Camera Calibration Using Inaccurate and Asynchronous Discrete GPS Trajectory from Drones

R. YANG
Y. BAR-SHALOM
H. A. J. HUANG

This paper considers a stationary camera calibration problem that estimates the camera orientation angles yaw, pitch, and roll, using a drone trajectory recorded by a GPS. There are three challenges in using a GPS trajectory as ground truth for camera calibration. One, the altitude of GPS data is inaccurate with an unknown bias. Two, the GPS receiver and camera are not time synchronized, and there is an unknown time offset between the two systems. Three, the GPS trajectory is time discrete, and accurate interpolation is needed. This is actually an estimation problem since velocity is also needed. To address the first two challenges, we formulate the problem as a parameter estimation problem to estimate a vector consisting of the GPS altitude bias and time offset in addition to the camera yaw, pitch, and roll biases. We then develop a special maximum-likelihood estimator using the Iterated Least-Squares algorithm, which can work with a nonsynchronized time-discrete GPS trajectory for the third challenge. Since the camera measurement errors are usually small, this requires a high calibration accuracy so that the residual bias error following the calibration should not be significant compared to the measurement error standard deviation. The calibration accuracy depends highly on the drone's trajectory. This paper also recommends an appropriate drone trajectory that can yield a good calibration accuracy, namely, 14% of the measurement error standard deviation. Simulation tests are conducted to demonstrate the algorithm performance. The estimation results meet the Cramer–Rao lower bound (CRLB) since the normalized estimation error squared w.r.t. the CRLB is statistically acceptable.

Manuscript received November 14, 2022; revised January 24, 2023; released for publication May 29, 2023.

Refereeing of this contribution was handled by Florian Meyer.

Y. Bar-Shalom is with the Department of ECE, University of Connecticut, Storrs, CT 06269, USA (e-mail: yaakov.bar-shalom@uconn.edu). R. Yang and H. A. J. Huang are with the DSO National Laboratories, Singapore 118225 (e-mail: yrong@dso.org.sg; hhongan@dso.org.sg).

1557-6418/23/$17.00

I. INTRODUCTION

This paper presents a camera calibration approach for a stationary camera that looks at air targets. We assume the camera is of "pinhole" type without radial and tangential distortion. The position of the camera is assumed to be known. The calibration computes the camera orientation, which is defined by three rotation angles: yaw, pitch, and roll. Since the camera is looking for air targets, there is no fixed object with known position in its field of view (FOV). The calibration is based on the trajectory of a drone instrumented with a GPS receiver, which, however, usually has a significant altitude bias error.[1] Also, the camera and GPS receiver are not time synchronized. This introduces an unknown fixed time offset between the GPS and camera time-stamps. Furthermore, the drone trajectory is a sequence of discrete points with a certain time interval, and there is no analytical expression for the trajectory. This paper will develop a practical approach for the problem of estimating the camera orientation, the GPS altitude bias and the time offset.

Camera calibration is not a new problem. Numerous works have been done before, and they can be categorized into two areas: computer vision-related applications and estimation theory-based approaches. The camera calibration in computer vision is developed from the Perspective-n-Point (PnP) problem [4], [13]. The original PnP problem is described as follows: Given the relative spatial locations of n control points P_i with $i = 1, \ldots, n$, and given the angle to every pair of these points from an additional point, called the center of perspective C, find the lengths of the line segments joining C to each of the control points. The camera calibration is based on the matching of n 3D control points and their corresponding points in the 2D image space. They share the same angles of arrival with reference to the camera center of perspective C. A number of solutions have been developed with this approach [4], [7], [8], [10], [13]. Some focused on the solution of the minimum number of control points required ($n = 3$) as P3P problem [7], [8], [10], [13], and some deal with a large number of points consisting of outliers and inaccurate points. The RANSAC [4] scheme can be applied to select good samples. Some extensions on the camera calibration take unknown focal length and radial distortion into consideration [9], [21].

If we apply the PnP approach to our problem, then a 3D GPS-instrumented drone trajectory needs to match the camera-measured 2D trajectory. Since there is an unknown time offset between GPS-based 3D and camera 2D trajectories and an unknown altitude bias on the 3D trajectory, it is not practical to apply point-to-point 3D–2D matching. We then seek a solution from the estimation theory.

[1] The altitude estimate from GPS is substantially less accurate than the horizontal position since there are fewer high-orbit satellites, which provide most of the altitude information versus low-orbit satellites, which provide most of the horizontal location information.

Unlike the computer vision approach, which develops the camera calibration as a particular geometric problem, the estimation theory approach formulates the camera calibration as a parameter estimation problem with stochastic models. It defines the unknown parameter to be estimated as θ, and builds a relationship between θ and measurements that include noise. If the problem is observable (i.e., with a unique solution), optimization algorithms, such as gradient descent, Newton's algorithm, or Iterated Least Squares (ILS) [1], can be applied in a systematic manner. A number of works along this line have been carried out to estimate the sensor position/orientation and measurement biases. This is often referred to as the sensor registration problem. It can be solved offline using either ILS or maximum-likelihood (ML) estimator from a batch of data [3], [5], [19], [22], [23], [27], or online (estimating the sensor biases and target trajectories simultaneously) using a Kalman filter (KF) type dynamic estimator or Recursive Least-Squares (RLS) approach [2], [17], [24], [25]. The online approaches (also referred to as auto-calibration) sound more attractive. However, they estimate a large augmented state consisting of all target states and sensor biases. This large state may create computational infeasibility for real time when the number of targets is large. Furthermore, sensor bias estimation accuracy is not always guaranteed, as arbitrary target trajectories do not reduce the bias error compared to a dedicated special trajectory. The calibration accuracy (or sensor bias estimation accuracy) is paramount in our problem, as camera orientation must be accurately estimated so that the residual bias error should not be significant compared to the camera measurement error. We therefore prefer an offline approach that allows a GPS-equipped drone to fly in a special predefined path dedicated to accurate camera calibration. Such a path will be discussed in the sequel. The previous work on offline sensor registration mainly dealt with radar pose and measurement biases [5], [19], [22], [23]. Camera calibration was conducted in [3], [27]. The yaw, pitch, and roll biases of multiple cameras and target locations are estimated simultaneously using the ILS method in [3] for a satellite-based camera observing an exoatmospheric target of opportunity. In [27], a camera was calibrated through observing a planar pattern shown at several different orientations, and camera intrinsic and extrinsic parameters were estimated using a closed-form solution. Neither of them deals with unknown time offsets among different systems, for example, sensor and ground truth systems—the different sensors are assumed to be time synchronized.

Online and offline calibration with an unknown time offset have been discussed in various applications [6], [11], [14]–[16], [18], [20]. We focus on the offline solutions [6], [11], [18], [20]. In [11] and [18], the time offset and spatial calibration were conducted separately in sequence. The time offset was estimated first, and then spatial calibration was conducted. A more robust approach [6], [20] estimated the time offset and spatial biases simultaneously. This was a robotics application with camera, inertial measurement unit (IMU) and laser rangefinder. It estimated the time offset among sensors and measurement transformation. However, the camera was assumed well calibrated. The Levenberg–Marquardt (LM) algorithm was used to minimize an objective function based on the ML criterion, using stationary objects detected by the camera and rangefinder on a moving platform. Although the approach included unknown time offsets into its estimation parameter, camera calibration was not conducted.

In this paper, we develop our approach based on estimation theory, which will include the GPS altitude bias and time offset in the estimation of the parameter vector θ. Another challenge is that the GPS 3D trajectory is given in numerical form. The preliminary version of the present study, [26], conducted calibration assuming an accurate GPS without altitude bias. An ILS algorithm was developed to perform calibration based on a stochastic model dealing with a GPS trajectory expressed by a sequence of discrete-time points. In the present paper, inaccurate GPS with unknown altitude bias is used. The calibration accuracy drops significantly with this additional unknown unless it is part of the estimated parameter vector. If the residual bias error (following the calibration) is not small enough compared to the camera measurement error, then the calibration is not meaningful. We will develop an enhanced ILS algorithm to improve the estimation accuracy, and recommend a practical drone path to achieve good calibration accuracy.

The rest of the paper is structured as follows. Section II describes the three coordinate systems used in this paper. Section III describes the problem formulation, namely, the stochastic model for estimation. Section IV presents the estimation algorithm based on the stochastic model dealing with numeric GPS trajectories. Section V presents simulation results on calibration error, and recommends a suitable drone path. Section VI draws the conclusions.

II. COORDINATE SYSTEMS

The following three coordinate systems are used in this paper:

- Common coordinate system with x-y-z as East, North, and Up (ENU).
- Camera coordinate system with x^{C}-y^{C}-z^{C} centered at the camera position $(x^{\mathrm{s}}, y^{\mathrm{s}}, z^{\mathrm{s}})$, shown in Fig. 1.
- Image coordinate system with x^{I}-y^{I} shown in Fig. 1.

The notations used in the paper are listed in Table I. The conversion of $\mathbf{x}$ to $\mathbf{x}^{\mathrm{C}}$ is given by

$$\begin{aligned}\mathbf{x}^{\mathrm{C}} &= \mathbf{T}(\alpha, \epsilon, \rho)(\mathbf{x} - \mathbf{x}^{\mathrm{s}}) \\ &= \mathbf{T}^{z}(\rho)\mathbf{T}^{x}(\epsilon - 90^{\mathrm{o}})\mathbf{T}^{z}(-\alpha)(\mathbf{x} - \mathbf{x}^{\mathrm{s}}),\end{aligned} \tag{1}$$

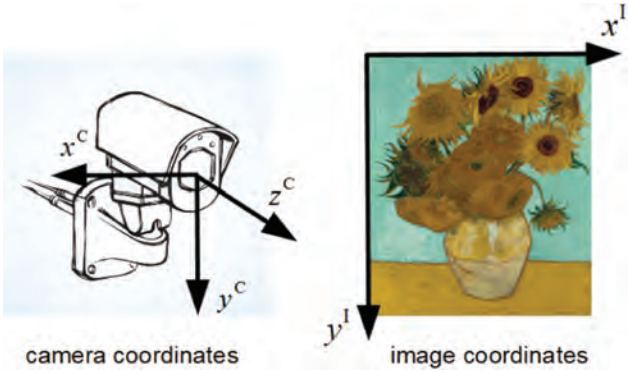

Fig. 1. Camera and image coordinate systems.

where we use the following mnemonic notations for rotations between 3D Cartesian systems:

$$\mathbf{T}^x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}, \tag{2}$$

for a rotation around the x-axis by ϕ from y toward z,

$$\mathbf{T}^z(\phi) = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

for a rotation around the z-axis by ϕ from x toward y. The rotation around the y-axis is not necessary as $\mathbf{T}^x(90^o - \epsilon)$ replaces the y-axis by the z-axis, so that rotation around the z-axis occurs twice. The combined rotation in (1) is

$$\mathbf{T}(\alpha, \epsilon, \rho) = \begin{bmatrix} c_\alpha c_\rho + s_\alpha s_\epsilon s_\rho & c_\alpha s_\epsilon s_\rho - s_\alpha c_\rho & -c_\epsilon s_\rho \\ s_\alpha s_\epsilon c_\rho - c_\alpha s_\rho & s_\alpha s_\rho + c_\alpha s_\epsilon c_\rho & -c_\epsilon c_\rho \\ s_\alpha c_\epsilon & c_\alpha c_\epsilon & s_\epsilon \end{bmatrix}, \tag{4}$$

where

$$s_\alpha = \sin\alpha, \quad s_\epsilon = \sin\epsilon, \quad s_\rho = \sin\rho, \tag{5}$$

$$c_\alpha = \cos\alpha, \quad c_\epsilon = \cos\epsilon, \quad c_\rho = \cos\rho. \tag{6}$$

Table I
Notations

$\mathbf{x}$	$[x\ y\ z]'$, a point in the common (ENU) coordinate system.
$\mathbf{x}^C$	$[x^C\ y^C\ z^C]'$, a point in the camera coordinate system.
$\mathbf{x}^I$	$[x^I\ y^I]'$, a point in the image coordinate system.
$\mathbf{x}^s$	$[x^s\ y^s\ z^s]'$ sensor (camera) position.
α	Camera pointing azimuth or yaw (clockwise from N).
ϵ	Camera pointing elevation or pitch (up from horizontal).
ρ	Camera roll (ideally zero), clockwise around the center of the FPA.
$\hbar$	GPS altitude bias. The GPS-provided altitude is higher than the true value when $\hbar$ is positive; otherwise, $\hbar$ is negative.
τ	Time offset between the drone GPS and the camera. The GPS clock is ahead of the camera clock when τ is positive; otherwise, τ is negative.

The conversion of $\mathbf{x}^C$ to $\mathbf{x}^I$ is

$$\mathbf{x}^I = \mathbf{f}(\mathbf{x}^C) = \begin{bmatrix} \frac{P_x}{2} + \frac{x^C f}{z^C} \\ \\ \frac{P_y}{2} + \frac{y^C f}{z^C} \end{bmatrix}, \tag{7}$$

where f is the focal length with units of pixel (assumed square)

$$f = \frac{P_x}{2\tan(\Theta_x/2)} = \frac{P_y}{2\tan(\Theta_y/2)}, \tag{8}$$

and P_x and P_y are the numbers of pixels in x^I and y^I coordinates, respectively; Θ_x and Θ_y are the FOV—angular spans—in x^I and y^I, respectively.

III. PROBLEM FORMULATION

This section formulates the estimation problem in a stochastic model. The parameter to estimate is

$$\theta = [\alpha\ \epsilon\ \rho\ \hbar\ \tau]', \tag{9}$$

which consists of three camera orientation angles α, ϵ and ρ, GPS altitude bias $\hbar$, and the time offset between the drone GPS and camera systems τ, which are estimated simultaneously. The stochastic model for estimating θ is

$$\mathbf{Z} = \mathbf{H}(\theta, \mathbf{X}) + \mathbf{w}, \tag{10}$$

where $\mathbf{H}(\cdot)$ is defined in (17), $\mathbf{Z}$ is the camera measurement vector consisting of n discrete-time points in the image coordinates as

$$\begin{aligned} \mathbf{Z} &= [\mathbf{z}(t_1)'\ \ldots\ \mathbf{z}(t_n)']' \\ &= [\mathbf{x}^I(t_1)'\ \ldots\ \mathbf{x}^I(t_n)']' + \mathbf{w}, \end{aligned} \tag{11}$$

with measurement times $t_1, \ldots, t_n$, $\mathbf{w}$ is a $2n$ zero-mean Gaussian measurement noise vector with covariance

$$\mathbf{R} = \mathbf{I}_{2n\times 2n}\sigma_F^2, \tag{12}$$

and σ_F^2 is the variance of the measurement noise in the focal-plane array (FPA). For details of how this is obtained based on the optics' point spread function (PSF) and pixel size, see [12]. $\mathbf{X}$ is the GPS drone trajectory (with unknown altitude bias and time offset) represented by a set of discrete-time points in the common coordinate system (ENU) at times corresponding to the camera measurement times, corrected by the (unknown) time offset. It is defined as

$$\mathbf{X} = [\mathbf{x}(t_1 + \tau)'\ \ldots\ \mathbf{x}(t_n + \tau)']'. \tag{13}$$

However, $\mathbf{X}$ is not known exactly. The available information on the GPS trajectory is

$$\overline{\mathbf{X}} = [\mathbf{x}(\bar{t}_1)'\ \ldots\ \mathbf{x}(\bar{t}_m)']', \tag{14}$$

where $\bar{t}_1, \ldots, \bar{t}_m$ do not correspond to the times in $\mathbf{X}$, and $\overline{\mathbf{X}}$ and $\mathbf{X}$ intervals can differ. We need to find the relationship between $\overline{\mathbf{X}}$ and $\mathbf{X}$, so that the model in (10) can be utilized for estimation. This will be solved in the next section.

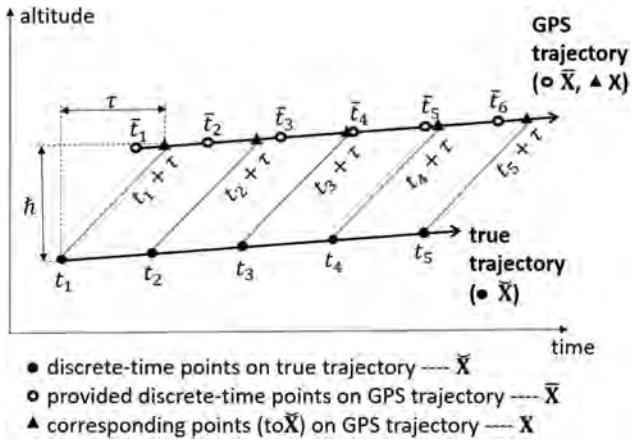

Fig. 2. True and GPS trajectories.

Figure 2 shows the relationship of the true trajectory $\breve{\mathbf{X}}$ and GPS trajectory of the drone, where

$$\breve{\mathbf{X}} = [\breve{\mathbf{x}}(t_1)' \ \ldots \ \breve{\mathbf{x}}(t_n)']'. \tag{15}$$

Each discrete-time point (•) on the true trajectory has a corresponding point (▲) on the GPS trajectory. The relationship of the ith points of $\breve{\mathbf{X}}$ and $\mathbf{X}$ is

$$\breve{\mathbf{x}}(t_i) = \mathbf{x}(t_i + \tau) - [0\ 0\ \hbar]'. \tag{16}$$

The measurement function $\mathbf{H}$ in (10) is then

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1[\alpha, \epsilon, \rho, \breve{\mathbf{x}}(t_1)] \\ \vdots \\ \mathbf{h}_n[\alpha, \epsilon, \rho, \breve{\mathbf{x}}(t_n)] \end{bmatrix} \tag{17}$$

with

$$\begin{aligned} \mathbf{h}_i(\cdot) &= \mathbf{f}\left\{\mathbf{T}(\alpha, \epsilon, \rho)[\overline{\mathbf{x}}(t_i + \tau) - [0\ 0\ \hbar]' - \mathbf{x}^{\mathrm{s}}]\right\} \\ &= \mathbf{f}(\mathbf{x}_i^{\mathrm{C}}) = \mathbf{x}_i^{\mathrm{I}} \\ & \qquad\qquad i = 1, \ldots, n. \end{aligned} \tag{18}$$

The above converts a position "▲" $\mathbf{x}_i$ to a position "•" $\breve{\mathbf{x}}_i$, then converts to camera coordinates as $\mathbf{x}_i^{\mathrm{C}}$ using (1), and finally converts to the image space as $\mathbf{x}_i^{\mathrm{I}}$ using (7).

IV. ESTIMATION ALGORITHM

This section solves the problem described in Section III using a unique ILS algorithm, which is illustrated in Fig. 3. Its uniqueness lies in the fact that $\mathbf{Z}$ and $\overline{\mathbf{X}}$ are used to estimate $\mathbf{X}$ and $\dot{\mathbf{X}}$, and then $\mathbf{Z}$ and $\mathbf{X}$ are used in the iterative estimation of θ, defined in (9).

Given the camera measurement $\mathbf{Z}$, the GPS trajectory $\overline{\mathbf{X}}$ and the initial value of the parameter $\hat{\theta}_0$, the algorithm finds $\hat{\theta}$ through iteration, indexed by j, based on the nonlinear model given in (10). We will describe the algorithm with the following three steps:

(A) Estimation of $\mathbf{X}_j$ and its velocity $\dot{\mathbf{X}}_j$ from $\overline{\mathbf{X}}$ and $\hat{\theta}_j$ in the jth iteration. $\mathbf{X}_j$ and $\dot{\mathbf{X}}_j$ are needed in the θ estimation in (B);

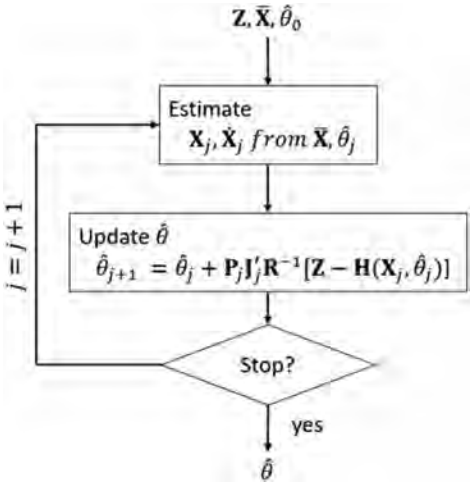

Fig. 3. The ILS estimation algorithm.

(B) Updating of $\hat{\theta}_j$ to $\hat{\theta}_{j+1}$ using an optimization algorithm based on the model (10) with estimated $\mathbf{X}_j$ and $\dot{\mathbf{X}}_j$;

(C) Stop the iteration when a satisfactory $\hat{\theta}$ is obtained.

A. Estimate $\mathbf{X}$ and its Velocities $\dot{\mathbf{X}}$

To estimate θ, we need to know $\mathbf{X}$ and its velocities $\dot{\mathbf{X}}$ from the positions $\overline{\mathbf{X}}$ (namely, to estimate the GPS trajectory "▲" points from "∘" points in Fig. 2), so that the discrete-time points on the GPS trajectory are at times $[t_1 + \tau, \ldots, t_n + \tau]$ corresponding to the camera measurements at times $[t_1, \ldots, t_n]$.[2] The Least-Squares (LS) fitting algorithm developed in [26] used a sliding window containing the neighboring "∘" points before and after a particular "▲" to estimate its position and velocity. However, this will not perform well when a maneuver happens within the window. We therefore enhance it as a two-step LS fitting approach in this paper. Figure 4 illustrates the two steps. We can see the one-step LS approach in (a) has a large error when there is a maneuver. The two-step LS fitting approach shown in (b) uses two LS estimators on the neighboring points before and after the "▲". They obtain two estimates "b" and "a", respectively. The final estimate "c" is a combination of "b" and "a". The estimation error of the two-step LS fitting is therefore reduced significantly.

In the two-step LS fitting approach, we illustrate LS1 (applied to the neighbors before the "▲") to obtain point "b" in detail in the following. The estimation of point "a" is similar. First of all, we estimate the velocities and accelerations of the nearest "∘" [assuming $\mathbf{x}(\bar{t}_i)$] before the "▲". Its velocity and acceleration are

$$\dot{\mathbf{x}}(\bar{t}_i) = [\dot{x}(\bar{t}_i)\ \dot{y}(\bar{t}_i)\ \dot{z}(\bar{t}_i)]', \tag{19}$$

$$\ddot{\mathbf{x}}(\bar{t}_i) = [\ddot{x}(\bar{t}_i)\ \ddot{y}(\bar{t}_i)\ \ddot{z}(\bar{t}_i)]'. \tag{20}$$

[2]This amounts to more than interpolation since the velocities are also estimated, and a special approach is used when the drone maneuvers.

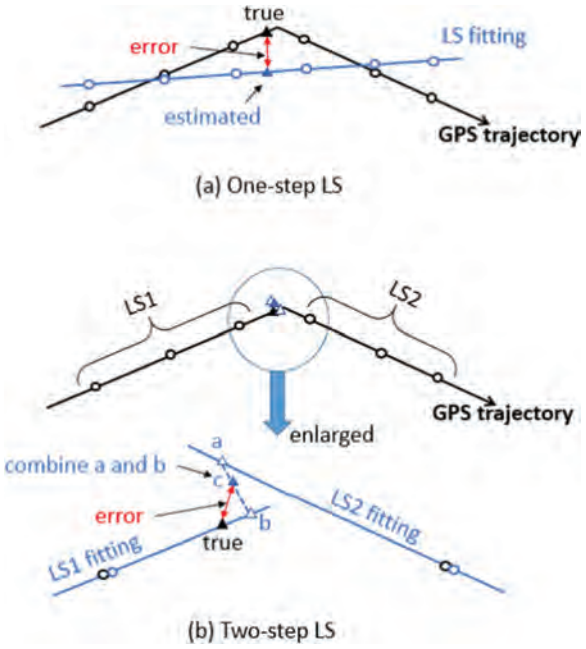

Fig. 4. The LS fitting algorithms. (a) One-step LS fitting approach developed in [26]. (b) Two-step LS fitting approach used in the present paper.

The vectors consisting of velocities and accelerations in x, y, and z coordinates are defined as

$$\mathbf{d}_x^i = [\dot{x}(\bar{t}_i)\ \ddot{x}(\bar{t}_i)]', \tag{21}$$

$$\mathbf{d}_y^i = [\dot{y}(\bar{t}_i)\ \ddot{y}(\bar{t}_i)]', \tag{22}$$

$$\mathbf{d}_z^i = [\dot{z}(\bar{t}_i)\ \ddot{z}(\bar{t}_i)]'. \tag{23}$$

They are estimated separately. The model to estimate $\mathbf{d}_x^i$ from its neighbors is

$$\mathbf{\Delta}_x^i = \mathbf{D}^i \mathbf{d}_x^i, \tag{24}$$

where

$$\mathbf{\Delta}_x^i = \begin{bmatrix} x(\bar{t}_{i-\delta}) - x(\bar{t}_i) \\ \vdots \\ x(\bar{t}_{i-1}) - x(\bar{t}_i) \end{bmatrix}, \tag{25}$$

$$\mathbf{D}^i = \begin{bmatrix} \overline{T}_{i-\delta} & -0.5\overline{T}_{i-\delta}^2 \\ \vdots & \vdots \\ \overline{T}_{i-1} & -0.5\overline{T}_{i-1}^2 \end{bmatrix}, \tag{26}$$

and

$$\overline{T}_{i-k} = \bar{t}_{i-k} - \bar{t}_i \tag{27}$$
$$k = [\delta, \ldots, 1].$$

The number of neighboring points used in (24) is $\delta = 3$. LS is applied to estimate $\mathbf{d}_x^i$ as

$$\hat{\mathbf{d}}_x^i = [(\mathbf{D}^i)'\mathbf{D}^i]^{-1}(\mathbf{D}^i)'\mathbf{\Delta}_x^i, \tag{28}$$

and $\mathbf{d}_y^i$ and $\mathbf{d}_z^i$ are estimated similarly as

$$\hat{\mathbf{d}}_y^i = [(\mathbf{D}^i)'\mathbf{D}^i]^{-1}(\mathbf{D}^i)'\mathbf{\Delta}_y^i, \tag{29}$$

$$\hat{\mathbf{d}}_z^i = [(\mathbf{D}^i)'\mathbf{D}^i]^{-1}(\mathbf{D}^i)'\mathbf{\Delta}_z^i, \tag{30}$$

Next, we compute positions and velocities of "▲", namely, "b" point in Fig. 4(b). We assume the "▲" is the kth point in $\mathbf{X}$. The positions and velocities are computed by

$$\begin{bmatrix} x_b(t_k+\tau_j) \\ \dot{x}_b(t_k+\tau_j) \end{bmatrix} = \begin{bmatrix} 1 & T_k & \frac{T_k^2}{2} \\ 0 & 1 & T_k \end{bmatrix} \begin{bmatrix} x(\bar{t}_i) \\ \dot{x}(\bar{t}_i) \\ \ddot{x}(\bar{t}_i) \end{bmatrix}, \tag{31}$$

$$\begin{bmatrix} y_b(t_k+\tau_j) \\ \dot{y}_b(t_k+\tau_j) \end{bmatrix} = \begin{bmatrix} 1 & T_k & \frac{T_k^2}{2} \\ 0 & 1 & T_k \end{bmatrix} \begin{bmatrix} y(\bar{t}_i) \\ \dot{y}(\bar{t}_i) \\ \ddot{y}(\bar{t}_i) \end{bmatrix}, \tag{32}$$

$$\begin{bmatrix} z_b(t_k+\tau_j) \\ \dot{z}_b(t_k+\tau_j) \end{bmatrix} = \begin{bmatrix} 1 & T_k & \frac{T_k^2}{2} \\ 0 & 1 & T_k \end{bmatrix} \begin{bmatrix} z(\bar{t}_i) \\ \dot{z}(\bar{t}_i) \\ \ddot{z}(\bar{t}_i) \end{bmatrix}, \tag{33}$$

where $T_k = t_k + \tau_j - \bar{t}_i$ and τ_j is from θ_j in the jth iteration. The likelihood of point "b" [see in Fig. 4(b)] is computed using the measurement residual

$$\mathbf{v}_b = [(\mathbf{\Delta}_x^i - \mathbf{D}^i\mathbf{d}_x^i)'\ \ (\mathbf{\Delta}_y^i - \mathbf{D}^i\mathbf{d}_y^i)'\ \ (\mathbf{\Delta}_z^i - \mathbf{D}^i\mathbf{d}_z^i)']', \tag{34}$$

according to

$$\mathcal{L}_b = \mathcal{N}(\mathbf{v}_b; \mathbf{0}, \mathbf{I}), \tag{35}$$

where $\mathcal{N}(\cdot)$ is the standard 3δ-multivariate Gaussian pdf.

The second step LS is computed in a similar manner to obtain the positions, velocities, and likelihood of point "a". The final estimate, for point "c" in Fig. 4(b) is based on a weighted average as follows:

$$\begin{bmatrix} \hat{x}(t_k+\tau_j) \\ \hat{\dot{x}}(t_k+\tau_j) \\ \hat{y}(t_k+\tau_j) \\ \hat{\dot{y}}(t_k+\tau_j) \\ \hat{z}(t_k+\tau_j) \\ \hat{\dot{z}}(t_k+\tau_j) \end{bmatrix} = \frac{\mathcal{L}_a}{\mathcal{L}_a+\mathcal{L}_b} \begin{bmatrix} x_a(t_k+\tau_j) \\ \dot{x}_a(t_k+\tau_j) \\ y_a(t_k+\tau_j) \\ \dot{y}_a(t_k+\tau_j) \\ z_a(t_k+\tau_j) \\ \dot{z}_a(t_k+\tau_j) \end{bmatrix} + \frac{\mathcal{L}_b}{\mathcal{L}_a+\mathcal{L}_b} \begin{bmatrix} x_b(t_k+\tau_j) \\ \dot{x}_b(t_k+\tau_j) \\ y_b(t_k+\tau_j) \\ \dot{y}_b(t_k+\tau_j) \\ z_b(t_k+\tau_j) \\ \dot{z}_b(t_k+\tau_j) \end{bmatrix}. \tag{36}$$

B. Update the Estimate of θ

The parameter given in (9) is estimated based on

$$\hat{\theta} = \arg\min_{\theta} ||\mathbf{Z} - \mathbf{H}(\theta, \mathbf{X})||_{\mathbf{R}^{-1}}^2. \tag{37}$$

Using the ILS [1] to solve the above optimization,[3] one has

$$\hat{\theta}_{j+1} = \hat{\theta}_j + \mathbf{P}_j\mathbf{J}_j'\mathbf{R}^{-1}[\mathbf{Z} - \mathbf{H}(\theta_j, \mathbf{X}_j)], \tag{38}$$

[3]The ILS is the numerical algorithm to solve for the ML estimate under Gaussian assumption.

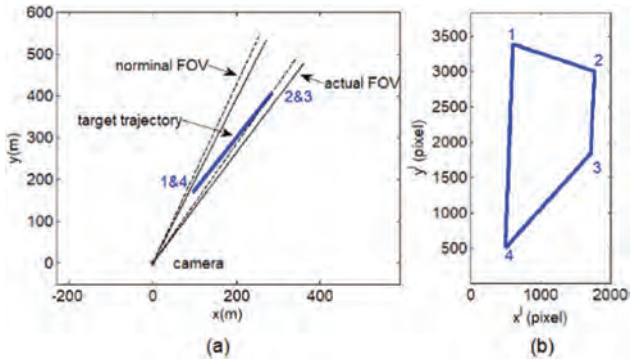

Fig. 5. Test scenario 1. Target moves in constant velocity in a vertical rectangle $(1, 2, 3, 4)$ twice. The higher and lower horizontal edges are at altitudes of 284 m and 84 m, respectively, and the near and far vertical edges are at ranges of 200 and 500 m, respectively. The target speed is 12.5 m/s. (a) Top view in the 3D common coordinates. (b) Trajectory in image coordinates.

$$\mathbf{P}_j = (\mathbf{J}_j' \mathbf{R}^{-1} \mathbf{J}_j)^{-1}, \qquad j = 1, \ldots, n_j \tag{39}$$

with the Jacobian

$$\mathbf{J}_j = [\nabla_{\theta_j} \mathbf{H}(\theta_j, \mathbf{X})']' = [\nabla_{\theta_j} \mathbf{h}_1(\cdot)' \ \ldots \ \nabla_{\theta_j} \mathbf{h}_n(\cdot)']', \tag{40}$$

where j is the iteration index. The final estimate $\hat{\theta}$ is the value to which the iteration (38) converged using a stopping criterion. The derivatives needed for (40) are given in Appendix B.

C. Stopping Criterion

To obtain a good calibration result, we set a tight stopping criterion. First, we normalize the measurement residual squared element by element in iteration j

$$\mathbf{V}_j = [\mathbf{Z} - \mathbf{H}(\mathbf{X}_j, \hat{\theta}_j)] \otimes [\mathbf{Z} - \mathbf{H}(\mathbf{X}_j, \hat{\theta}_j)] \sigma_{\mathrm{F}}^{-2}. \tag{41}$$

Then, we check every element $v_{j,i}$ with $(i = 1 \ldots 2n)$ in $\mathbf{V}_j$, where $2n$ is the number of measurements times the measurement dimension 2. All $v_{j,i}$ must be below the "3 sigma" limit

$$v_{j,i} \leq 3^2. \tag{42}$$

This element-wise checking criterion can prevent a few large measurement residuals being smoothed by a large number of small residuals. Also, to prevent a run that cannot meet the stopping criterion, the maximum number of iterations is set to 20.

V. SIMULATION RESULTS

This section evaluates the performance of the algorithm described in Section IV. We simulate two test scenarios. Scenario 1 shown in Fig. 5 has a drone (quadcopter) moving in a vertical rectangular trajectory 1, 2, 3, 4 with two cycles. Points 1 and 4 are at near range of 200 m with altitudes of 284 m and 98 m, respectively. Points 2 and 3 are at farther range of 500 m with altitudes 284 m and 98 m, respectively. The drone moves

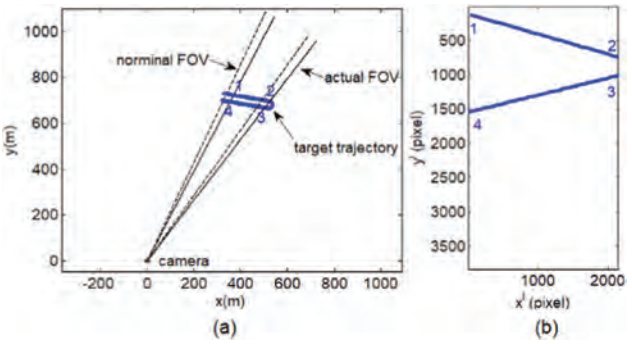

Fig. 6. Test scenario 2. Target moves in constant velocity, makes a 180° turn, and flies back in constant velocity. The target speed is 12.5 m/s. (a) Top view in the 3D common coordinates. (b) Trajectory in image coordinates.

with a nearly constant speed of 12.5 m/s between the four edges. When reaching a corner, it decelerates to 0 m/s, then accelerates to 12.5 m/s on the new direction. The total duration is 109.2 s with 546 measurements. The design principle of the trajectory for this scenario is to span the entire FOV (with near and far motion, i.e., also in depth). Scenario 2 uses the recommended drone path in [26]. This is shown in Fig. 6 with the drone moving with speed of 12.8 m/s at altitude 100 m, and then it makes a 180° turn, and flies back with the same speed and altitude. The total duration is 36 s with 126 measurements. The inaccurate GPS trajectories are discretized with a time interval of 0.1 s. Camera measurements sampling interval is 0.2 s. The camera to calibrate has a FOV of 10° and 17.8° horizontal and vertical, respectively. The nominal orientation angles[4] are set as $\alpha = 30°, \epsilon = 2°$, and $\rho = 0°$. However, their actual values (to be estimated) are $\alpha = 32°, \epsilon = 4.1°$, and $\rho = 2.3°$. The camera provided the measurements only when the target is in its FOV with measurement error standard deviation $\sigma_{\mathrm{F}} =$ 1pixel. The time and altitude offsets are $\tau = 1.35$ s and $\hbar = 10$ m in both scenarios, respectively. We set the time offset precision lower than the GPS discretized precision[5] on purpose to observe the algorithm estimation accuracy better. We will study the estimation accuracy, the statistical efficiency through normalized estimation error squared (NEES) w.r.t. the Cramer–Rao lower bound (CRLB) [2] and the real impact of the results next.

A. Estimation Accuracy

We conducted 100 Monte Carlo runs for each scenario, and recorded the root mean square error (RMSE). The CRLB-based covariance matrix is also computed as a benchmark, and is given by

$$\mathbf{P} = (\mathbf{J}' \mathbf{R}^{-1} \mathbf{J})^{-1} \tag{43}$$

[4]The nominal values are the design values. After system installation, the actual values are usually biased w.r.t. the nominal values.
[5]The smallest significant digit for the offset is 0.01 s, while for the GPS, it is 0.1 s.

Table II
CRLB and RMSE From 100 Runs

Scenario	Parameter	RMSE	σ_{CRLB}
1	α—yaw (mdeg)	0.23	0.21
	ϵ—pitch (mdeg)	0.87	0.82
	ρ—roll (mdeg)	2.90	2.81
	$\hbar$— (mm)	4.69	4.35
	τ (ms)	0.27	0.22
2	α—yaw (mdeg)	1.34	1.05
	ϵ—pitch (mdeg)	36.08	32.21
	ρ—roll (mdeg—yaw)	14.63	14.25
	$\hbar$— (mm)	531	475
	τ (ms)	0.80	0.75
2*	α—yaw (mdeg)	0.45	0.43
	ϵ—pitch (mdeg)	0.47	0.43
	ρ—roll (mdeg)	8.89	8.42
	τ (ms)	0.57	0.50

where $\mathbf{J}$ is computed by (38), but θ used in (38) is the true value, namely,

$$\theta = [32^{\text{o}}\ 4.1^{\text{o}}\ 2.3^{\text{o}}\ 10\text{ m}\ 1.35\text{ s}]'. \tag{44}$$

Note the three angles in θ should be converted to radians as the unit of measurement in both CRLB and ILS computing, as discussed before. The CRLB standard deviations of the estimated parameters are

$$\alpha^{\text{CRLB}} = \sqrt{\mathbf{P}(1,1)}, \tag{45}$$

$$\epsilon^{\text{CRLB}} = \sqrt{\mathbf{P}(2,2)}, \tag{46}$$

$$\rho^{\text{CRLB}} = \sqrt{\mathbf{P}(3,3)}, \tag{47}$$

$$\hbar^{\text{CRLB}} = \sqrt{\mathbf{P}(4,4)}, \tag{48}$$

$$\tau^{\text{CRLB}} = \sqrt{\mathbf{P}(5,5)}. \tag{49}$$

Table II gives the RMSE and CRLB for scenarios 1 and 2. It also lists the results of the scenario 2 (under 2*) obtained in [26], where the same drone path was used, but GPS altitude was assumed perfect without bias. It can be seen that the estimate RMSEs of scenario 1 are close to their CRLBs. The algorithm is statistically efficient in this scenario, as shown in the next subsection. However, the results of scenario 2 are significantly less accurate. The CRLBs are significantly larger than those of scenario 1, especially for ϵ and $\hbar$ with values 32.21 mdeg and 475 mm, respectively. This indicates the observabilities of ϵ and $\hbar$ are marginal in this scenario.

We plot the drone trajectories as seen by the camera and the GPS converted positions in the image space for scenarios 1 and 2 in Figs. 7 and 8, respectively. The parameters used for GPS conversion are set the same as the true values, except for the two marginally observable parameters ϵ and $\hbar$. The true values are $\epsilon = 4.1^{\text{o}}$ and $\hbar = 10$ m, respectively. The values used in the GPS conversion are $\epsilon = 2^{\text{o}}$ and $\hbar = 0$ m, respectively. It

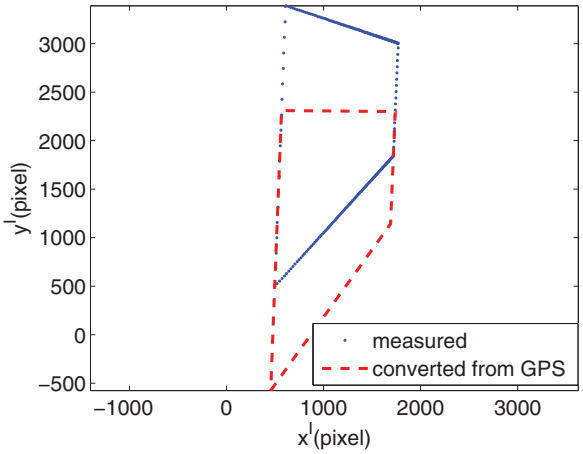

Fig. 7. Measured and GPS converted trajectories of scenario 1, where the actual and nominal yaw, roll, and time difference are set to the same values as $\alpha = 32^{\text{o}}$, $\rho = 2.3^{\text{o}}$, and $\tau = 1.35$ s, respectively. The actual and nominal pitches are $\epsilon = 4.1^{\text{o}}$ and 2^{o}, respectively. The actual and nominal GPS altitude bias are 10 m and 0 m, respectively.

can be seen that the true and the GPS converted trajectories for scenario 1 (Fig. 7) are quite different. This is mainly because the longer vertical edge is at near range 200 m, and the shorter vertical edge is at farther range 500 m. One cannot match them without correct values on both ϵ and $\hbar$. However, the trajectories for scenario 2 (Fig. 8) are almost parallel. Since the two legs on the drone path are at similar range, one can change either GPS altitude or pitch to match the two trajectories. Furthermore, we can also observe that the difference between the RMSE and σ_{CRLB} for ϵ and $\hbar$ are also significantly larger in scenario 2 than those of scenario 1. The algorithm does not perform well when the problem observability is marginal, as in scenario 2, which does not meet the design principle of Scenario 1.

Comparing scenarios 2 and 2*, we can see that including GPS altitude bias (which is generally present)

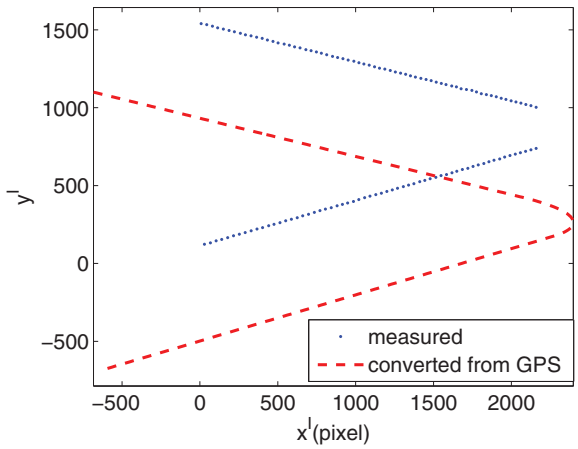

Fig. 8. Measured and GPS converted trajectories of scenario 2, where the actual and nominal yaw, roll, and time difference are set to the same values as $\alpha = 32^{\text{o}}$, $\rho = 2.3^{\text{o}}$, and $\tau = 1.35$ s, respectively. The actual and nominal pitch are $\epsilon = 4.1^{\text{o}}$ and 2^{o}, respectively. The actual and nominal GPS altitude bias are 10 m and 0 m, respectively.

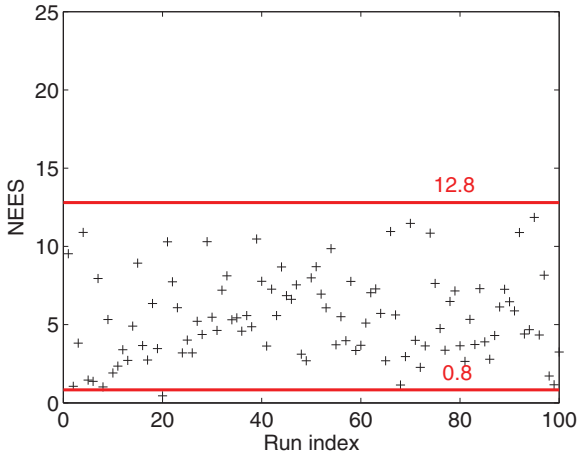

Fig. 9. NEES of 100 runs of the scenario 1.

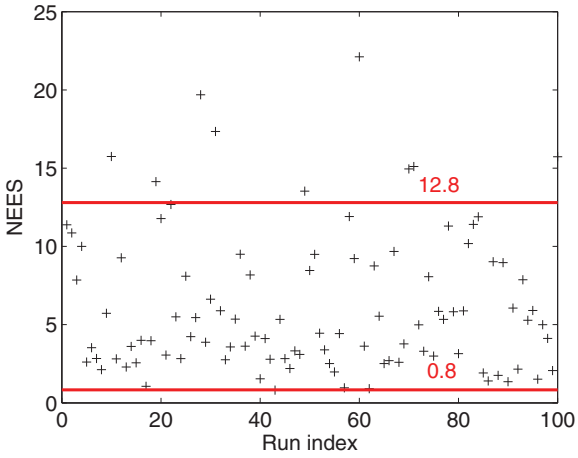

Fig. 10. NEES of 100 runs of the scenario 2.

significantly increases the estimation error using the drone path recommended in [26]. The path is not practical for camera calibration when GPS altitude bias is taken into consideration. The reason is that the trajectory of scenario 2 has poor observerbility when both GPS altitude and camera pitch are unknown.

Another interesting observation is the estimation accuracy of τ is smaller than the GPS time discretization of 100 ms. The best RMSE reaches 0.27 ms in test scenario 1. This shows that the trajectory estimation algorithm described in Section IV overcomes the discretization of the GPS trajectory problem effectively.

B. Statistical Efficiency

The statistical efficiency analysis was conducted using the NEES [2] computed w.r.t. the CRLB, namely,

$$\epsilon^i(t_k) = (\theta - \hat{\theta})' \mathbf{P}^{-1} (\theta - \hat{\theta}) \tag{50}$$

where $\hat{\theta}$ and θ are the parameter estimate and true value, respectively. The NEESs of $N = 100$ runs were recorded and the analysis is carried out for each run, as well as using the average. The NEES of the parameter (with dimension 5) is a 5° of freedom chi-square random variable if the errors are Gaussian. Its two-sided $p = 95\%$ probability region is [0.8, 12.8]. The estimation is statistically efficient, if 95% of NEESs are within this region. Figures 9–10 show the NEES for the two test scenarios, and the number of NEES out of the region [0.8, 12.8] for scenario 1 is 0 (versus the expected value of 5), i.e., the algorithm produced statistically efficient estimates—consistent with equality in the CRLB. However, the number of NEES out of the this interval from 100 runs is scenario 2 is 9. This shows that the estimation algorithm for a marginally observable scenario is marginally statistically efficient. This is because the standard deviation of the number of exceedances of the 95% probability interval is $\sqrt{Np(1-p)} \approx 2$, thus the borderline efficiency.

For the average NEES over 100 runs, the 95% probability region, based on $\chi^2_{500}/100$, is the interval [4.1 5.63]. For scenario 1, the average NEES is 4.69, while for scenario 2, it is 5.86. Thus, the same conclusions can be drawn: for scenario 1, the algorithm is efficient, while for scenario 2, it is borderline.

C. Impact of the Residual Biases

The real impact is further discussed based on the calibration result of scenario 1, which yields a good calibration result. The pixel bias error in the image space caused by the residual calibration errors should be much lower than the measurement error, so that the residual calibration errors are negligible. We compute the pixel bias error based on the calibration RMSE of yaw, pitch, roll, and their combination. The residual bias error impact is obtained from the shifted distances (the unit of measure is pixel) for uniformly distributed 5×5 pixel grid elements covering the whole image space[6] (1–2160 in x^{I}, 1–3840 in y^{I}) when the residual yaw, pitch, and roll errors are introduced. The residual bias error of the kth grid is

$$b_k = |(\breve{x}^{\mathrm{I}}_k, \breve{y}^{\mathrm{I}}_k) - (x^{\mathrm{I}}_k, y^{\mathrm{I}}_k)|, \tag{51}$$

where $(x^{\mathrm{I}}_k, y^{\mathrm{I}}_k)$ is the center of the kth grid element in pixel units and $(\breve{x}^{\mathrm{I}}_k, \breve{y}^{\mathrm{I}}_k)$ is the shifted grid center when the residual calibration errors are added to the nominal yaw, pitch, and roll; b_k is the distance in pixel units between these two grids. We recorded the residual errors b_k of all the grids and plot them in Fig. 11 for three cases. Case (a) has 0.23 mdeg calibration error added to yaw only. Cases (b) and (c) have 0.87 mdeg error added to pitch and 2.9 mdeg error added to roll, respectively. Figure 12 shows the effect of the combination of yaw, pitch, and roll errors. Case (a) increases the yaw, pitch, and roll by 0.23 mdeg, 0.87 mdeg, and 2.9 mdeg, respectively. Case (b) reduces the yaw, pitch, and roll by 0.23 mdeg, 0.87 mdeg, and 2.9 mdeg, respectively. The

[6]The errors for each pixel in such a small grid are practically the same, so there is no point in evaluating the impact of the errors in each pixel separately.

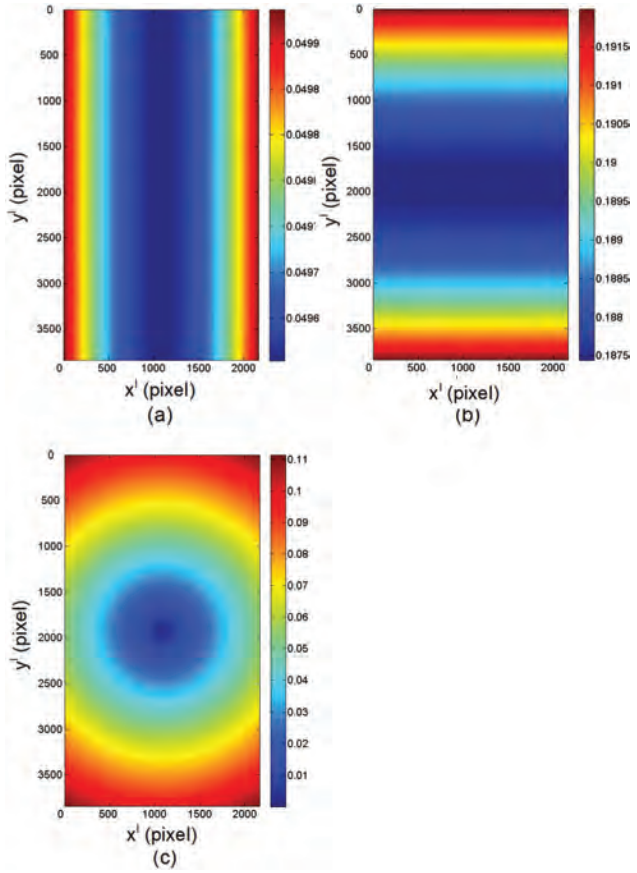

Fig. 11. Biased errors on all 5 × 5 grids. (a) The biased error caused by calibration error on yaw of 0.23 mdeg. (b) The biased error caused by calibration error on pitch of 0.87 mdeg. (c) The biased error caused by calibration error on roll of 2.9 mdeg.

statistics of the grid biases are summarized in Table III. It shows the bias min., max., mean, standard deviation, and root mean square (RMS) value for the five cases in Figs. 11 and 12. From these results, we observe the following:

- The residual bias error is negligible compared to the measurement error. The highest RMSE due to the residual bias is 0.20 pixel. The measurement RMSE in one coordinate (either x^{I} or y^{I}) is 1 pixel. Assuming they are uncorrelated between the coordinates, the total measurement error standard deviation is 1.41 pixel. The highest RMSE due to residual bias is 7.2 times smaller than the measurement RMSE. Thus, the calibration using the scenario 1 drone trajectory achieves negligible bias error.

Table III
Biased Error in Pixel Caused by the Calibration Error

Calibration error (mdeg)			Bias (pixel)				
α	ϵ	ρ	Min.	Max.	Mean	Sthv.	RMS
0.23	0	0	0.050	0.050	0.050	0.000	0.050
0	0.87	0	0.187	0.192	0.189	0.001	0.189
0	0	2.90	0.000	0.110	0.064	0.025	0.059
0.23	0.87	2.90	0.134	0.285	0.210	0.033	0.200
−0.23	−0.87	−2.90	0.134	0.285	0.210	0.033	0.200

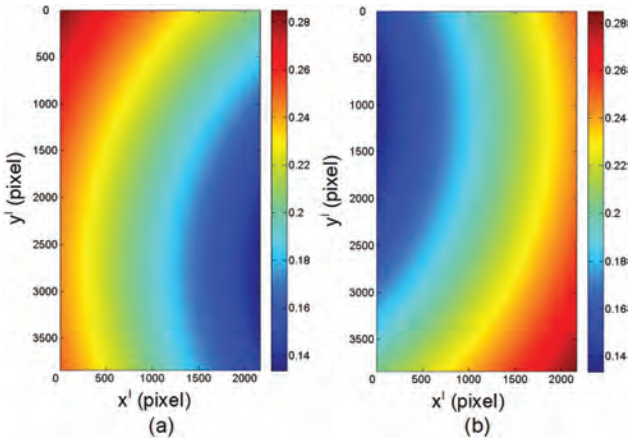

Fig. 12. Biased errors on all 5 × 5 grids. (a) Increases the yaw, pitch, and roll by 0.23 mdeg, 0.87 mdeg, and 2.9 mdeg, respectively. (b) Reduces the yaw, pitch, and roll by 0.23 mdeg, 0.87 mdeg, and 2.9 mdeg, respectively.

- A yaw error creates higher bias on the two vertical edges, and pitch error creates higher bias on the two horizontal edges, as shown in Fig. 11(a) and (b). The differences between the edges and the center are, however, very small.
- A roll error creates higher bias at the four corners, the furthest distance to the center, and the center has zero bias in Fig. 11(c). Nevertheless, the bias at the corners is negligible.
- A combined yaw, pitch, and roll error creates the highest bias at one of the corners from Fig. 12. However, the max. 0.29 pixels is still negligible compared to the measurement RMSE of 1.41 pixel. The max. combined RMSE (measured and bias) is $\sqrt{1.41^2 + 0.29^2} = 1.44$ pixel.

VI. CONCLUSIONS

In this paper, we develop a camera calibration algorithm using drone trajectories recorded by a GPS receiver. However, the recorded GPS data has an unknown altitude bias and an unknown time offset between the GPS and camera systems. The GPS trajectories are discretized with a time interval of 0.1 s. The paper developed a special ML/ILS algorithm dealing with discretized GPS trajectories to estimate camera orientation angles (yaw, pitch, and roll), GPS altitude bias, and time offset simultaneously. The simulation tests were conducted, and an appropriate drone trajectory is recommended whose estimation results met the CRLB and NEES requirements. The time offset estimation error was much smaller than the discretization of the GPS reference trajectory (0.27 ms versus 100 ms). The recommended drone trajectory is suitable for practical use. Its residual calibration bias RMSE was 14% of the measurement error standard deviation, which is negligible.

In our real camera setup and calibration experiments, we realized that more work needs to be done along this research. First, the camera's focal length cannot be fixed beforehand accurately. It needs to be adjusted during setup based on the real situation. Due to a lack of accurate equipment to measure a camera's focal length, it should be an additional camera parameter included in the estimation. Second, the GPS equipment usually has quantization errors in latitude and longitude. This error cannot be ignored when a target is in a near range (with ten-pixel quantization). The ILS algorithm proposed in this paper needs to be further developed to handle these types of errors.

APPENDIX A. THE IMPORTANCE OF BEING EARNEST ABOUT RADIANS

When trigonometric functions are expressed as Taylor expansion, one has to use radians as the unit of measure. This can be illustrated using the following simple example, using the first-order Taylor expansion to compute $\sin(30.01^{\circ})$. The answer should be 0.50015. If we use degrees as the unit of measure, then we will have wrong result as

$$\begin{aligned}\sin(30.01^{\circ}) &= \sin(30^{\circ} + 0.01^{\circ}) \\ &\approx \sin(30^{\circ}) + 0.01^{\circ} \times [\sin(30^{\circ})]' \\ &\approx \sin(30^{\circ}) + 0.01^{\circ} \times \cos(30^{\circ}) \\ &\approx 0.5 + 0.01 \times 0.866 \\ &\approx 0.50866. \end{aligned} \tag{52}$$

If we use radians, then the correct result is

$$\begin{aligned}\sin\left(\frac{30.01 \times \pi}{180}\right) &\approx \sin\left(\frac{30 \times \pi}{180}\right) + \frac{0.01 \times \pi}{180} \cos\left(\frac{30 \times \pi}{180}\right) \\ &\approx 0.5 + 0.00175 \times 0.866 \\ &\approx 0.50015. \end{aligned} \tag{53}$$

Although $\sin(\cdot)$ and $\cos(\cdot)$ should give the same values whether the units are degrees or radians, the small difference 0.01° in front of $\cos(\cdot)$ in (52) leads to wrong result in (53). Thus, angles must be converted to radians when using series expansions.

APPENDIX B. DERIVATIVES FOR (40)

The iteration index j is omitted for simplicity. The gradients needed are

$$[\nabla_{\theta} \mathbf{h}_k(\cdot)']' = \frac{\partial \mathbf{x}_k^{\mathrm{I}}}{\partial \mathbf{x}_k^{\mathrm{C}}} \frac{\partial \mathbf{x}_k^{\mathrm{C}}}{\partial \theta} \qquad k = 1 \ldots n, \tag{54}$$

$$\frac{\partial \mathbf{x}_k^{\mathrm{I}}}{\partial \mathbf{x}_k^{\mathrm{C}}} = \begin{bmatrix} \dfrac{f}{z_k^{\mathrm{C}}} & 0 & -\dfrac{f x_k^{\mathrm{C}}}{(z_k^{\mathrm{C}})^2} \\ 0 & \dfrac{f}{z_k^{\mathrm{C}}} & -\dfrac{f y_k^{\mathrm{C}}}{(z_k^{\mathrm{C}})^2} \end{bmatrix}, \tag{55}$$

$$\frac{\partial \mathbf{x}_k^{\mathrm{C}}}{\partial \theta} = \begin{bmatrix} \dfrac{\partial x_k^{\mathrm{C}}}{\partial \alpha} & \dfrac{\partial x_k^{\mathrm{C}}}{\partial \epsilon} & \dfrac{\partial x_k^{\mathrm{C}}}{\partial \rho} & \dfrac{\partial x_k^{\mathrm{C}}}{\partial \hbar} & \dfrac{\partial x_k^{\mathrm{C}}}{\partial \tau} \\ \dfrac{\partial y_k^{\mathrm{C}}}{\partial \alpha} & \dfrac{\partial y_k^{\mathrm{C}}}{\partial \epsilon} & \dfrac{\partial y_k^{\mathrm{C}}}{\partial \rho} & \dfrac{\partial y_k^{\mathrm{C}}}{\partial \hbar} & \dfrac{\partial y_k^{\mathrm{C}}}{\partial \tau} \\ \dfrac{\partial z_k^{\mathrm{C}}}{\partial \alpha} & \dfrac{\partial z_k^{\mathrm{C}}}{\partial \epsilon} & \dfrac{\partial z_k^{\mathrm{C}}}{\partial \rho} & \dfrac{\partial z_k^{\mathrm{C}}}{\partial \hbar} & \dfrac{\partial z_k^{\mathrm{C}}}{\partial \tau} \end{bmatrix}, \tag{56}$$

and

$$\frac{\partial x_k^{\mathrm{C}}}{\partial \alpha} = \Delta x_k (c_\alpha s_\epsilon s_\rho - s_\alpha c_\rho) - \Delta y_k (s_\alpha s_\epsilon s_\rho + c_\alpha c_\rho), \tag{57}$$

$$\frac{\partial x_k^{\mathrm{C}}}{\partial \epsilon} = \Delta x_k s_\alpha c_\epsilon s_\rho + \Delta y_k c_\alpha c_\epsilon s_\rho + \Delta z_k s_\epsilon c_\rho, \tag{58}$$

$$\begin{aligned}\frac{\partial x_k^{\mathrm{C}}}{\partial \rho} &= \Delta x_k (s_\alpha s_\epsilon c_\rho - c_\alpha s_\rho) \\ &\quad + \Delta y_k (c_\alpha s_\epsilon c_\rho + s_\alpha s_\rho) - \Delta z_k c_\epsilon c_\rho, \end{aligned} \tag{59}$$

$$\frac{\partial x_k^{\mathrm{C}}}{\partial \hbar} = c_\epsilon s_\rho, \tag{60}$$

$$\begin{aligned}\frac{\partial x_k^{\mathrm{C}}}{\partial \tau} &= \hat{\dot{x}}(t_k + \tau)(c_\alpha c_\rho + s_\alpha s_\epsilon s_\rho) \\ &\quad + \hat{\dot{y}}(t_k + \tau)(c_\alpha s_\epsilon s_\rho - s_\alpha c_\rho) - \hat{\dot{z}}(t_k + \tau) c_\epsilon s_\rho, \end{aligned} \tag{61}$$

$$\frac{\partial y_k^{\mathrm{C}}}{\partial \alpha} = \Delta x_k (c_\alpha s_\epsilon c_\rho + s_\alpha s_\rho) + \Delta y_k (c_\alpha s_\rho - s_\alpha s_\epsilon c_\rho), \tag{62}$$

$$\frac{\partial y_k^{\mathrm{C}}}{\partial \epsilon} = \Delta x_k s_\alpha c_\epsilon c_\rho + \Delta y_k c_\alpha c_\epsilon c_\rho + \Delta z_k s_\epsilon c_\rho, \tag{63}$$

$$\begin{aligned}\frac{\partial y_k^{\mathrm{C}}}{\partial \rho} &= -\Delta x_k (s_\alpha s_\epsilon s_\rho + c_\alpha c_\rho) \\ &\quad + \Delta y_k (s_\alpha c_\rho - c_\alpha s_\epsilon s_\rho) + \Delta z_k c_\epsilon s_\rho, \end{aligned} \tag{64}$$

$$\frac{\partial y_k^{\mathrm{C}}}{\partial \hbar} = c_\epsilon c_\rho, \tag{65}$$

$$\begin{aligned}\frac{\partial y_k^{\mathrm{C}}}{\partial \tau} &= \hat{\dot{x}}(t_k + \tau)(s_\alpha s_\epsilon c_\rho - c_\alpha s_\rho) \\ &\quad + \hat{\dot{y}}(t_k + \tau)(s_\alpha s_\rho + c_\alpha s_\epsilon c_\rho) - \hat{\dot{z}}(t_k + \tau) c_\epsilon c_\rho, \end{aligned} \tag{66}$$

$$\frac{\partial z_k^{\mathrm{C}}}{\partial \alpha} = \Delta x_k c_\alpha c_\epsilon - \Delta y_k s_\alpha c_\epsilon, \tag{67}$$

$$\frac{\partial z_k^{\mathrm{C}}}{\partial \epsilon} = -\Delta x_k s_\alpha s_\epsilon - \Delta y_k c_\alpha s_\epsilon + \Delta z_k c_\epsilon, \tag{68}$$

$$\frac{\partial z_k^{\mathrm{C}}}{\partial \rho} = 0, \tag{69}$$

$$\frac{\partial x_k^{\mathrm{C}}}{\partial \hbar} = -s_\epsilon, \tag{70}$$

$$\frac{\partial z_k^{\mathrm{C}}}{\partial \tau} = \hat{\dot{x}}(t_k+\tau)s_\alpha c_\epsilon + \hat{\dot{y}}(t_k+\tau)c_\alpha c_\epsilon + \hat{\dot{z}}(t_k+\tau)s_\epsilon, \tag{71}$$

where

$$\Delta x_k = \hat{x}(t_k+\tau) - x^{\mathrm{s}}, \tag{72}$$

$$\Delta y_k = \hat{y}(t_k+\tau) - y^{\mathrm{s}}, \tag{73}$$

$$\Delta z_k = \hat{z}(t_k+\tau) - \hbar - z^{\mathrm{s}}. \tag{74}$$

The point $[\hat{x}(t_k+\tau), \hat{y}(t_k+\tau), \hat{z}(t_k+\tau)]$ in (72)–(74) on the drone trajectory and its velocity $[\hat{\dot{x}}(t_k+\tau), \hat{\dot{y}}(t_k+\tau), \hat{\dot{z}}(t_k+\tau)]$ in (61), (66), and (71) have been estimated in Section IV-A.

The unit of measure for the three angles α, ϵ, and ρ has to be radians—see Appendix A.

REFERENCES

[1] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan
Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software. New York, NY, USA: Wiley, 2001.

[2] Y. Bar-Shalom, P. K. Willet, and X. Tian
Tracking and Data Fusion: A. Handbook of Algorithms. Storrs, CT, USA: YBS Publishing, 2011.

[3] D. Belfadel, R. W. Osborne, III, and Y. Bar-Shalom
"Bias estimation and observability for optical sensors with targets of opportunity,"
J. Adv. Inf. Fusion, vol. 9, no. 2, pp. 59–74, Dec. 2014.

[4] M. A. Fischler and R. C. Bolles
"Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,"
Commun. ACM, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[5] S. Fortunati, A. Farina, F. Gini, A. Graziano, M. S. Greco, and S. Giompapa
"Least squares estimation and Cramér–Rao type lower bounds for relative sensor registration process,"
IEEE Trans. Signal Process., vol. 59, no. 3, pp. 1075–1087, Mar. 2011.

[6] P. Furgale, J. Rehder, and R. Siegwart
"Unified temporal and spatial calibration for multi-sensor systems,"
in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1280–1286.

[7] R. Haralick, C. Lee, K. Ottenberg, and M. Nolle
"Analysis and solutions of the three point perspective pose estimation problem,"
in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1991, pp. 592–598.

[8] R. Haralick, C.-N. Lee, K. Ottenberg, and M. Nolle
"Review and analysis of solutions of the three point perspective pose estimation problem,"
Int. J. Comput. Vis., vol. 13, no. 3, pp. 331–356, Dec. 1994.

[9] K. Josephson and M. Byrod
"Pose estimation with radial distortion and unknown focal length,"
in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 2419–2426.

[10] S. Linnainmaa, D. Harwood, and L. Davis
"Pose estimation of a three-dimensional object using triangle pairs,"
IEEE Trans. Pattern Anal. Mach. Intell., vol. 10, no. 5, pp. 634–647, Sep. 1988.

[11] J. Kelly and G. S. Sukhatm
"A general framework for temporal calibration of multiple proprioceptive and exteroceptive sensors,"
in *Proc. 12th Int. Symp. Exp. Robot.*, 2010, pp. 195–209.

[12] Q. Lu, Y. Bar-Shalom, and P. Willett
"Measurement extraction for a point target from an optical sensor,"
IEEE Trans. Aerosp. Electron. Syst., vol. 54, no. 6, pp. 2735–2745, Dec. 2018.

[13] L. Kneip, D. Scaramuzza, and R. Siegwart
"A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation,"
in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 2969–2976.

[14] W. Li, H. Leung, and Yifeng Zhou
"Space-time registration of radar and ESM using unscented Kalman filter,"
IEEE Trans. Aerosp. Electron. Syst., vol. 40, no. 3, pp. 824–836, Jul. 2004.

[15] M. Li and A. I. Mourikis
"Online temporal calibration for camera-IMU systems: Theory and algorithms,"
Int. J. Robot. Res., vol. 33, no. 7, pp. 947–964, May 2014.

[16] S. Li, Y. Cheng, D. Brown, R. Tharmarasa, G. Zhou, and T. Kirubarajan
"Comprehensive time-offset estimation for multisensor target tracking,"
IEEE Trans. Aerosp. Electron. Syst., vol. 56, no. 3, pp. 2351–2373, Jun. 2020.

[17] X. D. Lin, Y. Bar-Shalom, and T. Kirubarajan
"Exact multisensor dynamic bias estimation with local tracks,"
IEEE Trans. Aerosp. Electron. Syst., vol. 40, no. 2, pp. 576–590, Apr. 2004.

[18] E. Mair, M. Fleps, M. Suppa, and D. Burschka
"Spatio-temporal initialization for IMU to camera registration,"
in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2011, pp. 557–564.

[19] N. Okello and B. Ristic
"Maximum likelihood registration for multiple dissimilar sensors,"
IEEE Trans. Aerosp. Electron. Syst., vol. 39, no. 3, pp. 1074–1083 Jul. 2003.

[20] J. Rehder, R. Siegwart, and P. Furgale
"A general approach to spatiotemporal calibration in multisensor systems,"
IEEE Trans. Robot., vol. 32, no. 2, pp. 383–398, Apr. 2016.

[21] C. Wu
"P3.5P: Pose estimation with unknown focal length,"
in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2440–2448.

[22] Y. Zhou, H. Leung, and P. C. Yip
"An exact maximum likelihood registration algorithm for data fusion,"
IEEE Trans. Signal Process., vol. 45, no. 6, pp. 1560–1572, Jun. 1997.

[23] Y. Zhou, H. Leung, and M. Blanchette
"Sensor alignment with earth-centered earth-fixed (ECEF) coordinate systems,"
IEEE Trans. Aerosp. Electron. Syst., vol. 35, no. 2, pp. 410–417, Apr. 1999.

[24] E. F. Wilthil and E. F. Brekke
"Compensation of navigation uncertainty for target tracking on a moving platform,"
in *Proc. 19th Int. Conf. Inf. Fusion*, 2016, pp. 1616–1621.

[25] C. Yang, E. Blasch, and P. Douville
"Design of Schmidt–Kalman filter for target tracking with navigation errors,"
in *Proc. IEEE Aerosp. Conf.*, 2010, Mar. 2010,
[26] R. Yang, Y. Bar-Shalom, and H. A. J. Huang
"Camera calibration with unknown time offset between the camera and drone GPS systems,"
in *Proc. 25th Int. Conf. Inf. Fusion*, 2022, pp. 1–8.
[27] Z. Zhang, "A flexible new technique for camera calibration,"
IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

Rong Yang received the B.E. degree in information and control from Xi'an Jiao Tong University, Xi'an, China, in 1986, the M.Sc. degree in electrical engineering from the National University of Singapore, Singapore, in 2000, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 2012. She is currently a Principal Member of Technical Staff at DSO National Laboratories, Singapore. Her research interests include passive tracking, low-observable target tracking, GMTI tracking, hybrid dynamic estimation, and data fusion. She was Publicity and Publication Chair of FUSION 2012 and received the FUSION 2014 Best Paper Award (first runner up).

Yaakov Bar-Shalom (F'84) received the B.S. and M.S. degrees in electrical engineering from the Technion, Haifa, Israel, in 1963 and 1967, respectively, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 1970. He is currently a Board of Trustees Distinguished Professor with the ECE Department and Marianne E. Klewin Professor with the University of Connecticut, Storrs, CT, USA. His current research interests are in estimation theory, target tracking, and data fusion. He has published more than 650 papers and book chapters. He coauthored/edited eight books, including *Tracking and Data Fusion* (YBS Publishing, 2011). He has been elected as a Fellow of IEEE for "contributions to the theory of stochastic systems and of multitarget tracking." He served as an Associate Editor for the IEEE Transactions on Automatic Control and Automatica. He was General Chairman of the 1985 ACC, General Chairman of FUSION 2000, President of ISIF, in 2000 and 2002, and Vice President for Publications from 2004 to 2013. Since 1995, he has been a Distinguished Lecturer of the IEEE AESS. He is a corecipient of the M. Barry Carlton Award for the best paper in the IEEE TAE Systems in 1995 and 2000. In 2002, he received the J. Mignona Data Fusion Award from the DoD JDL Data Fusion Group. He is a member of the Connecticut Academy of Science and Engineering. In 2008, he was awarded the IEEE Dennis J. Picard Medal for Radar Technologies and Applications, and in 2012, the Connecticut Medal of Technology. He has been listed by academic.research.microsoft (top authors in engineering) as #1 among the researchers in aerospace engineering based on the citations of his work. He is the recipient of the 2015 ISIF Award for a Lifetime of Excellence in Information Fusion. This award has been renamed in 2016 as the Yaakov Bar-Shalom Award for a Lifetime of Excellence in Information Fusion. He has the following Wikipedia page: `https://en.wikipedia.org/wiki/Yaakov` Bar-Shalom.

Huang Hong'An Jack was born in Singapore in 1983. He received the B.E. from the National University of Singapore (NUS), Singapore, in 2008. He is currently a Senior Member of Technical Staff at DSO National Laboratories, Singapore. His research interests in target tracking including GMTI tracking, passive tracking, and image tracking. He received the FUSION 2014 Best Paper Award (first runner up).