emptyset)$ is added to include the possibility of $u_2$ not matched. This traversal continues until all vertices have been visited. As a result, every leaf node reached denotes a possible correspondence of the two graphs formed by collecting vertex correspondences along its path from the root.

Multiple correspondence sets are often generated by the tree search algorithm. A metric is thus needed for comparing them. Let $C$ be a generated correspondence set, and consider a correspondence $(u_i,v_j) \in C$. We denote by $u_i$ the image coordinates of $u_i$. The image coordinates $v_j$ of $v_j$ is obtained as follows. We apply the transformation in the prior state estimate $\mathbf{p}_j$ to the body coordinates of $\mathbf{p}_j$, which corresponds to $v_j$, and then project the result to image coordinates $v_j = \mathbf{p}_j$ via the chain of equations in (23). The mean square error of the set $C$ is then defined as follows:

$$d(C) = \frac{1}{|C|} \sum_{(u_i,v_j) \in C} \|u_i - v_j\|^2.$$  

A correspondence set $C_1$ is larger than another correspondence set $C_2$, denoted $C_1 \supset C_2$, if

$$\left( |C_1| = |C_2| \land d(C_1) < d(C_2) \right) \lor \left( |C_1| > |C_2| \land d(C_1) - d(C_2) < \delta \right),$$  

for some small threshold $\delta > 0$. If neither $C_1 \supset C_2$ or $C_1 \subset C_2$, then $C_1 = C_2$. The 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 new image graph built upon the $k$-th frame generated by the second camera. Update of the hypothesis $H$ is completed for the time instant. The Kalman filter is applied as described in Section IV to generate the posterior estimate $\hat{x}_k^+$. 

B. Evaluating a Hypothesis

After $\hat{x}_k^+$ has been obtained, we measure the quality of the hypothesis $H$ to decide whether it should be passed on to the $(k+1)$-st time instant. For every vertex correspondence $(u_i,v_j)$ in, say, $C(G_k,M)$, we re-evaluate the image coordinates $v_j$ of the vertex $v_j$ on the polyhedron at the pose determined by the posterior state estimate $\hat{x}_k^+$ and under the projection model of the first camera. Then the error $d(C(G_k,M))$ is calculated according to (52). Similarly, the error $d(C(G_k',M))$ is obtained. The error of the hypothesis $H$ is defined as:

$$\epsilon(H) = d(C(G_k,M)) + d(C(G_k',M))$$

When $\epsilon(H)$ is high, we must decide whether to retain $H$ in case the error improves, or reject it otherwise. Images of the

---

1 Many methods exist to track points or edges between frames. Here we compare nearby points by their position relative to the object’s centroid.
flying object are subject to noise that may lead to erroneous edges and vertices appearing. To cope with this and provide some tolerance, the hypothesis $H$ is not rejected only if it has error less than $\Gamma$ at least once in a sliding window of $k$ consecutive images. Proper tuning of $\Gamma$ and $\kappa$ helps the algorithm settle on a small set of low-error hypotheses while subjected to noise that periodically appears in the image.

C. Generating New Hypotheses

In the case that no hypothesis is active, either at the start of tracking ($k = 1$) or after elimination of all hypotheses at the time instant, the algorithm attempts to generate a new set of hypotheses. Multiple sets of vertex correspondences between the image graphs $G_k$ and $G'_k$, and the model graph $M$ are constructed to form new hypotheses, of which a few with the smallest errors are then selected. This is carried out in two steps that are described below.

1) Algorithm Summary: Start with two correspondences, say, $(u_1, v_1)$ and $(u_2, v_2)$, which involve adjacent vertices $u_1$ and $u_2$ in $G_k$ and $v_1$ and $v_2$ in $G'_k$. A maximum correspondence set $C(G_k, G'_k)$ is constructed using the tree search algorithm described in Section V-A. Next, by matching the edge $(u_1, u_2)$ 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 $(u', v) \in C'_i$ whenever $(u, v) \in C_i$. A hypothesis $H_{new} = (C_1, C')$ is thus obtained. To evaluate its error $\epsilon(H_{new})$, we need a state estimate $\hat{x}_k$ of the polyhedron at time instant $k$. The estimate is composed of the object’s position $\mathbf{o}$ and orientation $\mathbf{r}$ initialized under hypothesis $H_{new}$, and velocities $\mathbf{v}$ and $\mathbf{\omega}$ chosen depending on when generation occurs. If $\epsilon(H_{new}) < \Gamma$, the hypothesis is added along with its Kalman filter initialized by $\hat{x}_k$. The above procedure repeats by matching $(u_1, u_2)$ with other edges of $M$ at its start, returning any hypotheses with low errors.

2) Pose Initialization from Stereo Vision: The object’s pose can be initialized based on the depth information provided by the two cameras. Given a newly generated hypothesis, its image-to-model correspondences identify pairs of vertices, $u_i, u'_i$ such that $(u_i, v) \in C(G_k, M)$ and $(u'_i, v) \in C(G'_k, M)$ for some vertex $v$ in $M$. The image coordinates of these vertices that form pairs can be back-projected into world coordinates, effectively mapping two 2D points to one 3D point. The obtained world coordinates can then be used to recover the transformation from the body frame to the world frame. Backprojection and recovery are detailed in Appendix A.

VI. EXPERIMENTS

To demonstrate the accuracy and robustness of the presented estimation scheme, experiments were conducted with four different objects: a plastic cuboid, wooden frame, rugby ball, and foam polyhedron. Various compositions of the three basic modules (aerodynamics, Kalman filtering, and graph-based feature tracking), with increasing complexity, were validated. The first two experiments involving the cuboid and frame objects were conceived to have sources of ground truth available, while the next two involved more practical objects in the rugby ball and polyhedron. 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, and had the objective of demonstrating estimation with non-fiducial features.

Table I summarizes the experiments. Four different estimators were used: an extended Kalman filter (EKF), unscented Kalman filter (UKF) [77], quadratically constrained Kalman filter (QCKF), and the QCKF with aerodynamics modeled (QCKF-A). All the four estimators used the dynamics model (15) of Section II (with the lift force $f_1$ and drag force $f_d$ set to zero by the first three estimators). They also used the projection model of Section III-A (with the epipolar constraint of Section III-B and the quaternion constraints of Section IV incorporated by the last three estimators). In the case of the UKF, the quaternion constraints were enforced using a projection method [37], while the epipolar constraint remained the same. For QCKF-A, lift force was calculated per the procedure in Appendix B. The final experiment involving the polyhedron made use of the graph-based feature tracking algorithm from Section V.

In each experiment, two Ximea MQ022CG-CM high-speed, color cameras in stereo vision configuration were used to capture images simultaneously of an object in flight. Both cameras were equipped with Navitar NMV-6 lenses with an 81.9 degree field of view. The cameras were carefully calibrated with point data obtained by images of a 3D calibration object\(^3\) via bundle adjustment [78] and individually initialized camera parameters [79]. The two camera’s parameters were estimated and poses approximated together via optimization while satisfying the epipolar constraint in equation (28) and minimizing reprojection error. Images were captured at more
than 200 fps depending on the image size in the respective experiment.

Estimation started at time 0 s when an initial estimate was determined, and ran until the object left both cameras’ fields of view. The initial estimate of the object’s pose was computed from a procedure given in Appendix A upon detecting four (or more) pairs of image points from features/marks on the object visible to both cameras. During the first few frames, the object’s initial velocity was estimated roughly by numerical differentiation of position via finite differencing. The initial angular velocity had its estimate set close to zero since an estimate from differentiation would contain a large error. Initial error covariances $P_0$ of the state estimate and $R_0$ of the vector of observables were determined from tuning of the Kalman filter over more than 100 experiment trials.$^4$

Last, we briefly introduce the metric of reprojection error used for validation of pose estimates. The object in its currently estimated pose is projected onto the image plane through the chain of equations in (23). Reprojection error is calculated as the root-mean-squared error (RMSE) between the observed and estimated image coordinates of the object’s vertices. More specifically, in each image it is computed as $\sqrt{\frac{1}{m} \sum_{i=1}^{m} \| u_i - \hat{u}_i \|^2}$, where $u_i$ is the observed image point of the $i$-th visible vertex out of a total of $m$, and $\hat{u}_i$ is the image point obtained by projecting the body coordinates $b_p_i$ corresponding to the $i$-th vertex in the object’s estimated pose.

In Section VI-E we will discuss the adequacy of reprojection errors at measuring pose errors.

A. Spinning Cuboid

In the first experiment, we demonstrated the effectiveness of the Kalman filter and the chosen measurement model at estimating angular velocities. A plastic cuboid was fixed to a stepper motor approximately 1.5 m away from the cameras, and configured to spin at 12 rad/s. The $y$-axis of the body and world frames coincided with the motor’s axis of rotation such that its angular velocity could be measured. The constant speed of rotation served as the ground truth for comparison of angular velocity estimates. The effects of gravitational and aerodynamic forces on the object were negligible and thus omitted from the system dynamics.

The plastic cuboid had a mass of 0.103 kg and inertia tensor $Q = \text{diag}(1.738, 2.812, 1.738) \cdot 10^{-4}$ kg m$^2$. Each face was colored differently, allowing vertex and edge features to be uniquely determined. Image processing was performed to extract these features and provide image points as measurements to the QCKF. A sequence of images of the spinning cuboid was taken at 525 fps$^5$ for 2 s. Fig. 8 shows images of the cuboid at a few time instants along with the estimated poses drawn by yellow lines. In the two leftmost images, the calculated initial pose estimate is reasonably close to the real pose. A few frames later at 0.04 s, the pose estimate lags behind the spinning cuboid due to not knowing the initial angular velocity. The Kalman filter actively corrects its initial estimate, and from approximately 0.1 s on the pose closely matches with the cuboid. Throughout estimation, the obtained reprojection errors were 3.6511 (mean) $\pm$ 1.086 (st. dev.) px for the first camera, and 3.683 $\pm$ 1.051 px for the second camera. Meanwhile, Fig. 9 plots angular velocity estimates over the first 0.4 s when the estimate changed the most. The angular velocity estimate’s $y$-component quickly approached 12 rad/s at approximately 0.1 s, at which point a close alignment of pose is seen in Fig. 8. For the remaining 1.6 s, the estimator maintained a mean error in angular velocity of (-0.048, -0.118, -0.063)$^\top$ rad/s.

B. Wooden Frame with Accelerometers

Next, 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. The object contained four vertices shown in Fig. 10(a), with one at the center as the “origin” of the frame, and the remaining three at the ends of the three links, colored red, green, and blue, respectively, and on their axes of symmetry. Each link, treated as an edge of the object, was easily detected by its color during image processing. The object was estimated by the QCKF, as well as the EKF and UKF for comparison.

Near each vertex of the frame, linear accelerations were measured by four tri-axis ADXL335 accelerometers from Analog Devices, with range $\pm 29.4$ m/s$^2$. The data were transmitted

---

$^4$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 in equation (29), and 0.003 for the three coordinates of the epipolar normals in equation (30).

$^5$Due to the lack of translation, a smaller image size and hence high frame rate could be achieved.
wirelessly using four Digi XBee Series 1 modules. Fig. 10(b) shows the configuration of one axis of the frame. Before the object was thrown, the accelerometers were calibrated by repeatedly measuring voltages corresponding to the force of gravity while laying flat on a table. The object’s inertia tensor was also adjusted to account for the accelerometers as point masses, yielding the value \( Q = \text{diag}(5.57, 5.57, 5.60) \times 10^{-5} \text{ kg \cdot m}^2 \). With a total mass of 0.31 kg, the wooden frame is subjected to negligible effects of aerodynamic forces on the wooden frame.

The object was estimated for 0.61 \( s \) at a rate of 202 fps, and over a distance of 2.13 m within the image. Fig. 11(a)–(b) shows images of the wooden frame in flight. The object’s pose estimates from the EKF and QCKF at a few moments during its flight are drawn in the image. Between images of the two cameras, pairs of poses estimated by the EKF do not appear to align as well with the object. Slight offsets in the pose estimates within each pair are seen in opposing directions from the object. This is primarily due to absence of the epipolar constraint and the unit quaternion constraints, leading to a poor pose estimate that fails to align with image points. On the contrary, the pose estimated by the QCKF aligns better in both images as a result of including the epipolar normals as observables. Consequently, the EKF yielded higher reprojection errors while the UKF (not shown) obtained very similar errors to the QCKF due to both enforcing the constraints.

Finally, we compare the estimated velocities with those measured by accelerometers. First, the object’s initial pose is calculated using carefully extracted image points and the procedure of Appendix A, and its initial velocity \( \dot{v} \) by differenting the object’s position. The \( i \)-th accelerometer produces readings \( \alpha_i \) that satisfy the kinematics equation \( \alpha_i = b\dot{v} + b\omega \times b\mathbf{r}_i + b\omega \times (b\omega \times b\mathbf{r}_i) \), where \( b\mathbf{r}_i \) locates the accelerometer in the body frame. Using four accelerometers, we obtain twelve quadratic equations in total with nine unknowns in \( b\dot{v}, b\omega \), and \( b\omega \). The equations can instead be written as a linear system with twelve (dependent) unknowns: three for \( b\dot{v} \), and nine for a \( 3 \times 3 \) matrix \( W \) composed from \( b\omega \) and \( b\dot{v} \). Formulation of the linear system and extraction of \( b\omega \) from \( W \) is discussed in [18]. 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 \( b\dot{v} \).

Fig. 11(c)–(d) plots the measured and estimated values of velocity and angular velocity. The method described above of integrating accelerations produces the smooth curves plotted by solid lines in lighter colors. Due to obtaining low reprojection errors, the UKF and QCKF were selected for comparison with the velocity data. Velocity estimates of the two filters mostly overlap, while small gaps can be seen among \( wv_x \) and \( wv_y \). The final estimation errors of the QCKF at time 0.62 \( s \) were \((-0.056, -0.153, -0.091)\) \( \text{m/s} \) for velocity, and \((-0.004, 0.032, -0.026)\) \( \text{rad/s} \) for angular velocity. Final estimation errors of the UKF were \((-0.185, -0.284, -0.107)\) \( \text{m/s} \) for velocity, and \((-0.023, -0.008, -0.009)\) \( \text{rad/s} \) 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 \( b\omega_y \) and \( b\omega_z \) up to time 0.23 \( s \). This is due to the \( x \)-axis not being detected as shown in Fig. 11, effectively reducing the information available to estimate rotation about the two
estimates are obtained at time 0.12 s produce the green curves for QCKF-A, and the red curve for QCKF. Both estimates are propagated including aerodynamics to time 0.58 s near the end of estimation. The two resulting position trajectories are projected onto the image plane of the first camera for comparison with the estimated positions drawn by blue dots. A sequence of rugby ball snapshots are superposed on the image plane. The object is spinning primarily about its major axis, drawn by a thin white line. The root-mean-squared errors (RMSE) are plotted from the (a) first and (b) second cameras by two polylines, one in blue calculated from estimates of the QCKF, and one black from estimates of the QCKF-A.

adjacent orthogonal axes. Once the \( x \)-axis becomes visible again, estimation errors in \( b_1 \omega_y \) and \( b_2 \omega_z \) quickly improve.

C. Rugby Ball

In the third experiment, a rugby ball was thrown and estimation took place with aerodynamics. The hollow ball was primarily composed of rubber with a mass of 0.42 kg, and took on the shape of a prolate spheroid whose semi-axes have 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 the intersections with its two minor axes. The ball was thrown with a spin about its major axis (i.e., a “spin pass”), such that the 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 an estimated velocity \( \mathbf{v} = (-2.148, 1.991, -3.547)^T \) m/s and angular velocity \( \omega = (18.280, -0.1576, 1.772)^T \) rad/s.

Now we see how the modeling of aerodynamics improves prediction of the object’s trajectory. Starting at estimates of the two estimators, Fig. 12(a) plots trajectories of position to illustrate improvements in prediction error. The starting estimates are obtained at time 0.12 s and forward propagated to time 0.58 s near the end of flight. To evaluate the estimates equally, both trajectories are produced with inclusion of aerodynamics, and then compared to the “true” trajectory from estimation over the remainder of the flight. Hence, errors from the comparisons correspond to the accuracies of the starting estimates. The green trajectory starting from the estimate of the QCKF-A is seen to yield a much closer prediction. In particular, the position trajectory of the QCKF falls faster than that of the QCKF-A due to the absence of an upward lift force. Since the ball was thrown with a back-spin, lift from rotation (i.e., Magnus effect) produced an upward lift force that prolonged the ball’s flight. Fig. 13 shows the estimated trajectory of the ellipsoid with the estimated velocities and resulting lift forces drawn by arrows.

In addition, prediction errors of the object’s pose 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 errors about the \( z \)-axis were slightly larger due to the larger rate of rotation. Inclusion of aerodynamics also yielded slightly lower re-projection errors, where at the start of estimation the estimates were roughly the same. As more image frames were acquired, integration of dynamics propagated errors to the object’s pose, resulting in an improvement of approximately 2 px for the QCKF-A by the end of flight.

D. Irregular Polyhedron

In the final experiment, a hollow, polyhedral foam object was thrown by hand to demonstrate motion estimation alongside feature tracking when fiducial markers are not present on the object. The polyhedron was constructed by combining eight triangular facets cut from foam boards\(^6\). It had a mass of 0.069 kg and inertia tensor of \( Q = \text{diag}(3.20, 1.96, 2.45) \times 10^{-4} \) kg · m\(^2\). Due to its large size and low weight, the polyhedron was subject to the effects of aerodynamics, to the extent that air flew over its faces produced noticeable lift force when thrown. Overall, the aerodynamics of the

\(^6\)The polyhedron has the following vertices in the frame \( F_b \) aligned with its principal axes: \( (0.051, -0.008, 0.121)^T, (-0.04, -0.055, 0.101)^T, (0.073, 0.103, -0.068)^T, (-0.071, 0.156, -0.002)^T, (-0.044, -0.095, -0.102)^T, (0.031, -0.101, -0.05)^T \)
To demonstrate how the graph-based tracking algorithm performs over a sequence of images of the polyhedron, Fig. 14(c) plots a timeline of hypothesis errors for the object’s flight. The algorithm initially generates the 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. At approximately 0.35 s, errors again increase due to noise that occurs as a face of the object disappears. This moment is seen in Fig. 14(a) and (b), at which image processing fails in subsequent iterations to distinguish between the vertices labeled v₁ and v₂. A few iterations of the tracking algorithm pass before the two hypotheses are rejected and hypothesis generation is triggered, yielding five new hypotheses labeled ③–⑦ at 0.4 s. Of these, the hypothesis labeled ③ is tracked with low error as it appears to correctly identify the objects pose, while ⑥ and ⑦ are rejected, and ④ and ⑤ remain within the error threshold. The object’s state estimate at each frame is chosen as that of the hypothesis with the lowest error.

It is inevitable that image noise and imperfect image 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 transition of the state estimate to the new set of hypotheses, minimal progress in estimation is lost during the time that hypothesis generation takes place.

E. Experiment Summary

The experiments described in this section were conducted on a Windows PC equipped with a 4-core 3.5 GHz Intel Xeon CPU and 8 GB of memory. Table II presents computation times for the four 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 [80], [81] of integration were performed, requiring eight total evaluations of the dynamics equation, and in the case of QCKF-A, each time recalculating the aerodynamic forces. For both the rugby ball and polyhedron, non-uniform grids consisting of roughly 3,500 grid points were used, yielding a linear system of approximately 3,500 rows and 5,000 columns to be solved. It is for this reason that the QCKF-A is most time-consuming.

Meanwhile, the UKF as expected required more computation time than the EKF due to updating and correcting multiple sigma points. Although the UKF is known to better approximate some nonlinear models, it had no advantage in the estimation of the wooden frame. Moreover, were the UKF to consider aerodynamics, repeatedly calculating lift and drag forces for each point would prove the estimator infeasible for real-time motion estimation.

---

polyhedron were more complex than that of the rugby ball from the previous experiment due to geometric asymmetry and singularities at its edges and vertices.

First, detected corner and edge features on the object were used to produce an image graph described in Section V. Two image graphs, one for each camera, were used by the feature tracking algorithm to track visible vertices and edges between consecutive image frames, and update the set of hypotheses of the object’s pose.

The polyhedron was tracked and 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. 14(a)–(b) shows images of the polyhedron during its flight. The object’s pose estimates at a few time instants are drawn in the image, with only those edges visible to each camera shown. In a few of the images, some visible edges are clearly missing after not being detected due to poor lighting of faces of the polyhedron. For instance, at time 0.1 s both cameras were unable to distinguish the edge between adjacent triangular faces. Regardless, the feature tracking and estimation algorithms were able to keep up with the object.

Further iterations of tracking as the object’s pose changes would reveal higher errors in the these hypotheses.
Next, we discuss image-based validation of pose estimates. Were data sets of real images and corresponding motion data for a free-flying object to be available, estimation errors could be easily computed. To the best of our knowledge, no such data sets have been made available for research use. We rely on the measurement of pose error in both images (i.e. reprojection error) to represent the 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. For a single camera, the lack of depth information prevents the pose from being uniquely determined, causing a continuum of poses to produce the same view. More specifically, from the unknown world coordinates of the object’s vertices, the chain of equations in (23) contribute constraints that yield an under-determined system with infinite solutions. This is resolved by the addition of a second camera producing a different but overlapping view of the object relative to the first camera. Then, two simultaneously taken images induce six independent constraints that can only be satisfied by a finite number of poses. The remaining constraints, including the epipolar constraint discussed in Section III, further reduce the possible poses to almost always a unique one. This reasoning is strengthened by considering not a single pose but a sequence of poses estimated from simultaneously taken images of the flying object. The probability of two different initial states of the object producing the same sequence of poses is negligible, if not zero. Generally, a unique correspondence exists between a sequence of poses in the world and a sequence of pairs of poses in two images. As a result of this uniqueness, we regard the aggregation of reprojection errors from the two simultaneously taken images to be sufficient for measuring 3D pose error.

VII. DISCUSSION

This work presents a scheme for motion estimation under conditions that have previously rendered acquisition of accurate pose and velocity estimates infeasible. The object is during a fast free flight, susceptible to large aerodynamic forces (if with low mass density), and observed for only a fraction of a second. Experiments have demonstrated reduced estimation errors with improvements over the baseline EKF approach. Thorough nonlinear models of the object’s dynamics and vision-based observables are presented for use in robust Kalman filtering. Lift force is calculated using the Laplace equation, and drag force from closed-form approximation, yielding improved trajectory estimates for light objects. Estimation error is further reduced by enforcing multiple nonlinear state constraints on the state in an EKF at minimal computational cost. Finally, planar graph matching effectively tracks features of polyhedron with no fiducial markers, while dynamic updating of hypotheses robustly deals with large image noise. These contributions, though combined into a motion estimation solution, can also be employed independently to improve accuracy, enforce multiple constraints, or deal with noise.

While accuracy of pose and motion estimates has been the main focus of our approach, computational cost is a major factor for real time estimation. Table II presented computation times in which the QCKF algorithm obtained a processing rate of 110 Hz. Adding in the repeated calculations of aerodynamic forces reduced the processing rate to below 10 Hz. A decrease in computation time by at least one order of magnitude would be necessary to enable high speed capabilities. Use of modern computing hardware, tuning of parameters, and utilization of GPU processing could help achieve this. Calculation of lift force remains a computational burden. Orders of magnitude improvements could be achieved by use of GPU-enabled linear algebra libraries and tuning of the number of points in the non-uniform grid. The number of hypotheses tracked by the feature tracking algorithm, as well as new ones generated via the exhaustive search, should be kept low to reduce the number of active estimators. Furthermore, optimizations of the QCKF can be made to reduce the number of matrix calculations, including reduction of quaternion noise to a 3-vector, and exclusion of velocities from equations during correction for observables [32]. Such improvements would help make the approach feasible for high speed motion estimation.

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. These include objects with a single curved surface or with a surface consisting of 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 for its originally intended purpose of targeted batting in 3D, an extension of the authors’ work in [10]. Third, robotic catching would also 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]. The methods presented here can provide accurate estimates of any visible trajectory, yielding low prediction error in a variety of scenarios, and a higher overall success rate.

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, even in the case that it is moving directly towards the robot\(^8\). The cameras may be mounted on other mobile robots or vehicles for estimation where additional parameters may need to be

\(^8\)with a possible loss in accuracy due to errors in measurement of depth
estimated for localization\(^9\), 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**

Here we estimate the object’s initial pose, composed of position \(\mathbf{o}^{(0)}\) and rotation \(\mathbf{r}^{(0)}\) at some time instant \(t_0\). The estimates are used to initialize a Kalman filter in Section IV and generate hypotheses for feature tracking in Section V.

Stereo cameras viewing a point in the world induce the epipolar geometry described in Section III-B, which allows for a pair of 2D image points to be back-projected to a 3D point in the world. Given such a pair of image points \(\mathbf{p}'\) and \(\mathbf{q}'\) generated by the first and second camera, respectively, the chain of equations in (23) are solved in reverse order to obtain the corresponding world coordinates \(\mathbf{w}'\). In particular, the second step involves undistortion of \(\mathbf{p}'\) and \(\mathbf{q}'\) by following the procedure at the end of Section III-B, yielding \(\mathbf{r}'\mathbf{P}\) and \(\mathbf{r}'\mathbf{Q}\). Then from equation (18) we are tasked with recovering the point’s third coordinate. This is achieved by performing “rectification” of the points \(\mathbf{r}'\mathbf{P}\) and \(\mathbf{r}'\mathbf{Q}\) to transform their image planes to be coplanar [82, p. 159]. To that effect, the transformed points lie on a horizontal line parallel to the line through the camera centers, allowing for “triangulation” to then determine the point’s \(z\)-coordinate relative to the two parallel cameras [83]. That is, rectification and triangulation yield the coordinate \(\mathbf{c}\), and eventually \(\mathbf{w}'\) after applying the inverse of the transformation in equation (17).

All that remains is to solve the inverse of equation (16) for the object’s initial position \(\mathbf{o}^{(0)}\) and orientation \(\mathbf{r}^{(0)}\). Let \(\mathbf{w}_1, \ldots, \mathbf{w}_n\) be the back-projected points with respect to the world frame \(\mathcal{F}_w\), and \(\mathbf{w}_1, \ldots, \mathbf{w}_n\) the coordinates of vertices of the object with respect to the body frame \(\mathcal{F}_b\). The correspondence between these points is assumed to be known, in our case by the image-to-model correspondence of a hypothesis from Section V.\(^{10}\). The two sets are provided as input to Horn’s method [84] to compute the translation \(\mathbf{r}^{(0)}\) and rotation \(\mathbf{r}^{(0)}\) of the frame \(\mathcal{F}_b\) from \(\mathcal{F}_w\). As long as \(n \geq 3\), the pose is obtained with minimal least-squared error between the world coordinates and the transformed body coordinates.

**APPENDIX B**

**SOLVING THE LAPLACE EQUATION**

The finite differencing method (FDM) is used to compute an approximate solution to the Laplace equation and its boundary conditions. A linear system is constructed from grid points distributed throughout the domain of fluid, that is, relative to a stationary frame instantaneously coinciding with \(\mathcal{F}_b\).

\(^9\)refer to SLAM and VINS works for solutions to the localization problem
\(^{10}\)The image points \(\mathbf{p}_i\) and \(\mathbf{i}'\mathbf{p}_i\) are associated with vertices that, via a hypothesis, correspond with a vertex located at \(\mathbf{q}_i\). Hence, the world coordinate \(\mathbf{w}_i\) computed from the image points shares this correspondence.

Suppose the object’s surface is defined implicitly as \(f(x, y, z) = 0\) in the body frame \(\mathcal{F}_b\). The object is encompassed in fluid flowing at a uniform rate and satisfying the Laplace equation in (6). The fluid is bounded on the interior by the function \(f\), and on the exterior by a 3D box. The domain encompassing the flow is discretized to produce grid points. Since the number of grid points in a volume can quickly exceed computational requirements, finer discretization is used near the surface where higher accuracy is desired. More specifically, grid points near the surface (within some distance threshold of \(f\)) are projected onto it along the normal direction of the closest surface point. Details of this projection are given in Appendix B-C. Fig. 15 shows the non-uniform grid for the two objects used in our third and fourth experiments presented in Sections VI-C and VI-D: an ellipsoidal rugby ball in (a) and polyhedral object in (b). The ellipsoid has the surface function \(f(x, y, z) = x^2/0.06^2 + y^2/0.03^2 + z^2/0.03^2 - 1 = 0\). The surface function of polyhedron is a piecewise one generated below. The distance of a grid point \(\mathbf{p} = (x, y, z)^T\) to the polyhedron is defined by finding the first intersection of the surface with a ray from \(\mathbf{p}\) to the center of mass of the object. The intersection point \(\mathbf{q}\) induces a surface function locally defined as \(f(\mathbf{p}) = \|\mathbf{p} - \mathbf{q}\| = 0\). The function \(f\) becomes piecewise smooth when extended over the polyhedron’s surface.

**A. Generating the Non-uniform Grid**

The Laplace equation (6) and its boundary conditions (7) and (8) are approximated with finite differences at grid points and solved together. Let \(\mathbf{p}\) be a grid point and \(\phi_{i,j,k}\) its velocity potential. Denote the velocity potential of grid points neighboring \(\mathbf{p}\) as \(\phi_{i\pm1,j,k}, \phi_{i,j\pm1,k}, \phi_{i,j,k\pm1}\) in the \(x\), \(y\), and \(z\) directions, respectively. The ‘+’ or ‘−’ in each subscript is chosen based on which direction has a neighboring grid point. We let \(\Delta x^+, \Delta y^+, \Delta z^+\) denote the distance between \(\mathbf{p}\) and its respective neighbor in the positive \(x\), \(y\), and \(z\) direction, and similarly \(\Delta x^−, \Delta y^−, \Delta z^−\) in the negative directions.

The partial derivatives of the velocity potential \(\phi\) are approximated by finite differences derived from the Taylor series expansion of \(\phi\) with respect to \(x\), \(y\), and \(z\) [85, pp. 186–187].
Using forward differences, the boundary condition in (7) with respect to \( x \) is approximated as
\[
\frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{\Delta x^+} = -b_i v_x, \tag{54}
\]
where in a slight abuse of notation, \( \phi_{i,j,k} \) corresponds to grid points only along the exterior boundary. The partial derivative with respect to \( y \) is formed by replacing occurrences of \( \Delta x^+ \) with \( \Delta y^+ \), and \( \phi_{i+1,j,k} \) with \( \phi_{i,j+1,k} \) and partial derivatives with respect to \( z \) follow in the same fashion. Backward and central differences make use of grid points in the negative direction, i.e., \( \Delta x^- \) and \( \phi_{i-1,j,k} \) for partial derivatives with respect to \( x \). Letting \( \hat{n} = (\hat{n}_x, \hat{n}_y, \hat{n}_z) \), the boundary condition of (8) is approximated similarly as
\[
\frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{\Delta x^+} \hat{n}_x + \frac{\phi_{i,j+1,k} - \phi_{i,j,k}}{\Delta y^+} \hat{n}_y + \frac{\phi_{i,j,k+1} - \phi_{i,j,k}}{\Delta z^+} \hat{n}_z = 0, \tag{55}
\]
where \( \phi_{i,j,k} \) corresponds to grid points at the interior boundary.

Last, the Laplace equation (6) expands into
\[
\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} = 0. \tag{56}
\]
From the second order Taylor series approximations of \( \phi \), the central finite difference is written for non-uniform grid points as
\[
\frac{\partial^2 \phi}{\partial x^2} = \frac{2 \Delta x^- (\phi_{i+1,j,k} - \phi_{i,j,k}) + \Delta x^+ (\phi_{i-1,j,k} - \phi_{i,j,k})}{\Delta x^- \Delta x^+ (\Delta x^+ + \Delta x^-)}
\]
At the boundaries, forward and backward differences are used under the assumption of uniformly spaced points (i.e. \( \Delta x^- = \Delta x^+ = \Delta x \))\(^{11}\). Partial derivatives with respect to \( y \) and \( z \) follow similarly. Substitution of the appropriate finite differences into equation (56) yields a linear equation in velocity potentials.

With the Laplace equation and its two boundary conditions approximated, we gather the velocity potentials at all \( n_L \) grid points into a vector \( \phi \). Equations (54)-(56) are subsequently combined to form a linear system in \( \phi \):
\[
C \phi = b, \tag{57}
\]
where the coefficient matrix \( C \) has dimensions \( m_L \times n_L \). We have \( m_L > n_L \) since the Laplace equation over all grid points contributes \( n_L \) rows to \( C \), and the boundary conditions contribute \( m_L - n_L \) rows from points lying on the boundaries. The system (57) can be solved via singular value decomposition. Since the matrices \( C \) and \( b \) are sparse, factorization methods such as sparse LU decomposition can be used.

C. Finite Differences of Surface Points

\(^{11}\)The assumption is met as long as the non-uniform grid is generated with three evenly spaced grid points moving out from each boundary in any direction.

While the system (57) can be employed to solve for the velocity field, we note that the partial derivatives for grid points near the surface could be better approximated. These grid points, previously considered on the surface, are actually a small distance away. Let a point \( p \) near the surface be projected along the direction opposite to the surface gradient \( \nabla f(p) \). That is,
\[
p' = p - l \nabla f(p)^\top = p + (h_x, h_y, h_z)^\top \tag{58}
\]
for some constant \( l > 0 \). Fig. 16 shows the projection of \( p \) onto the surface.

Let \( \phi' \) denote the velocity potential at \( p' \) and \( \hat{n}' \) its surface normal. Taking the first order Taylor series approximation of \( \phi' \) about \( \phi \) gives
\[
\phi' = \phi + h_x \frac{\partial \phi}{\partial x} + h_y \frac{\partial \phi}{\partial y} + h_z \frac{\partial \phi}{\partial z} \tag{59}
\]
The Laplace equation and boundary condition at the surface are then respectively
\[
\nabla^2 \phi' = 0 \quad \text{and} \quad \nabla \phi' \cdot \hat{n}' = 0, \tag{60}
\]
where the first and second order partial derivatives are obtained from differentiating equation (59). The resulting equations yield mixed partial derivatives of \( \phi \) up to the third order. The subsequent first and second order partial derivatives with respect to single variables are obtained from finite differences of the previous section, while the second and third order mixed partial derivatives are approximated below.

Consider the second order partial derivative \( \frac{\partial^2 \phi}{\partial x \partial y} \). Differentiate the first order forward finite difference of \( \phi \) along the \( x \)-direction with respect to \( y \) and plug in two more forward finite differences in the \( y \)-direction to yield
\[
\frac{\partial^2 \phi}{\partial x \partial y} = \frac{\phi_{i+1,j+1,k} - \phi_{i+1,j,k} - \phi_{i,j+1,k} + \phi_{i,j,k}}{h_x h_y}.
\]
For the third order mixed partial derivative \( \frac{\partial^3 \phi}{\partial x^2 \partial y} \), start with the second order central finite difference in the \( x \)-direction and differentiate with respect to \( y \). Substituting in three forward finite differences in the \( y \)-direction yields
\[
\frac{\partial^3 \phi}{\partial x^2 \partial y} = \frac{1}{h_x h_y} \left( \phi_{i+1,j+1,k} - 2 \phi_{i+1,j,k} + \phi_{i,j,k} - \phi_{i-1,j+1,k} - 2 \phi_{i,j,k} + \phi_{i-1,j,k} \right).
\]
Other second and third order partial derivatives are determined by choosing the appropriate forward, backward, or central differences based on available neighboring grid points. Substituting the finite differences into equations (60) yields linear equations in \( \phi \). For points lying close to the interior surface, the equation’s coefficients are written as rows in \( C \), replacing the corresponding rows from (55) and (56) for these points. The resulting system is solved in the same manner as before.