June 2023 Volume 18 Number 1 ISSN 1557-6418

JAIF 23

Journal of Advances in Information Fusion

A semi-annual archival publication of the International Society of Information Fusion

Regular Papers Page

ISIF

INTERNATIONAL SOCIETY OF INFORMATION FUSION

The International Society of Information Fusion (ISIF) is the premier professional society and global information resource for multidisciplinary approaches for theoretical and applied INFORMATION FUSION technologies. Technical areas of interest include target tracking, detection theory, applications for information fusion methods, image fusion, fusion systems architectures and management issues, classification, learning, data mining, Bayesian and reasoning methods.

Journal of Advances in Information Fusion (ISSN 1557-6418) is published semi-annually by the International Society of Information Fusion. The responsibility for the contents rests upon the authors and not upon ISIF, the Society, or its members. ISIF is a California Nonprofit Public Benefit Corporation at P.O. Box 4631, Mountain View, California 94040.

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

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

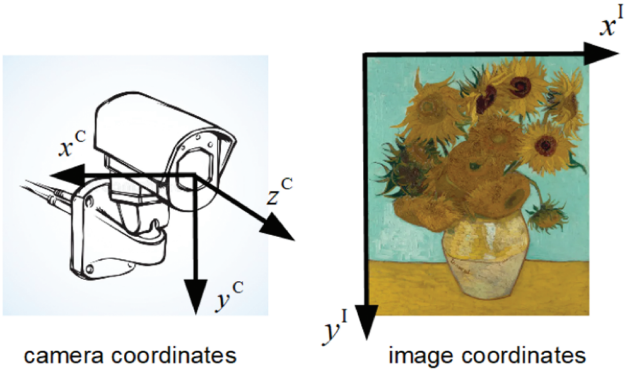


Fig. 1. Camera and image coordinate systems.

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 $\mathbf{T}^x(90^{\mathrm{o}} - \epsilon)$ 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.
$\mathbf{x}^{\mathrm{C}}$	$[x^{\mathrm{C}}\ y^{\mathrm{C}}\ z^{\mathrm{C}}]'$, a point in the camera coordinate system.
$\mathbf{x}^{\mathrm{I}}$	$[x^{\mathrm{I}}\ y^{\mathrm{I}}]'$, a point in the image coordinate system.
$\mathbf{x}^{\mathrm{s}}$	$[x^{\mathrm{s}}\ y^{\mathrm{s}}\ z^{\mathrm{s}}]'$ 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 $\mathbf{x}^{\mathrm{C}}$ to $\mathbf{x}^{\mathrm{I}}$ is

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

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

$$f = \frac{P_{\mathrm{x}}}{2\tan(\Theta_{\mathrm{x}}/2)} = \frac{P_{\mathrm{y}}}{2\tan(\Theta_{\mathrm{y}}/2)}, \tag{8}$$

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

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

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

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

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.

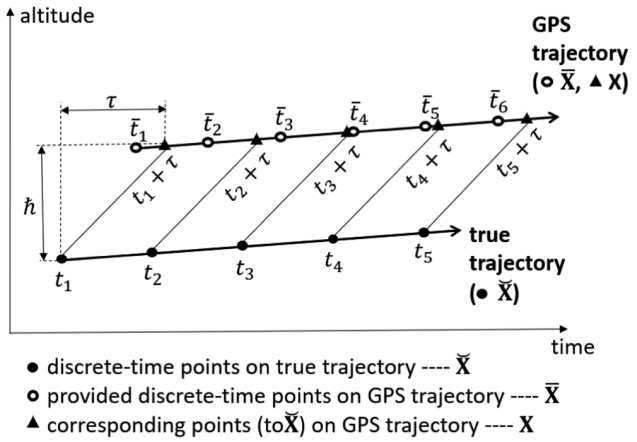


Fig. 2. True and GPS trajectories.

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

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

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 $\bar{\mathbf{X}}$ 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 $\bar{\mathbf{X}}$ 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 $\bar{\mathbf{X}}$ and $\hat{\theta}_j$ in the jth iteration. $\mathbf{X}_j$ and $\dot{\mathbf{X}}_j$ are needed in the θ estimation in (B);

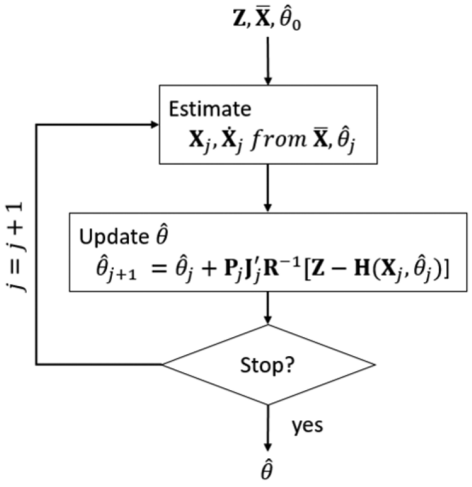


Fig. 3. The ILS estimation algorithm.

(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 $\bar{\mathbf{X}}$ (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.

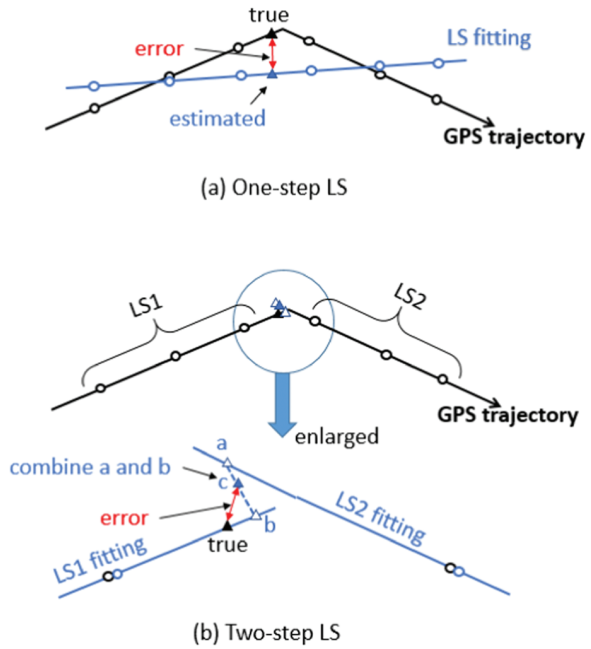


Fig. 4. The LS fitting algorithms. (a) One-step LS fitting approach developed in [26]. (b) Two-step LS fitting approach used in the present paper.

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

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

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.

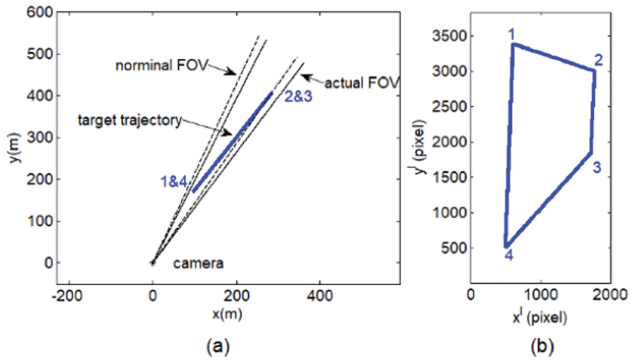


Fig. 5. Test scenario 1. Target moves in constant velocity in a vertical rectangle $(1, 2, 3, 4)$ twice. The higher and lower horizontal edges are at altitudes of 284 m and 84 m, respectively, and the near and far vertical edges are at ranges of 200 and 500 m, respectively. The target speed is 12.5 m/s.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

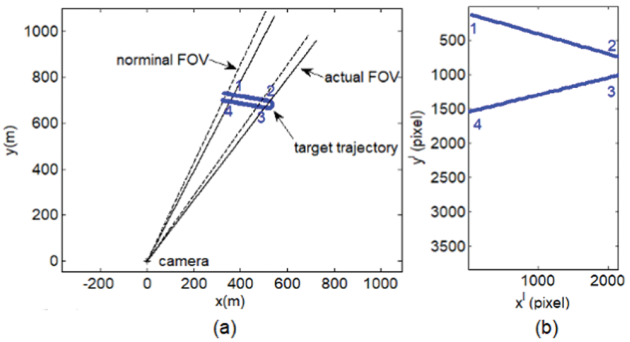


Fig. 6. Test scenario 2. Target moves in constant velocity, makes a 180° turn, and flies back in constant velocity. The target speed is 12.5 m/s. (a) Top view in the 3D common coordinates. (b) Trajectory in image coordinates.

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

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 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

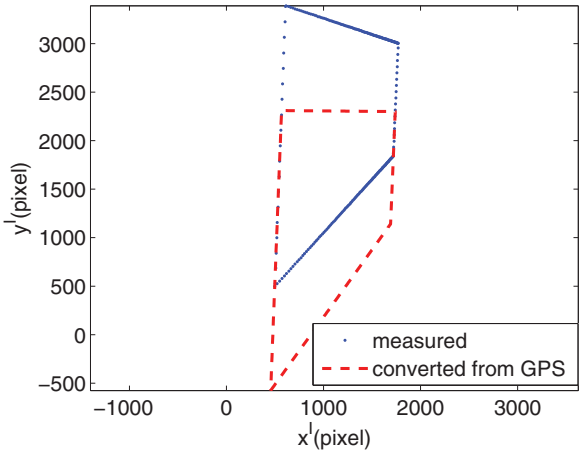


Fig. 7. Measured and GPS converted trajectories of scenario 1, where the actual and nominal yaw, roll, and time difference are set to the same values as $\alpha = 32^{\text{o}}$, $\rho = 2.3^{\text{o}}$, and $\tau = 1.35$ s, respectively. The actual and nominal pitches are $\epsilon = 4.1^{\text{o}}$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

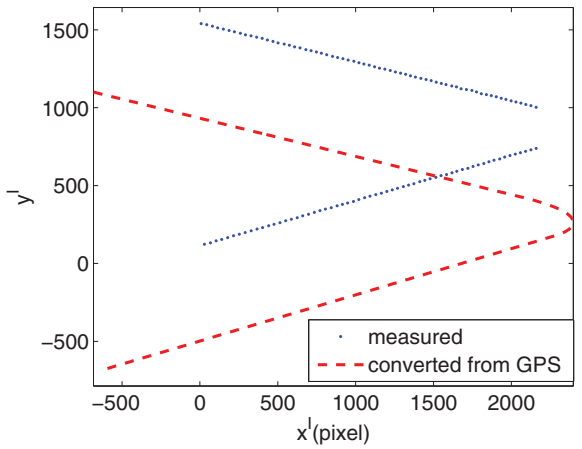


Fig. 8. Measured and GPS converted trajectories of scenario 2, where the actual and nominal yaw, roll, and time difference are set to the same values as $\alpha = 32^{\text{o}}$, $\rho = 2.3^{\text{o}}$, and $\tau = 1.35$ s, respectively. The actual and nominal pitch are $\epsilon = 4.1^{\text{o}}$ and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

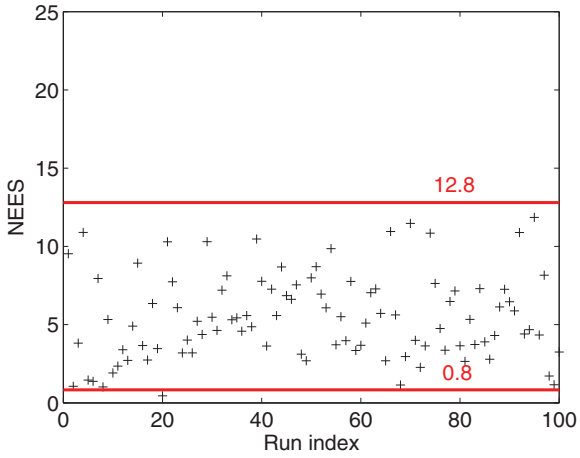


Fig. 9. NEES of 100 runs of the scenario 1.

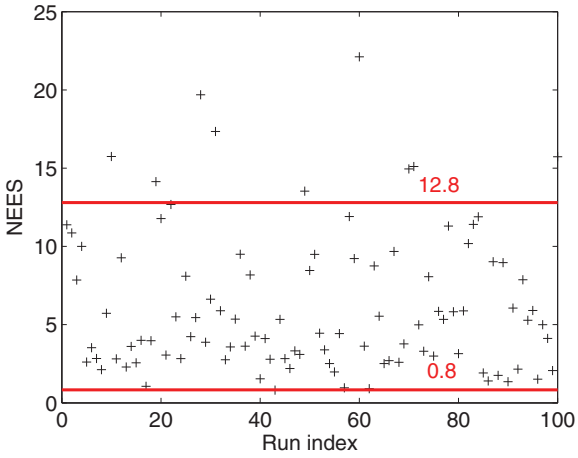


Fig. 10. NEES of 100 runs of the scenario 2.

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.

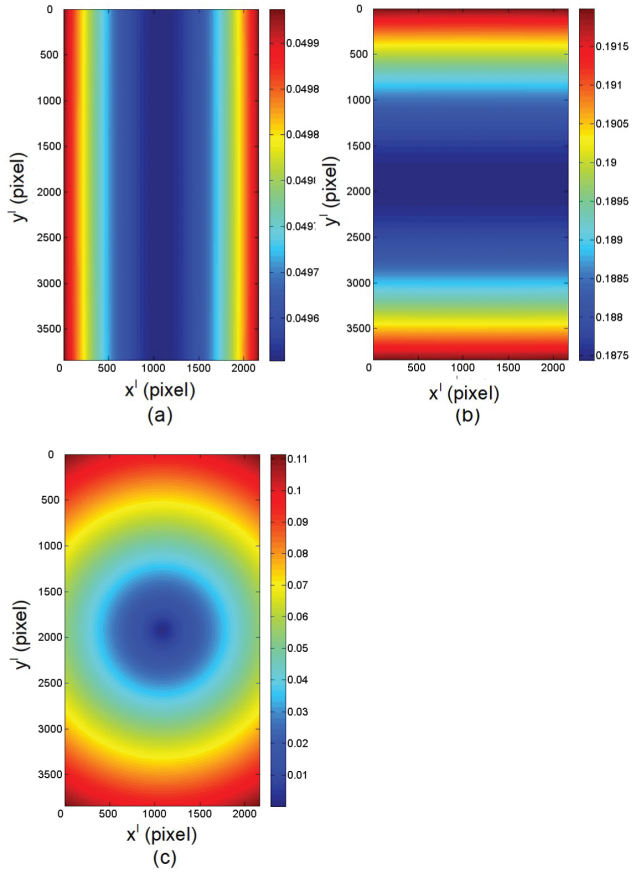


Fig. 11. Biased errors on all 5 × 5 grids. (a) The biased error caused by calibration error on yaw of 0.23 mdeg. (b) The biased error caused by calibration error on pitch of 0.87 mdeg. (c) The biased error caused by calibration error on roll of 2.9 mdeg.

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.
- 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.

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

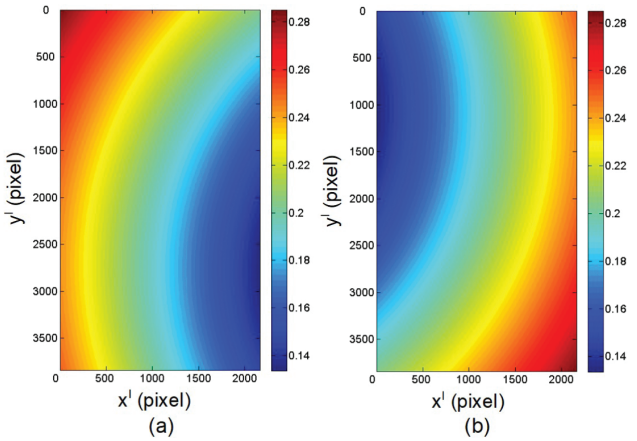


Fig. 12. Biased errors on all 5 × 5 grids. (a) Increases the yaw, pitch, and roll by 0.23 mdeg, 0.87 mdeg, and 2.9 mdeg, respectively. (b) Reduces the yaw, pitch, and roll by 0.23 mdeg, 0.87 mdeg, and 2.9 mdeg, respectively.

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).

The Generalized Fibonacci Grid as Low-Discrepancy Point Set for Optimal Deterministic Gaussian Sampling

DANIEL FRISCH
UWE D. HANEBECK

We propose a multivariate Gaussian sampling scheme. The samples exhibit an "optimal deterministic" configuration. This entails better quadrature or cubature results than with random or quasi-random samples. Our sampling is based on the generalized Fibonacci grid that makes the remarkable properties of the well-known two-dimensional Fibonacci grid applicable in higher dimensions. Two options for generating the multivariate generalized Fibonacci grid are presented, based on a rotated grid and a linear programming counter, respectively. Various options for covariance matching are explored to obtain an unscented transform.

Manuscript received October 7, 2022; revised February 16, 2023; released for publication May 29, 2023.

Refereeing of this contribution was handled by Stefano Coraluppi.

The authors are with the Intelligent Sensor-Actuator-Systems Laboratory (ISAS), 76131 Karlsruhe, Germany, and also with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany (e-mail: daniel.frisch@ieee.org; uwe.hanebeck@ieee.org).

This work is an enhanced and extended version of the FUSION 2021 conference paper [1].

Matlab source code is available here: https://codeocean.com/capsule/5750645/tree

I. INTRODUCTION

A. Context

In many practical applications, such as nonlinear filtering and control, moments of nonlinear functions of Gaussian random vectors must be approximated in real-time. Mathematically, this is a multidimensional integration, or cubature, that can be computationally very expensive. Nevertheless, filters and controllers often have to run under real-time constraints. A standard way to perform such integration is through Monte Carlo simulation using random samples. However, the convergence rate with independent samples is quite poor. Variance reduction techniques help to improve the efficiency of stochastic expectation value computations. After giving an overview of state-of-the-art variance reduction methods, we introduce novel Gaussian sampling schemes.

B. Considered Problem

We present a Gaussian sampling method for multivariate Gaussian densities based on a higher-dimensional generalization of the two-dimensional Fibonacci grid. See Fig. 1 for a visual comparison between random samples and proposed variance-reduced samples.

C. State-of-the-Art

Variance reduction techniques for expectation value calculations include antithetic variates [2], control variates [3], importance sampling [4], stratified sampling [5], low-discrepancy or quasi-random sampling [6], [7], moment matching [8]–[10], localized cumulative distribution (LCD)-based sampling [11], [12], and Projected Cumulative Distribution based sampling [13], [14]. These methods can also be combined, for example, LCD-based sampling with moment matching and antithetic variates [15]. In this work, we focus on Gaussian sampling and therefore present various state-of-the-art methods to obtain Gaussian samples in more detail. We comment on sampling techniques in common Gaussian estimators like the cubature Kalman filter (CKF), unscented Kalman filter (UKF), and Gaussian particle filter (GPF) and compare our proposed method against them.

The "standard" way of sampling from the normal distribution employs independent and identically distributed (iid) samples, e.g., by transforming iid uniform samples with the Box–Muller method [16]. This corresponds to standard Monte Carlo simulation. According to the central limit theorem (CLT) [17, p. 244], the standard deviation of the integration error equals the standard deviation of the integrand divided by the square root of the number of samples used [18, Sec. 2.1]. This slow convergence makes the computation inefficient.

Gauss–Hermite quadrature entails a finite set of predefined, weighted evaluation points for integration.

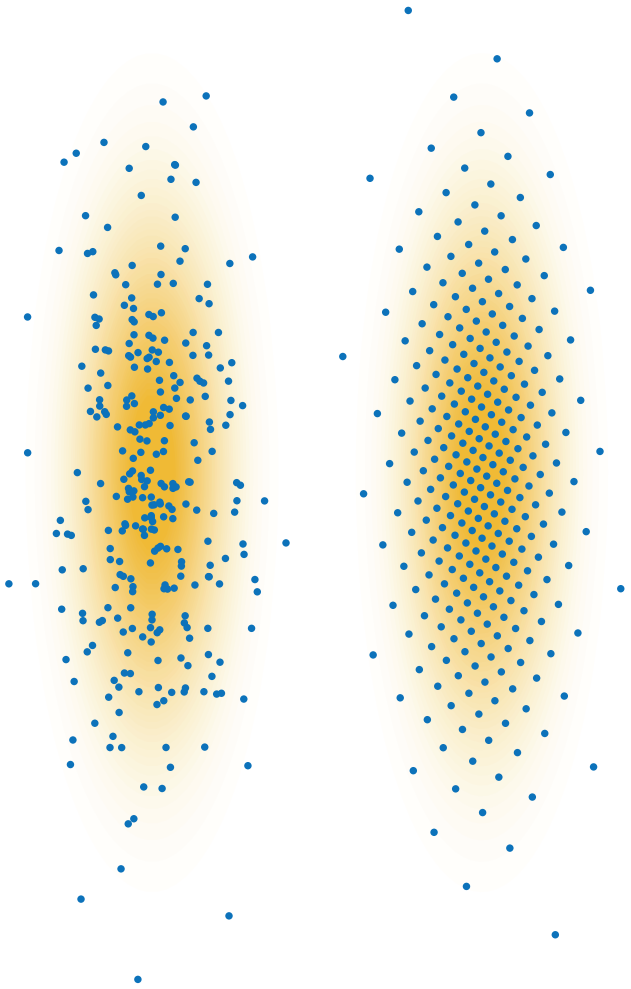


Fig. 1. Independent random samples (left) and Fibonacci-based samples (right) approximating a Gaussian density with covariance $\mathbf{C} = \mathrm{diag}([1, 0.3^2])$. A total of 300 samples are drawn.

It is ideally suited for scalar integrals of polynomial-like functions multiplied with a univariate Gaussian density function [19]. Extensions to higher dimensions require a Cartesian product of the evaluation points [20, Eq. (3.3)], [21], [22, Eq. (17)], so the number of required points increases exponentially with the number of dimensions. Kalman filters using this method for moment computation are called Gauss–Hermite quadrature filters (GHQF).

To avoid the "curse of dimensionality," one can place samples on the main axes only [23]. A more radical variant is the UKF, where only two samples, "sigma points," are placed on each coordinate axis [8], [24], plus one in the center, i.e., the number of samples is $L = 2D + 1$ for dimension D. The distances are chosen such that mean and covariance match. Very similarly, the third-order CKF places two samples on each coordinate axis, without the sample at the mode, hence $L = 2D$ [25]. The fifth-order CKF employs instead $L = 2D^2 + 1$ *weighted* samples [26], see also [27, Sec. 7], [28, Eqs. (48) and (49)]. The smallest possible sample set suitable to propagate mean and covariance has been explored in [29], [30]—it takes only $L = D+1$ or $L = D+2$ samples.

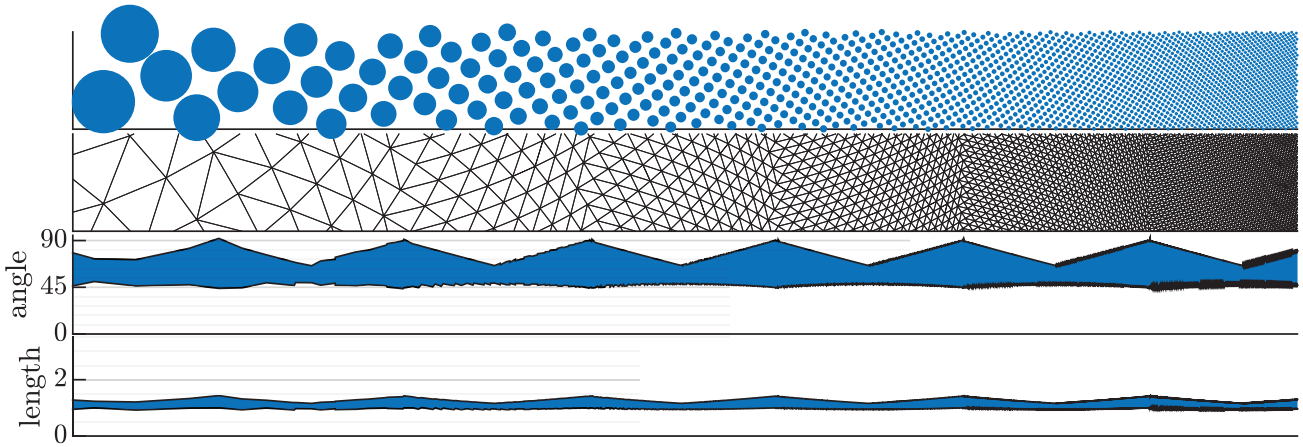
All these filters belong to the class of linear regression Kalman filters (LRKFs), introducing the second Gaussian assumption in the joint state and measurement space and thus performing an implicit linearization of the measurement equation. Particle filters avoid this and follow Bayes' theorem more directly. Thereby, the GPF [31], [32], its progressive variant [33], [34], and many other particle filters [35] need to draw medium to high numbers of samples from Gaussian priors, where our proposed Gaussian sampling technique could increase efficiency.

Now we focus on methods that allow the number of samples to be flexibly adapted to the problem and the desired accuracy. Given a suitable distance or optimality measure such as the LCD [11], optimal deterministic Gaussian sample sets can be computed using gradient optimization [36]. As this kind of sampling process is itself computationally expensive, for practical filtering it is necessary to compile a library of *standard* normally distributed samples beforehand, and transform them to the desired arbitrary Gaussian density online using the Cholesky factorization of its covariance matrix [12], [15], [37]. After such a transformation, however, the samples are usually no longer optimal as before [38, Figs. 4(a) and 5(a)].

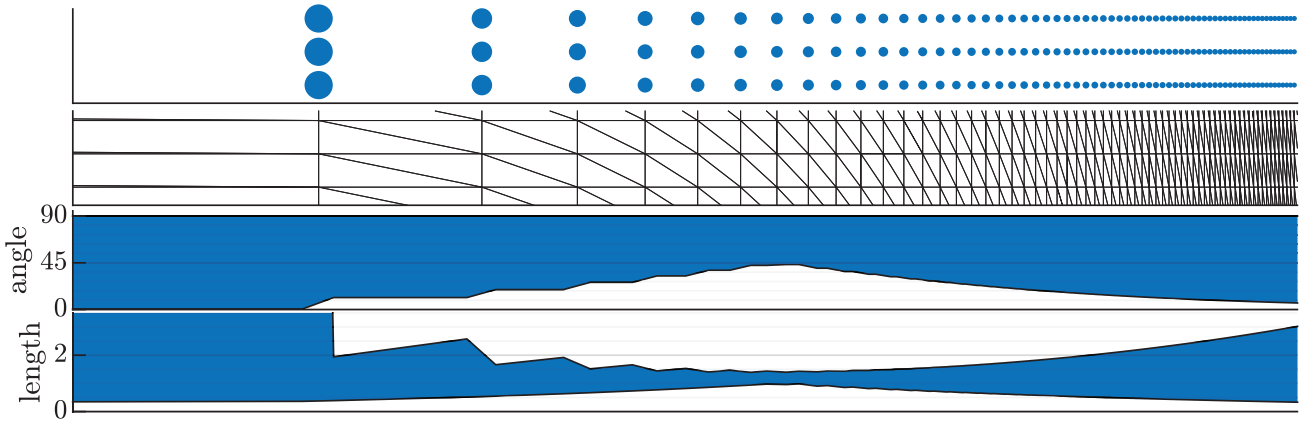
Therefore, it is convenient to use low-discrepancy sequences. They are exactly made to achieve optimal convergence when used for numerical integration—better than the well-known convergence rate of $\frac{1}{\sqrt{L}}$ for L samples obtained with independent random samples according to the CLT. This is also referred to as quasi-Monte Carlo integration [7]—as opposed to Monte Carlo integration with independent random samples. Refer to Section IV-C2 for a formal definition of discrepancy (13) and its relation to approximate cubature (14).

Low-discrepancy sequences can, under some constraints, be transformed to densities other than uniform while preserving their low discrepancy. These point sets have already successfully been applied to nonlinear filtering problems [39], [40], yet not using a discrepancy-preserving transformation as we propose; see Fig. 8 for a visual comparison. In the one-dimensional case, equidistant samples make the best possible low-discrepancy point set. In higher dimensions, Frolov and Fibonacci grids are the only low-discrepancy sequences known to attain the theoretical optimum under very universal conditions [41]—they are, so to speak, the "lowest-discrepancy grids." Thus, we can expect them to provide results similar to the optimal deterministic samples based on the LCD with nonlinear optimization while being transformable without compromising quality.

In 2008, James Purser published *generalized* Fibonacci grids for certain higher dimensions [42]. He formulates the reason why two-dimensional grids are optimal in such a deep way that higher-dimensional generalizations become tangible. The generalized Fibonacci grid has already been applied to Gaussian sampling [1] and rejection sampling [43].



(a) Rescaled Fibonacci Grid.



(b) Rescaled Regular Grid.

Fig. 2. Anisotropic scaling demonstration for Fibonacci grid (a) and regular grid (b). Scaling is altered exponentially along horizontal axis. Blue dots indicate grid points; black net the Delaunay triangulation of the same points. Note how in (a) the grid rearranges itself into square configurations of different sizes six times, evenly filling the space at any scaling, while in (b) there is only one square configuration (near the center) and away from that points clump together into horizontal or vertical lines, yielding bad space filling. For more quantitative assessment, maximum and minimum angle in Delaunay triangles, as well as maximum and minimum triangle side length (normalized to the side length of a square of appropriate size) are shown as well.

D. Key Idea

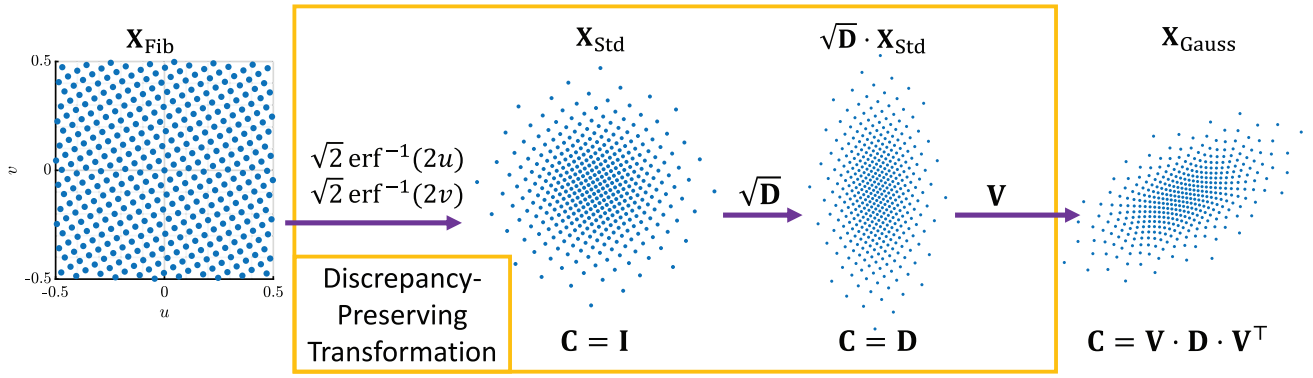
To produce univariate Gaussian samples, uniform samples can be transformed by the inverse Gaussian distribution. For multivariate Gaussians, this scalar transformation is applied along the directions of the eigenvectors of the covariance matrix, respectively. By doing so, the distribution of samples should stay locally homogeneous, i.e., without forming clumps or gaps. Therefore, we need a uniform point set being collision-avoiding under rescaling along certain axes.

An ideal candidate is the Fibonacci grid, as it can be anisotropically rescaled along the main axes while preserving the uniformity of points. Instead of colliding, Fibonacci grid points automatically get new neighbors, depending on the amount of rescaling. Refer to Fig. 2(a) for a visual demonstration of how the well-known two-dimensional Fibonacci grid remains uniform under inhomogeneous horizontal scaling, very much unlike the axis-aligned regular grid in Fig. 2(b). This remarkable property is what we take advantage of in this work. Grids with equivalent properties also exist in higher dimensions—the *generalized* Fibonacci grids [42].

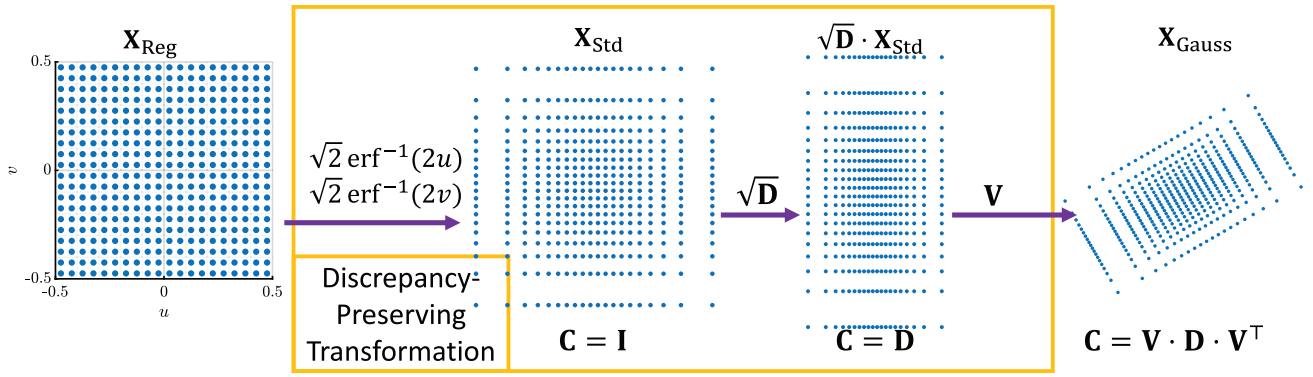
With a suitable mapping, these uniform samples can be transformed to an arbitrary density, similar to the well-known "inverse transform sampling" method. We introduce such a mapping for the Gaussian density. Refer to Fig. 3(a) for a visual demonstration of the mapping workflow using Fibonacci samples as compared to, e.g., a regular grid Fig. 3(b). In addition, we introduce some methods for moment matching so that the covariance of the samples is accurate to machine precision.

E. Overview

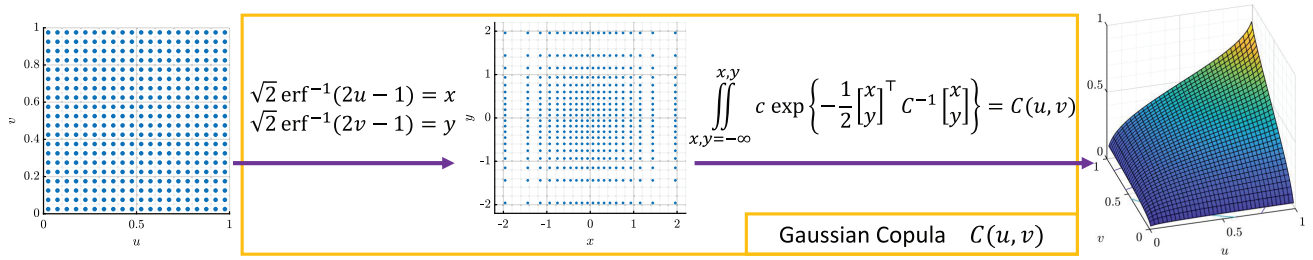
This paper is structured as follows: After explaining well-known and optimal two-dimensional uniform samples in Section II, we generalize to higher-dimensional uniform samples in Sections III and IV. Then in Section V, we explain how to obtain Gaussian instead



(a) Discrepancy-Preserving Transformation: Low-Discrepancy / Fibonacci Uniform → Arbitrary Gaussian.



(b) Same Transformation, but: Regular (*not* low-discrepancy) Uniform → Arbitrary Gaussian.



(c) Gaussian Copula Distribution (only for reference).

Fig. 3. (a) Transformation workflow from uniform distribution to arbitrary Gaussian. As opposed to the commonly used transformation via Cholesky factorization, see Fig. 8(a), our approach preserves the discrepancy of the input point set and is therefore better suited for low-discrepancy sequences. (b) Same transformation is applied to an axis-aligned regular grid instead of low discrepancy. Note how resulting Gaussian samples fill the space less homogeneously. (c) Gaussian copula, shown only for reference. Note that the first step in its transformation pipeline is related to what we discuss here.

of uniform samples and evaluate their optimality in Section VI.

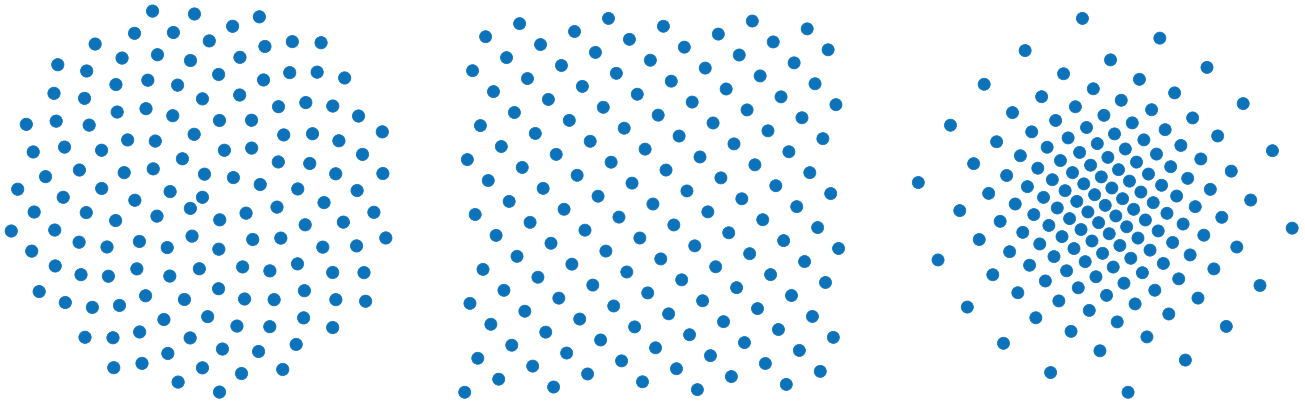
The merits of our Gaussian samples lie in (i) providing superior coverage of the state space and (ii) free choice of the number of samples. Thus, they can improve the convergence and accuracy of algorithms that utilize Gaussian samples, e.g., sample-based Gaussian state estimation filters and controllers.

II. THE TWO-DIMENSIONAL FIBONACCI GRID

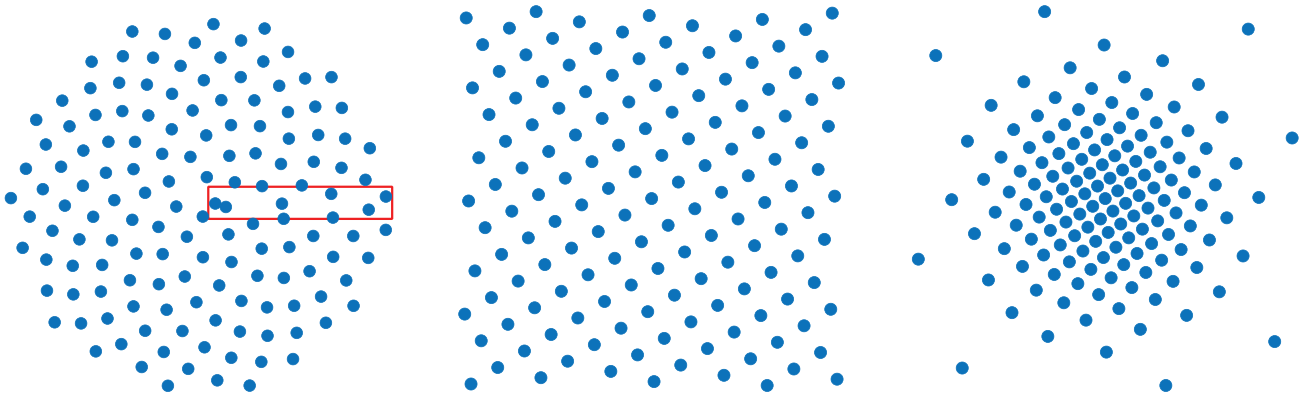
Two-dimensional Fibonacci grids have been known for a long time since they are ubiquitous in plant life. Seed heads are often arranged as a polar Fibonacci grid. Due to the size of its seeds, this is best seen in the sunflower, but a close look reveals similar structures in many other flower heads. It is well known that the two-dimensional Fibonacci grid is the best possible low-discrepancy point set [44, p. 186], [45, p. 61].

Arranging the seeds on a Fibonacci grid has two advantages. First, the space is well utilized and second, the arrangement is flexibly scalable along the radius. The former is important to utilize biological resources as efficiently as possible, and the latter is necessary because the whole thing is growing, with bigger seeds on the outside and younger, smaller seeds near the center. Although a hexagonal arrangement would make even better use of the space, this would require all seeds to always be of the same size.

In summary, the polar Fibonacci grid can be anisotropically rescaled along the radius—and the angle, for that matter, i.e., both main axes of the polar coordinate



(a) Lattice Rule – Polar. (b) Lattice Rule – Cartesian. (c) Lattice Rule – Gaussian.



(d) Frolov – Polar. (e) Frolov – Cartesian. (f) Frolov – Gaussian.

Fig. 4. Fibonacci grids computed via the lattice rule (a)–(c) and Frolov method (d)–(e). Points are arranged uniformly in polar coordinates on the left, in Cartesian coordinates in the middle, and in Gaussian density on the right.

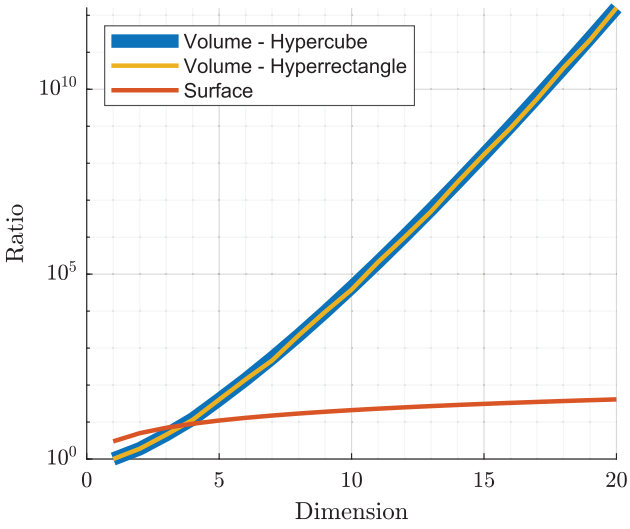


Fig. 5. Volume of the smallest hypercube (blue) and smallest hyperrectangle (yellow) that encloses the unit hypercube that is rotated by $\mathbf{V}^\top$. Note that there is no big difference between hypercube and hyperrectangle. Note also that the ratio for both increases exponentially with the dimension. For the linear programming method, the ratio between surface (red) and volume is more important. It increases much slower with the number of dimensions.

system. In Cartesian coordinates, the Fibonacci grid allows anisotropic scalings along the horizontal and vertical axes. This can be seen in Fig. 2(a), where the horizontal scaling of a Fibonacci grid is varied while the vertical scaling stays constant. Instead of colliding, the points change their neighborhood relationships periodically. Note how the Fibonacci grid repeatedly returns to a "regular grid" configuration, i.e., the Delaunay interior angles are 45° and 90°, and the normalized side lengths are 1 and $\sqrt{2}$ over and over again, only at different scales. For comparison, the same anisotropic scaling performed on a regular axis-aligned grid does produce point collisions, Fig. 2(b).

A. Fibonacci Matrix

The Fibonacci numbers F_k are defined as [46, Sec. 6.6]

$$F_{k+1} = F_k + F_{k-1}\,, \qquad F_0 = 0\,, \qquad F_1 = 1\,.$$

This recurrence can be expressed with the Fibonacci matrix

$$\mathbf{M} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \tag{1}$$

where Fibonacci numbers are then generated as

$$\begin{bmatrix} F_{k+1} \\ F_k \end{bmatrix} = \mathbf{M} \cdot \begin{bmatrix} F_k \\ F_{k-1} \end{bmatrix} \ .$$

With the eigenvalue decomposition of the Fibonacci matrix

$$\mathbf{M} = \mathbf{V} \cdot \mathbf{D} \cdot \mathbf{V}^\top \ , \tag{2}$$

we define the orthogonal unitary matrix $\mathbf{V}$ containing the eigenvectors of $\mathbf{M}$, and diagonal matrix $\mathbf{D}$ containing the eigenvalues.

Note that $\mathbf{M}$ is unimodular, i.e., it consists of integers and has a unit absolute determinant. The former implies that $\mathbf{M}$ transforms integer vectors $\underline{z}$ into other integer vectors $\mathbf{M}\underline{z}$, and the latter implies that convex sets of integer vectors $\underline{z}$ stay convex after transformation, i.e., no empty holes would appear. When the entire integer lattice $\mathbb{Z}^2$ is transformed by $\mathbf{M}$, exactly the same integer lattice comes out, as lattice points are indistinguishable. Relaxing the unit determinant restriction and including intermediate configurations in between the square-lattice configurations, we can apply continuous scaling, as visualized in Fig. 2(a), where square lattice configurations are reached at six instances. But the fact that we run into square lattice configurations again and again guarantees that the points will always have a homogeneous microstructure and never collide, as in Fig. 2(b).

B. Rank-One Lattice

Mathematically, the Fibonacci grid is often represented as a rank-1 lattice rule. To produce lattice point $\underline{x}_i$, a generating vector is multiplied with an integer index i, and the result is taken modulo 1

$$\underline{x}_i = \frac{i}{F_{k+1}} \cdot \begin{bmatrix} 1 \\ F_k \end{bmatrix} \mod 1 \ , \tag{3}$$
$$i = 0, 1, \ldots, F_{k+1} - 1 \ ,$$

where F_k is the kth Fibonacci number [7, Ex. 2.8]. The result is a Cartesian Fibonacci grid with F_{k+1} samples in $[0, 1)^2$. An example with $F_k = 144$ is shown in Fig. 4(b). The grid can not only be anisotropically rescaled along its coordinate axes [as demonstrated in Fig. 2(a)] but also transformed to other coordinate systems while maintaining its packing efficiency. Transformation to polar coordinates (and proper scaling along the radius axis) yields the conspicuous sunflower pattern; see Fig. 4(a).

C. Frolov Lattice

A slightly different Fibonacci grid can be computed as a Frolov lattice. Here, the regular axis-aligned integer grid $\mathbb{Z}^2$ is rescaled with factor δ (to achieve the desired number of points L) and linearly transformed with matrix $\mathbf{T}$ (e.g., a rotation matrix). The result is then confined to the unit square $[0, 1]^2$

$$\left\{\underline{x}_i\right\}_{i=1}^{L} = \left\{\mathbf{T} \cdot \delta \cdot \underline{z} : \underline{z} \in \mathbb{Z}^2\right\} \cap [0, 1]^2 \ . \tag{4}$$

Now, we use the eigenvectors of the Fibonacci matrix (2) as the linear transformation $\mathbf{T}$

$$\mathbf{T} = \mathbf{V}^\top \ , \tag{5}$$

with $\mathbf{V}^\top$ meaning the transpose of $\mathbf{V}$. This again yields a two-dimensional Cartesian Fibonacci grid; see Fig. 4(e). As opposed to (3), this grid is not periodic; therefore, a transformation to polar coordinates is not smooth at the angular coordinate's transition between 0 and 2π; see the red box in Fig. 4(d).

In this work, we focus on the nonperiodic Fibonacci lattice that is computed via Frolov-like construction. The advantages of nonperiodic generalized Fibonacci grids are their visually appealing symmetry, and that they can be generated for arbitrary numbers of points. On the downside, their generation becomes more difficult in higher dimensions.

III. PURSER'S GENERALIZED FIBONACCI GRID

James Purser showed that higher-dimensional generalizations with optimality properties analogous to the two-dimensional Fibonacci grid do exist [42]. Purser's higher-dimensional Fibonacci grid is based on a new theory that captures the concept behind two-dimensional Fibonacci grids on a deep level. From that perspective, it is then easy to see how Fibonacci-type grids can be conceptualized in higher dimensions as well. The theory involves quasi-Fibonacci matrices that generalize (1) to higher dimensions. Specific constructions are stated for dimensions D with the restriction that $(2D+1)$ is a prime number.

A. The Quasi-Fibonacci Matrix

The D-dimensional quasi-Fibonacci matrix $\mathbf{M}$ according to Purser [42, Appendix A] is given by

$$[\mathbf{M}]_{i,j} = \begin{cases} 1, & i + j \leq D + 1 \\ 0, & i + j > D + 1 \end{cases} \ , \tag{6}$$

for example,

$$\mathbf{M}^{(D=2)} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \ , \quad \mathbf{M}^{(D=3)} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \ .$$

Note that $\mathbf{M}^{(D=2)}$, also (1), is known as "Fibonacci Q-Matrix" [47], but attempts to generalize to higher dimensions, e.g., in [47] are different from the concept in [42, Appendix A] that we pursue here. An eigenvalue decomposition

$$\mathbf{M} = \mathbf{V} \cdot \mathbf{D} \cdot \mathbf{V}^\top \ ,$$

again splits $\mathbf{M}$ into unitary $\mathbf{V}$ and diagonal $\mathbf{D}$. The eigenvectors $\mathbf{V}$ can also be obtained by properly normalizing the unnormalized eigenvector matrix $\mathbf{V}^{\mathrm{u}}$, which is given

in closed form by [42, Eq. (A.4)]

$$[\mathbf{V}^{\mathrm{u}}]_{i,j} = \cos\left(\frac{\pi}{2} \cdot \frac{(2i-1)\,(2j-1)}{2D+1}\right) \quad ,$$
$$i, j \in \{1, 2, \ldots, D\} \quad . \tag{7}$$

B. Dimensions With $(2D + 1)$ Prime

In dimensions where $(2D + 1)$ is prime, the eigenvector matrix $\mathbf{V}$ of the generalized Fibonacci matrix $\mathbf{M}$ is used in Frolov lattice creation (4) just as in the two-dimensional case (5).

C. Other Dimensions

In dimensions where $(2D + 1)$ is not prime, (6) is not well suited: One column of $\mathbf{V}$ has entries with identical fractional parts because in (7) the numerator $(2i-1)\,(2j-1)$ "interferes" with the denominator $2D+1$. However, it is possible to search for alternative matrices. An example for $D = 4$ is given in [42, Sec. 7] as

$$\mathbf{M}^{(D=4)} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad .$$

It consists of block-diagonal replications of $\mathbf{M}^{(D=2)}$.

IV. Optimal Deterministic Uniform Fibonacci Grids

In this section, we describe how to enumerate the L samples $\underline{x}_i$ of the generalized Fibonacci grid using Frolov-like construction

$$\{\underline{x}_i\}_{i=1}^{L} = \{\mathbf{V}^{\top} \cdot \delta \cdot \underline{z} : \underline{z} \in \mathbb{Z}^D\} \cap \left[-\frac{1}{2}, \frac{1}{2}\right]^D \quad ,$$

where δ specifies the number of points L approximately according to $\delta \approx L^{-1/D}$, see Section IV-C for more details. Note that to simplify notation, we changed the unit hypercube under consideration from $[0, 1]^D$ to $\left[-\frac{1}{2}, \frac{1}{2}\right]^D$. The computation can be done by (i) enumerating the grid points of a regular grid inside the rotated hypercube or (ii) enumerating the grid points of a rotated regular grid in an axis-aligned hypercube.

A. Enclosing Hypercube Counter

The most simple and obvious method works as follows: Find the smallest axis-aligned hyperrectangle or hypercube that encloses the desired rotated hypercube. Iterate through all the points of the axis-aligned hyperrectangle or hypercube, check if the point is inside or outside the rotated hypercube, and return all points that are inside [1, Alg. 1].

How large is the smallest axis-aligned unit hyperrectangle enclosing a rotated unit hypercube? To find out it

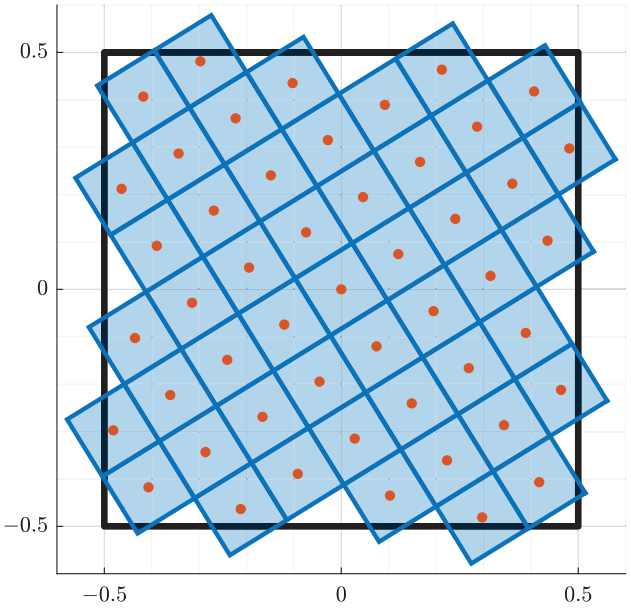


Fig. 6. Fibonacci grid with unit cells of side length $\delta = 50^{-1/2}$, calculated by $L_{\mathrm{vol}} = 50$ according to (10). The actual number of grid points turns out to be $L_2 = 49$.

is sufficient to examine, the 2^D corners $\underline{x}_{\mathrm{crn}}$ of the centered rotated hypercube. Their coordinates are

$$\underline{x}_{\mathrm{crn,j}} = \mathbf{V} \cdot \underline{u}_j \quad ,$$
$$\underline{u}_j \in \left\{-\frac{1}{2}, \frac{1}{2}\right\}^D \quad ,$$
$$j \in \left\{1, 2, \ldots, 2^D\right\} \quad .$$

Therefore, the side lengths β_d of the smallest enclosing hyperrectangle are

$$\beta_d = \sum_{j=1}^{D} \left|\mathbf{V}_{d,j}\right| \quad , \qquad d \in \{1, 2, \ldots, D\} \quad ,$$

and the side length β of the smallest enclosing hypercube is

$$\beta = \max_d \left\{\sum_{j=1}^{D} \left|\mathbf{V}_{d,j}\right|\right\} = \|\mathbf{V}\|_{\infty} \quad .$$

For simplicity, we will focus on the hypercube instead of the hyperrectangle here, as for generalized Fibonacci matrices obtained by (6), the smallest hyperrectangle is very close to a hypercube; see Fig. 5.

We can now define a sampling vector $\underline{r} \in \mathbb{R}^{L_1}$ with centered, equidistant elements r_j that represent the grid coordinates along each dimension

$$r_j = \delta \cdot \left(j + \frac{1 - L_1}{2}\right) \quad ,$$
$$j \in \{0, 1, \ldots, L_1 - 1\} \quad . \tag{8}$$

After replicating $\underline{r}$, we obtain a regular grid with L_1^D elements and spacing δ, stored column-wise in the matrix $\mathbf{X}_{\mathrm{reg}} \in \mathbb{R}^{D \times L_1^D}$. If L_1 is odd, then there will be a sample at

the origin, otherwise not. This grid is then transformed according to $\mathbf{V}^\top \mathbf{X}_{\text{reg}}$, followed by a rejection of points outside the centered unit hypercube $\left[-\frac{1}{2},\ \frac{1}{2}\right]^D$. The result are L_2 Fibonacci grid points $\mathbf{X}_{\text{Fib}} \in \mathbb{R}^{D\times L_2}$ that uniformly cover the centered unit hypercube $\left[-\frac{1}{2},\ \frac{1}{2}\right]^D$. By adding $\frac{1}{2}$, this can be transformed to cover the "standard" unit hypercube $[0,\ 1]^D$. See Fig. 6 for a visualization of the finally obtained samples inside the square for $D = 2$.

The volume ratio between the smallest enclosing hypercube β^D and a unit hypercube $1^D = 1$ increases exponentially with the dimension D; see Fig. 5. This means that the majority of samples are rejected in higher dimensions, and the method is applicable in dimensions smaller than 10 only, refer to Fig. 11(c) for more details.

B. Linear Programming Counter

In this section, we describe a method that avoids the excessive rejection of the enclosing hypercube counter. It is based on linear programming. The complexity is related more to the surface of the unit hypercube than to the volume of the enclosing hypercube; see Fig. 5. The generalized Fibonacci grid can be written as a system of linear inequalities for integer vectors that describe a bounded polytope—so to speak, a "Diophantine inequality system," yet with real (instead of rational) coefficients. In this perspective, the first step is to find all integer vectors $\underline{z}$ such that

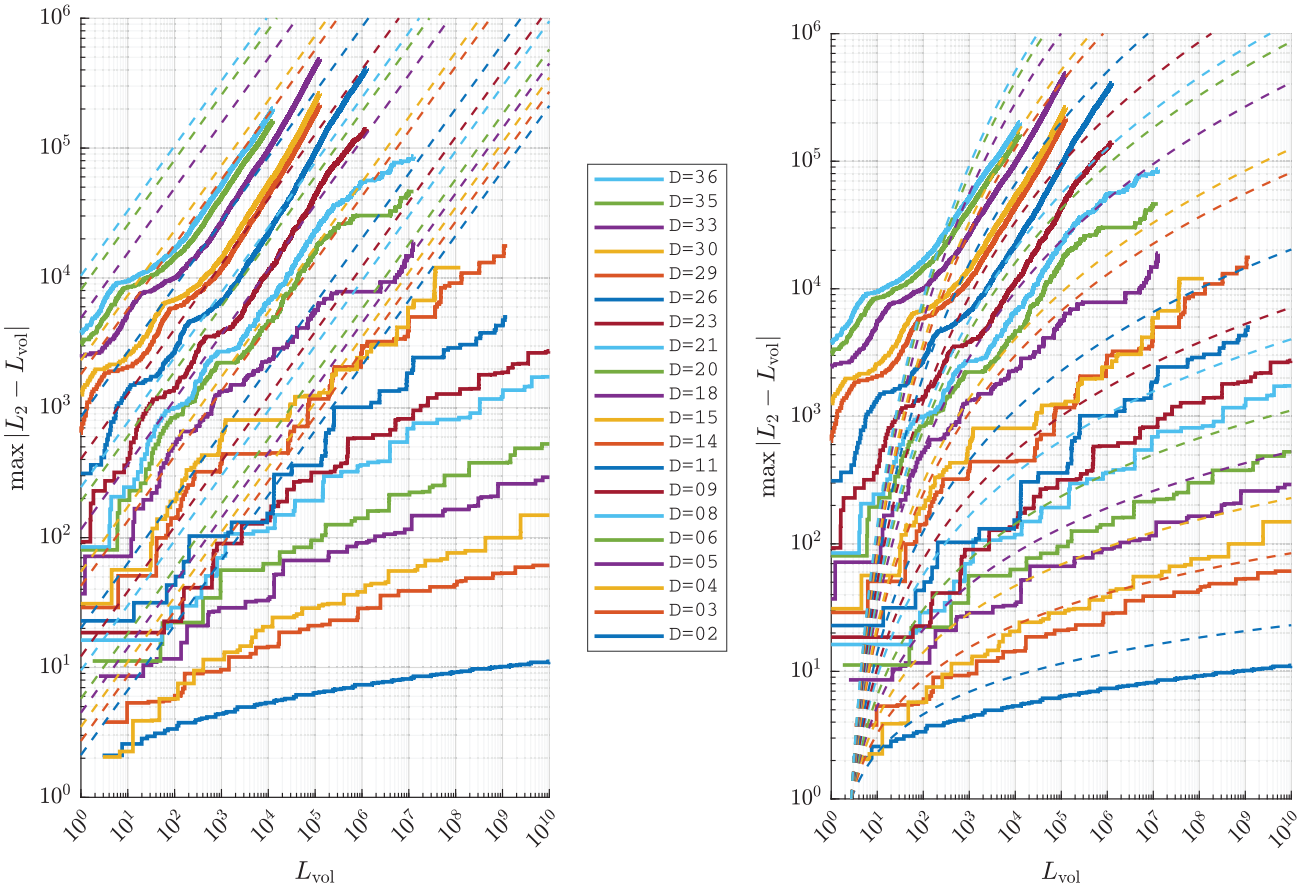
$$\mathbf{A}\,\underline{z} \le \underline{b}\ , \tag{9}$$

with coefficient matrix $\mathbf{A} \in \mathbb{R}^{N_c\times D}$, desired vectors $\underline{z} \in \mathbb{Z}^D$, and vector $\underline{b} \in \mathbb{R}^{N_c}$. Thereby, N_c is the number of linear inequality constraints that define the bounded polytope. For Fibonacci sampling, we define

$$\mathbf{A} = \delta \cdot \begin{bmatrix} \mathbf{V}^\top \\ -\mathbf{V}^\top \end{bmatrix}\ , \qquad \underline{b} = \frac{1}{2} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}\ ,$$

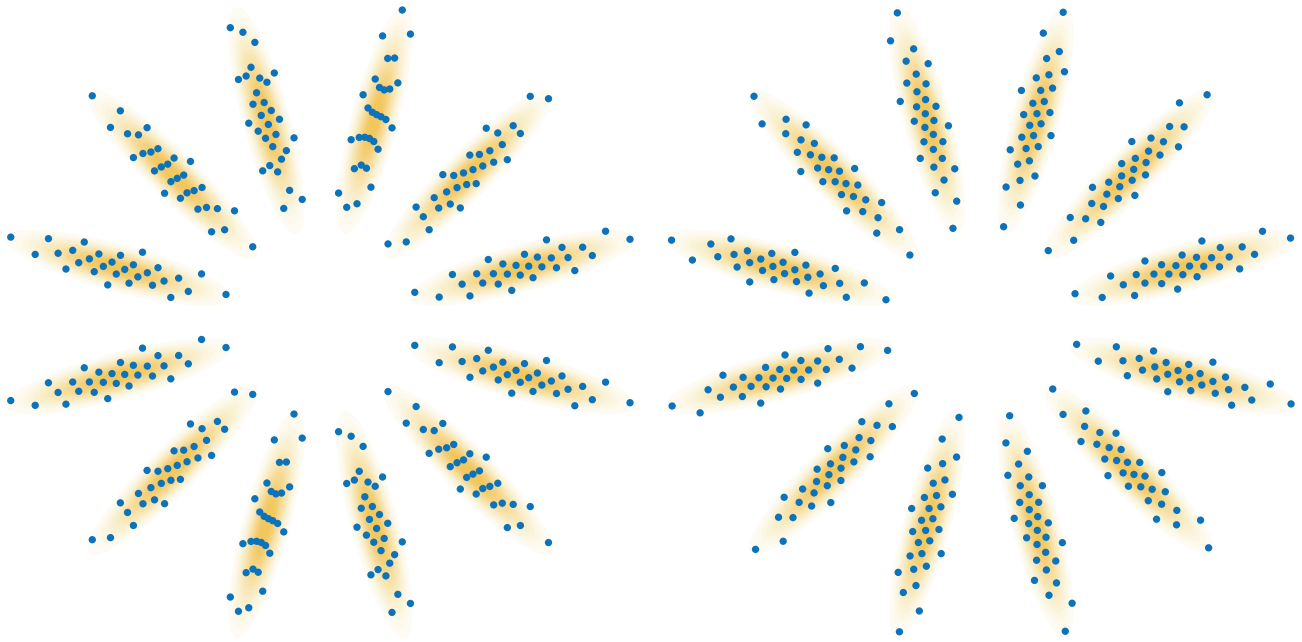
and find all integer vectors $\underline{z}$ with $\mathbf{A}\,\underline{z} \le \underline{b}$.

The method to obtain these points can best be described as a recursive procedure. Initially, we focus on the first coordinate z_1 and find its minimum and maximum values inside the polytope $\mathbf{A}\,\underline{z} \le \underline{b}$ via linear programming or integer linear programming. Then, recursively for each integer value $\hat{z}_{1,k}$ between said minimum and maximum: fix $z_1 = \hat{z}_{1,k}$ as a constant temporally. In the



(a) CLT (b) Discrepancy

Fig. 7. Absolute difference between the expected number of points due to the volume and the obtained number of points in generalized Fibonacci grids (solid lines). Dotted lines show bounds based on the CLT (a) and the point discrepancy (b). For all sets of curves, the lowest dimension $D = 2$ appears at the bottom, with the higher dimensions following upwards according to the legend.



(a) Non-discrepancy-preserving transformation. (b) Discrepancy-preserving transformation.

Fig. 8. Low-discrepancy point set transformed from uniform to Gaussian via nondiscrepancy-preserving transformations [39], [40] (a) and via discrepancy-preserving transformations as proposed here (b). While in (a), the sampling quality depends on the orientation of the Gaussian, in (b), it is always the same.

thus defined polytope

$$\mathbf{A} \cdot \underline{z} \leq \underline{b} \ \cap \ z_1 = \hat{z}_{1,k} \ ,$$

find the minimum and maximum values of z_2. Repeat for all integer values $\hat{z}_{2,k}$ in that range. By doing so recursively for all dimensions $1 \dots D$, all L_2 integer vectors that fulfill (9) are visited and collected in a matrix $\mathbf{Z} \in \mathbb{Z}^{D \times L_2}$. The Fibonacci point set $\mathbf{X}_{\text{Fib}} \in \mathbb{R}^{D \times L_2}$ that is uniform in $\left[-\frac{1}{2}, \ \frac{1}{2}\right]^D$ can then be derived from the obtained integral vectors $\mathbf{Z}$ via

$$\mathbf{X}_{\text{Fib}} = \mathbf{V}^\top \cdot \delta \cdot \mathbf{Z} \ .$$

An iterative version of this algorithm avoiding explicit recursion has been implemented in this work. For better efficiency it uses the GNU Linear Programming Kit GLPK [48].

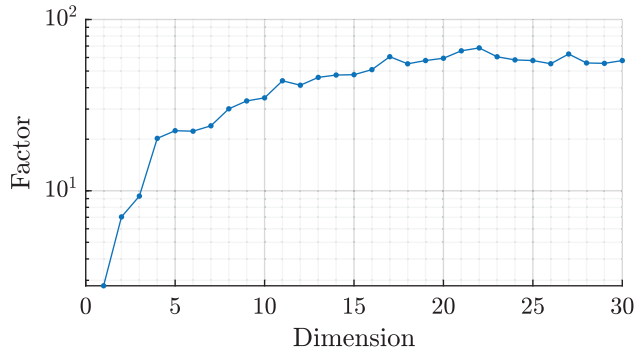


Fig. 9. Ratio of computation time between eigenvalue decomposition and the faster Cholesky factorization of random symmetric and positive definite matrices in Matlab.

Unfortunately, computational complexity increases exponentially with the dimension here as well, but slower than in the enclosing hypercube counter. This method is practical for dimensions up to about 20; see Fig. 11(c). Note that the samples, once created, can be stored and used henceforth to obtain sample sets of the same dimensionality.

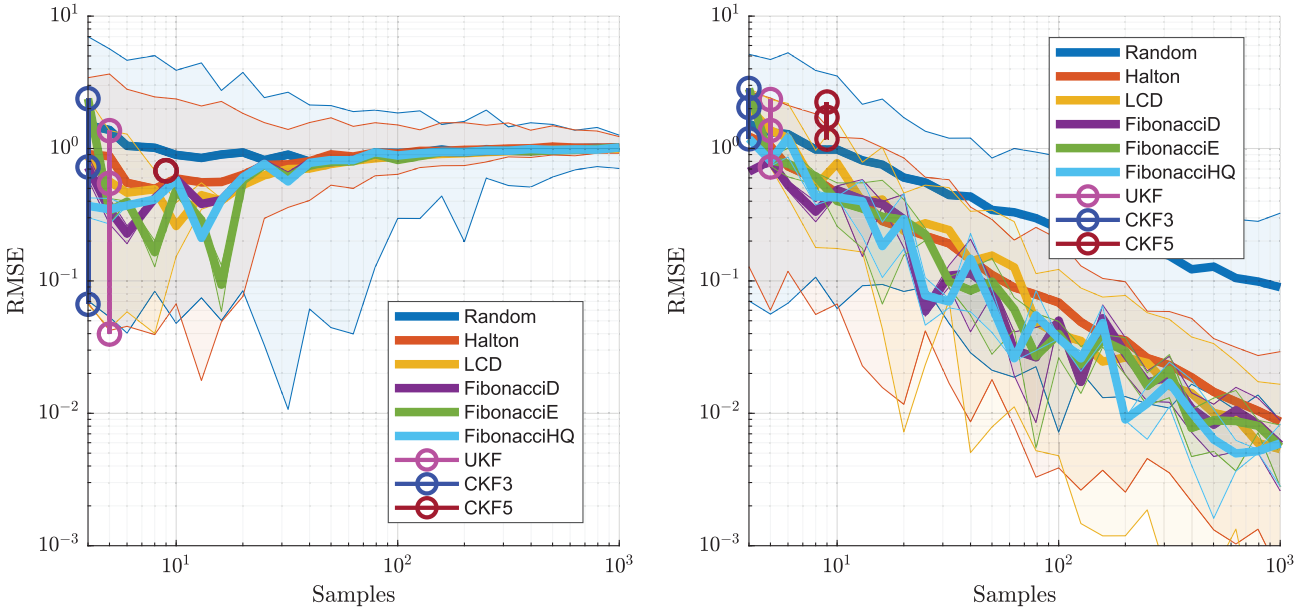
C. Number of Points Obtained

In this section, we will elaborate on the number L_2 of grid points that can be expected to be inside a rotated hypercube, or, equivalently, how many rotated grid points can be expected inside an axis-aligned hypercube. Of course, a rough estimate L_{vol} is the ratio between the volume of a unit cell δ^D representing one sample and the volume of the rotated hypercube, which is one, therefore

$$L_2 \approx L_{\text{vol}} = \delta^{-D} \ ,$$
$$\delta = {L_{\text{vol}}}^{-1/D} \ . \tag{10}$$

However, due to the rotation between unit cells and the rotated hypercube, this estimate is not necessarily correct; see Fig. 6.

Therefore, we aim to quantify the worst case of how many "missing" or "surplus" points we can expect in L_2 compared to L_{vol}. This is related to the Gauss circle problem, where the number of two-dimensional integer lattice points inside a circle with a given radius is determined or approximated. It is even more related to the convergence rate of Monte Carlo and quasi-Monte Carlo methods. To quantify this, we define a "hypercube



(a) UKF, cubic measurement update

(b) GPF, cubic measurement update

Fig. 10. Cubic measurement update, system model as explained in Section VI-A. Out of 100 trials, minimum, maximum, and mean RMSE between the estimated and true posterior mean are indicated, respectively. FibonacciD means diagonal variance matching according to (17); the same correction method has been applied to the quasi-random Halton [55] samples. FibonacciE means exact covariance matching by the eigenvalue method (19) that is however non discrepancy-preserving. FibonacciHQ means "high quality" covariance matching (21). Also included are Unscented Kalman Filter (UKF) samples [9] with scaling such that they have equal weights, CKF3 samples [25], and CKF5 samples [26].

function" $h_a(\cdot)$ centered around the origin

$$h_a(\underline{x}) = \begin{cases} 1, & |x_i| < \frac{a}{2} \quad \forall\, i \in [1, D] \\ 0 & \text{otherwise} \end{cases} ,$$
$$0 < a < 1 \ ,$$

and a centered unit-size hypercube

$$\mathcal{I} = \left[-\frac{1}{2}, \frac{1}{2}\right]^D \ .$$

1) CLT: The CLT states [18, Sec. 2.1] that the Monte Carlo integration error $\epsilon_{L,f}$ of an arbitrary function $f(x)$ over a unit cube $\mathcal{I}$

$$\epsilon_{L,f} = \left| \left(\frac{1}{L} \sum_{n=1}^{L} f(\underline{x}_n) \right) - \left(\int_{\mathcal{I}} f(\underline{x})\, d\underline{x} \right) \right|$$

(with a large number L of i.i.d. uniform random samples $\underline{x}_n$ on $\mathcal{I}$) is normally distributed with standard deviation $\sigma_f \cdot L^{-1/2}$, where

$$\sigma_f = \sqrt{\int_{\mathcal{I}} \left(f(\underline{x}) - \left(\int_{\mathcal{I}} f(\underline{\tilde{x}})\, \mathrm{d}\underline{\tilde{x}} \right) \right)^2 \mathrm{d}x} \ .$$

Thus, relying on a c-sigma-bound, we may assume

$$\epsilon_{L,f} \ \leq \ c \cdot \sigma_f \cdot L^{-1/2}$$

with high probability. Now we multiply both sides with L and insert h_a for f. This yields

$$\left| L_2 - L \cdot a^D \right| \ \leq \ c \cdot \sigma_h \cdot \sqrt{L} \ ,$$
$$\sigma_h = \sqrt{a^D - a^{2D}} \ ,$$

and with $L_{\text{vol}} = L \cdot a^D$, we write

$$|L_2 - L_{\text{vol}}| \ \leq \ c \cdot \sqrt{(1 - a^D) \cdot L_{\text{vol}}} \ .$$

For $a \to 0$, this finally becomes

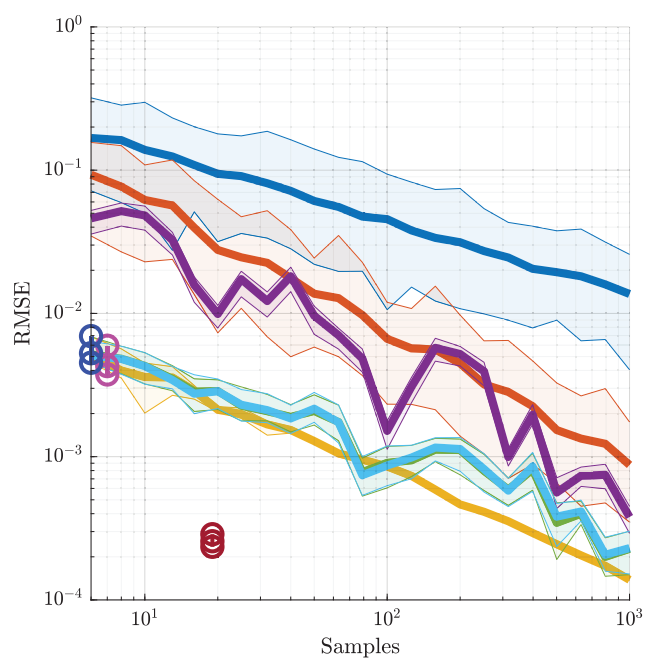
$$|L_2 - L_{\text{vol}}| \ \leq \ c \cdot \sqrt{L_{\text{vol}}} \ ,$$
$$L_2 \ \geq \ L_{\text{vol}} - c \cdot \sqrt{L_{\text{vol}}} \ . \tag{11}$$

Solving this for L_{vol}, we see that we should select

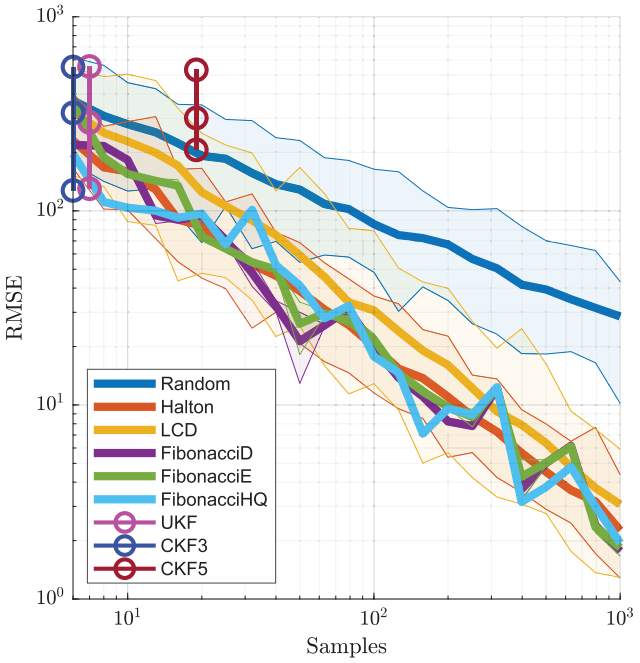
$$L_{\text{vol}} \geq L_2 + \frac{c^2}{2} + \sqrt{L_2 \cdot c^2 + \frac{c^4}{4}} \ , \tag{12}$$

and then δ according to (10), to always obtain at least the desired number of samples L_2. Fig. 7(a) shows a numerical overview, where $c = \exp\{\frac{D+1}{4}\}$ has been selected heuristically as the y-intercept. Note that for large L and in dimensions lower than 10, the CLT is a rather conservative estimate because the generalized Fibonacci points have a lower discrepancy and therefore better convergence rate than iid samples that the CLT assumes.

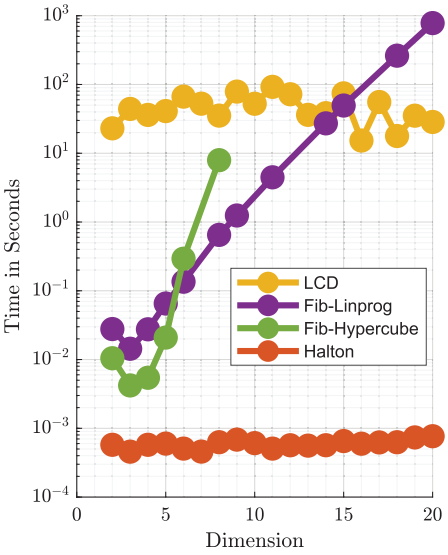
2) Discrepancy: We give an intuitive definition of the discrepancy of a point set $\hat{\underline{x}}_i \in [0, 1]^D, \ i \in [1, 2, \ldots, L]$.



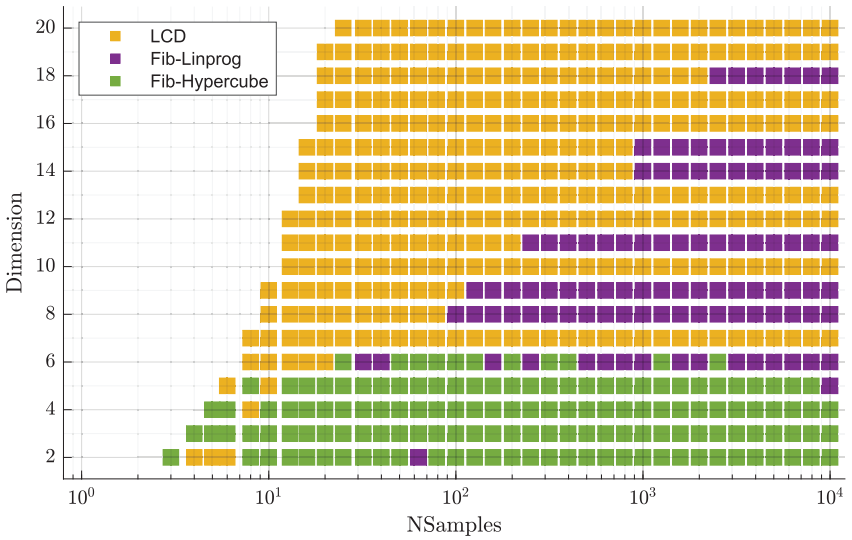
(a) Optimality, Standard Normal, $\tau = 0.5$



(b) Optimality, Gaussian with $\mathbf{C} = \mathrm{diag}([1, 0.1, 0.01])^2$, $\tau = 50$



(c) Computation Time



(d) Fastest Method

Fig. 11. (a) Fourier-based optimality measure with isotropic Gaussian. (b) Fourier-based optimality measure with strongly nonisotropic Gaussian. The best, worst, and mean RMSE of 100 trials are shown, respectively. Note that for a given accuracy, far less Fibonacci samples than, e.g., random samples are needed. (c) Calculation times for 1 000 optimal deterministic samples. Note that samples can be generated offline and tabulated for given D and L, for subsequent real-time use. (d) The fastest optimal deterministic sampling method for given number of samples and dimension. Fibonacci samples are only given for dimensions where $2D + 1$ is prime and $D = 4$, i.e., where suitable unimodular matrices are currently known.

Consider the volume of a hyperrectangle spanned between the origin $\underline{0}$ and point $\underline{x} \in [0,1]^D$. The proportion of points inside this hyperrectangle minus its volume yields the local discrepancy function $\Delta(\underline{x})$ of the point set. Aggregating all local discrepancies via a p-norm yields the discrepancy [7]

$$\mathrm{discr}_p = \left(\int_{\underline{x} \in [0,1]^D} |\Delta(\underline{x})|^p \right)^{\frac{1}{p}} . \tag{13}$$

According to the Koksma–Hlawka identity

$$\left| \frac{1}{L} \sum_{i=1}^{L} g(\underline{\hat{x}}_i) - \int_{[0,1]^D} g(\underline{x})\, d\underline{x} \right| \leq \mathrm{discr}_p \cdot V(g) \ , \tag{14}$$

i.e., the integration error is bounded by the discrepancy of the point set times the variation $V(g)$ of the integrand, a constant that depends on the smoothness of $g(\cdot)$. Hence, the discrepancy, as a function of the number of

samples L, is a measure for integration error, similar to the $L^{-1/2}$ term in the CLT integration error estimate. A proven lower bound on the $\mathcal{L}^2$ discrepancy of any finite point set is [49, Sec. 2]

$$L \cdot \mathrm{discr}_2 \geq c \cdot (\log L)^{(D-1)/2} \quad .$$

For $D = 2$, this is also known to be the best possible bound [49, Sec. 3]. Unfortunately, there is currently no known upper bound on the $\mathcal{L}^\infty$ discrepancy. Conjectures include

$$L \cdot \mathrm{discr}_\infty \leq c \cdot (\log L)^{D-1} \quad ,$$

$$L \cdot \mathrm{discr}_\infty \leq c \cdot (\log L)^{D/2} \quad ,$$

with respective unknown constants c that depend on the dimension [50, Sec. 4.2]. In Fig. 7(b), we plot the function

$$|L_2 - L_{\mathrm{vol}}| \leq \log(L_{\mathrm{vol}})^{\sqrt{D-1}} \quad . \tag{15}$$

It can be seen in Fig. 7(b) that (15) is an upper bound only for higher numbers of samples, but there it is a tighter bound than the CLT-based bound (11); see Fig. 7(a).

3) Removing Excess Samples: If too many samples have been generated, then the excess samples can easily be removed using the following strategy: Sort the samples with respect to the value of the first coordinate. Then, half the necessary number of points to remove are deleted at the beginning and end of that sorted list of samples, respectively. Finally, the first coordinate is stretched to fully cover the unit cube again. This is similar to the process depicted in [1, Fig. 3(c) and (d)].

Removing samples equally on both ends, instead of simply removing them at the end, preserves the symmetry of the point set and also the first moment, which is zero due to the symmetry. However, configurations with an odd number of samples can only be reduced to other odd configurations then. The same applies to even configurations. For the enclosed hypercube method, an even configuration is produced if the sampling vector (8) has an even number of entries, i.e., if L_1 is even and therefore has no entry at zero. For the linear programming counter, an odd grid is produced when the integer vectors $\underline{z} \in \mathbb{Z}$ are offset by $\frac{1}{2}$.

4) Sample Library: Since sample computation using Frolov-like methods as proposed here is quite expensive in higher dimensions, it makes sense to calculate and save the samples in advance for each dimension in an odd and an even configuration, respectively. The CLT formula can be solved in closed form (12) and hence allows direct computation of a scaling factor such that at least the wanted number of samples is generated in any case and therefore helps to avoid "trial and error" in producing the desired number of samples. In the real-time application, the respective number of samples can then be extracted from this library as described in Section IV-C3.

V. Optimal Deterministic Gaussian Fibonacci Grids

At this point, it has been shown how to find a given number L of uniform Fibonacci samples $\mathbf{X}_{\mathrm{Fib}} \in \mathbb{R}^{D \times L}$ in the unit hypercube $\mathcal{I}$. Now, we will see how these can be transformed into Gaussian samples with various levels of sophistication.

A. Types of Transformations

The Fibonacci grid has the unique property that point collisions are avoided under anisotropic rescaling along certain directions. We begin with reiterating two common conditions on transformations before quantifying the "discrepancy-preserving" property that we need here.

1) Rigid transformations preserve angles and distances. They are in the class of linear transformations, where the determinant of the transformation matrix is either 1 or -1. Rigid transformations are translations, rotations, and reflections.

2) Conformal maps preserve angles locally. Everywhere, the respective local Jacobian matrices are orthogonal matrices multiplied by a scalar.

3) Discrepancy-preserving transformations preserve the uniformity of Fibonacci grids. Therefore, their Jacobian matrices must be orthogonal matrices multiplied with a diagonal matrix from left. That is, the right angles between the main axes are preserved.

In the following, we describe a discrepancy-preserving mapping that transforms the uniform density on $\mathcal{I}$ to arbitrary Gaussian densities.

B. Standard Normal Samples

First of all, we apply the usual inverse transform to obtain standard normal samples

$$[\mathbf{X}_{\mathrm{std}}]_{d,i} = \sqrt{2} \cdot \mathrm{erf}^{-1}(2 \cdot [\mathbf{X}_{\mathrm{Fib}}]_{d,i}) \quad ,$$

with the Gauss error function

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z \exp\{-t^2\} \, \mathrm{d}t \quad .$$

This transformation maps samples uniform in $\left[-\frac{1}{2}, \frac{1}{2}\right]^D$ to standard normal samples in $\mathbb{R}^D$. Refer to Fig. 4(f) for a visualization.

C. Variance Correction

Due to the discrete sample locations, the diagonal elements of the covariance matrix $\mathbf{C}_{\mathrm{std}}$ of the resulting point set $\mathbf{X}_{\mathrm{std}}$ are only approximately one, and the off-diagonal elements are only approximately zero. We can easily correct the former because rescaling along the coordinate axes, i.e., multiplication with a diagonal matrix, is a discrepancy-preserving transformation. So, we

calculate the variances

$$v_d = \frac{1}{L}\sum_{i=1}^{L}[\mathbf{X}_{\text{std}}]_{d,i}^2 \tag{16}$$

and correct the variance of the samples accordingly by rescaling the coordinates individually

$$[\mathbf{X}_{\text{stdD}}]_{d,i} = \frac{[\mathbf{X}_{\text{std}}]_{d,i}}{\sqrt{v_d}} \quad . \tag{17}$$

D. Arbitrary Gaussian Samples

Now we want to transform the standard normal samples $\mathbf{X}_{\text{stdD}}$ with a covariance of approximately $\mathbf{I}$ (the off-diagonals are not entirely zero) to a sample set with a given, arbitrary positive definite covariance $\mathbf{C} \in \mathbb{R}^{D\times D}$. Usually, this is done via the Cholesky decomposition $\mathbf{C} = \mathbf{L}\cdot\mathbf{L}^\top$ according to $\mathbf{X}_{\text{Gauss}} = \mathbf{L}\cdot\mathbf{X}_{\text{stdD}}$. However, this transformation is *not* discrepancy-preserving as the orthogonality of the D main axes is not being preserved; see Fig. 8(a). Therefore, we have to calculate the eigenvalue decomposition $\mathbf{V}\cdot\mathbf{D}\cdot\mathbf{V}^\top = \mathbf{C}$, with $\mathbf{V}$ orthogonal and $\mathbf{D}$ diagonal, and transform the samples according to

$$\mathbf{X}_{\text{Gauss1}} = \mathbf{V}\cdot\sqrt{\mathbf{D}}\cdot\mathbf{X}_{\text{stdD}} \quad . \tag{18}$$

Refer to Fig. 3(a) for a visual description of the transformation workflow. While Cholesky decomposition requires $\frac{1}{3}D^3$ floating point operations, eigenvalue decomposition is an iterative procedure with computational complexity $\mathcal{O}(D^3)$ per iteration step. A quick test in Matlab showed that the eigenvalue decomposition takes up to 70 times longer than the Cholesky decomposition; see Fig. 9. This is a small disadvantage that we have to accept when working with Fibonacci grids or other low-discrepancy point sets.

E. Fast Cholesky Covariance Correction

Because the off-diagonals of $\mathbf{C}_{\text{stdD}}$ are not exactly zero, $\mathbf{C}_{\text{Gauss1}}$ in (18) will not mach $\mathbf{C}$ exactly. We can correct this by using a transformation that is not discrepancy-preserving. As the amount of correction is rather small, the downside that the transformation is not discrepancy-preserving should have a rather small effect. We determine $\mathbf{C}_{\text{StdD}}$ and its Cholesky decomposition $\mathbf{L}_{\text{StdD}}$

$$\mathbf{C}_{\text{StdD}} = \frac{1}{L}\cdot\mathbf{X}_{\text{StdD}}\cdot\mathbf{X}_{\text{StdD}}^\top = \mathbf{L}_{\text{StdD}}\cdot\mathbf{L}_{\text{StdD}}^\top$$

and perform the correction

$$\mathbf{X}_{\text{Gauss}} = \mathbf{V}\cdot\sqrt{\mathbf{D}}\cdot\mathbf{L}_{\text{StdD}}^{-1}\cdot\mathbf{X}_{\text{StdD}} \quad ,$$

resulting in unscented samples $\mathbf{X}_{\text{Gauss}}$ with matching covariance.

F. Fast Eigenvalue Covariance Correction

Instead of the Cholesky decomposition of $\mathbf{C}_{\text{stdD}}$, we can also use its eigenvalue decomposition

$$\mathbf{C}_{\text{StdD}} = \mathbf{V}_{\text{StdD}}\cdot\mathbf{D}_{\text{StdD}}\cdot\mathbf{V}_{\text{StdD}}^\top$$

and obtain unscented samples $\mathbf{X}_{\text{Gauss}}$ via

$$\mathbf{X}_{\text{Gauss}} = \mathbf{V}\cdot\sqrt{\mathbf{D}}\cdot\mathbf{V}_{\text{StdD}}\cdot\sqrt{\mathbf{D}_{\text{StdD}}^{-1}}\cdot\mathbf{V}_{\text{StdD}}^\top\cdot\mathbf{X}_{\text{StdD}} \quad . \tag{19}$$

Again, this transformation is not discrepancy-preserving, but due to the rather small amount of correction necessary, this should not cause real problems.

G. Discrepancy-Preserving Covariance Correction

Now we will derive a different covariance correction method that is discrepancy-preserving. Recall that the problem with $\mathbf{C}_{\text{stdD}}$ from (17) was that the off-diagonals are not exactly zero. That is, there are small correlations present that are expressed in inequality of eigenvalues, i.e., the Gaussian contour map looks like an ellipse (ellipsoid, hyperellipsoid) instead of a circle (sphere, hypersphere)—where the principal axes of the ellipse (ellipsoid, hyperellipsoid), i.e., the eigenvectors of the covariance matrix, do not coincide with the principal axes of the coordinate system. If we only manage to deform that covariance in a discrepancy-preserving way such that the eigenvalues match the wanted eigenvalues in $\mathbf{D}$, then we can easily find an appropriate rotation that puts the covariance in the right orientation subsequently.

Therefore, we have to find D deformations along the coordinate axes, collected in the vector $\underline{a} \in \mathbb{R}^D$, such that the eigenvalues of

$$\mathbf{C}_{\text{std},a} = \underline{a}\odot\mathbf{C}_{\text{std}}\odot\underline{a}^\top \quad , \tag{20}$$

where $\odot$ is the pointwise product, i.e., the Hadamard product, match the targeted eigenvalues in $\mathbf{D}$.

Note that eigenvalues must be compared with a Wasserstein or earth mover's distance, i.e., the eigenvalues from both sets have to be associated appropriately. For real eigenvalues, this can be done by sorting both sets of eigenvalues [51, Sec. 3].

The search for $\underline{a}$ is now a gradient-based nonlinear optimization problem with D variables. Derivatives of eigenvalues can be calculated analytically [52, Eq. (11)], [53, Eq. (5)]

$$\frac{\partial\underline{\lambda}}{\partial a} = \mathbf{V}^\top\cdot\frac{\partial\mathbf{C}}{\partial a}\cdot\mathbf{V} \quad ,$$

where $\mathbf{V}$ is the normalized right eigenvector matrix of symmetric matrix $\mathbf{C}$, and $\underline{\lambda}$ the corresponding eigenvalues. With the optimal deformation vector $\hat{\underline{a}}$ obtained by nonlinear optimization, and the eigenvector matrix $\mathbf{V}_{\text{std},\hat{a}}$ of $\mathbf{C}_{\text{std},\hat{a}}$, we get the transformed Gaussian samples

$$\mathbf{X}_{\text{Gauss}} = \mathbf{V}\cdot\mathbf{V}_{\text{std},\hat{a}}^\top\cdot(\hat{\underline{a}}\odot\mathbf{X}_{\text{std}}) \quad ,$$

where $\mathbf{V}$ is again the eigenvector matrix of the wanted covariance $\mathbf{C}$.

The transformation (20) can only increase the ratio between eigenvalues. Therefore, if some eigenvalues of $\mathbf{C}$ are equal or nearly equal, the optimization cannot match the eigenvalues perfectly, and some residual distance to the desired eigenvalues remains. Again, this can be fully matched via a nondiscrepancy-preserving transformation

$$\mathbf{X}_{\text{Gauss}} = \mathbf{V} \cdot \left(\underline{b} \odot \mathbf{V}_{\text{std},\hat{a}}^{\top} \cdot \left(\hat{\underline{a}} \odot \mathbf{X}_{\text{std}} \right) \right) \ , \qquad (21)$$

with nondiscrepancy-preserving deformation vector $\underline{b}$

$$\underline{b} = \sqrt{\operatorname{diag}(\mathbf{D}) \oslash \operatorname{diag}(\mathbf{V}_{\text{std},\hat{a}}^{\top} \mathbf{C}_{\text{std},\hat{a}} \mathbf{V}_{\text{std},\hat{a}})} \ ,$$

where $\oslash$ denotes element-wise division. The amount of nondiscrepancy-preserving deformation encoded in $\underline{b}$ that is needed to match the covariance exactly is rather small, so point collisions will not occur.

VI. EVALUATION

The application we had in mind when developing this method are the "unscented transform" for nonlinear Gaussian filtering. Approximating first and second moments, i.e., means and covariances of nonlinear functions of Gaussian random variables, facilitates probabilistic Kalman filtering based on stochastic linearization, i.e., the various forms of LRKFs and also GPFs. Low-discrepancy sequences like the proposed Fibonacci grid and other deterministic sampling methods like LCD allow for better accuracy by using more samples than only $L = 2 \cdot D$ as in the standard unscented transform. In [1, Sec. IV], we used one specific nonlinear function to compare various deterministic sampling methods. After showing a similar, simple evaluation, we evaluate Gaussian sample sets based on a linear space of nonlinear functions.

A. Simple Evaluation

To evaluate different sampling methods, we define a nonlinear system model with additive noise

$$\boldsymbol{y} = \left\| \underline{\boldsymbol{x}} \right\|_3^3 + \boldsymbol{v} \ , \qquad \mathrm{E}\{\boldsymbol{v}\} = 0 \ , \qquad \mathrm{E}\left\{\boldsymbol{v}^2\right\} = 30^2 \ .$$

Given prior moments

$$\underline{x}^{\mathrm{p}} = \begin{bmatrix} 2 \\ -2 \end{bmatrix} \ , \qquad \mathbf{C}^{\mathrm{p}} = \begin{bmatrix} 1^2 & 0 \\ 0 & 5^2 \end{bmatrix}$$

and the measurement

$$\hat{y} = 30 \ ,$$

we compute a UKF as well as a Gaussian Filter update step with various methods for Gaussian sampling.

For the UKF case, we take Gaussian samples $\hat{\underline{x}}_i, \ i \in \{1, \ldots, L\}$ from the prior Gaussian $\mathcal{N}(x; \underline{x}^{\mathrm{p}}, \mathbf{C}^{\mathrm{p}})$ using the respective sampling method, insert them into the measurement equation, add samples from the measurement noise, compute the empirical moments in the joint state and measurement space

$$\underline{z}^{\mathrm{p}} = \begin{bmatrix} \underline{x}^{\mathrm{p}} \\ y^{\mathrm{p}} \end{bmatrix} \ , \qquad \mathbf{C}^{z} = \begin{bmatrix} \mathbf{C}^{\mathrm{p}} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{C}_{yy} \end{bmatrix} \ ,$$

and obtain the posterior estimate

$$\underline{x}^{\mathrm{e}} = \underline{x}^{\mathrm{p}} + \mathbf{C}_{xy} \mathbf{C}_{yy}^{-1} \left(\hat{y} - y^{\mathrm{p}} \right) \ .$$

Refer to Fig. 10(a) for a quantitative evaluation of the root mean square error (RMSE) of the estimated posterior. Note that the real bottleneck here is the second Gaussian assumption, i.e., the statistical linearization between state space $\underline{x}$ and measurement y.

For GPF-style filtering, we take again L Gaussian samples $\hat{\underline{x}}_i$ from the prior Gaussian $\mathcal{N}(x; \underline{x}^{\mathrm{p}}, \mathbf{C}^{\mathrm{p}})$ using the respective sampling method and apply individual sample weights w_i according to the likelihood function value at the respective sample

$$w_i \propto \mathcal{N}\left(\hat{y}; \ \left\| \hat{\underline{x}}_i \right\|_3^3, \ \mathbf{C}^{v} \right) \ , \qquad \sum_{i=1}^{L} w_i = 1.$$

The posterior mean is then approximated as the empirical average of the weighted samples. Refer to Fig. 10(b) for a quantitative evaluation of the RMSE of the estimated posterior.

B. Measure of Quality

In this section, we will define a general quality measure for Gaussian samples. It describes how well the expected values of Gaussian random variables are estimated using the respective sample sets. Thereby, instead of focusing on one specific nonlinear function, e.g., $\|\underline{\boldsymbol{x}}\|_3^3$

$$\Theta = \left| \mathrm{E}\left\{ \left\| \underline{\boldsymbol{x}} \right\|_3^3 \right\} - \hat{\mathrm{E}}\left\{ \left\| \underline{\boldsymbol{x}} \right\|_3^3 \right\} \right| \ ,$$

where $\mathrm{E}\{\cdot\}$ is the true expected value, $\hat{\mathrm{E}}\{\cdot\}$ is the sample approximation, and Θ is an optimality measure for the employed samples, we take a broad class of smooth nonlinear functions into account.

1) Harmonic Expectations: The expectation value of the function $g(\cdot)$ of a random variable $\boldsymbol{x}$ is

$$\mathrm{E}\left\{ g(\underline{\boldsymbol{x}}) \right\} = \int_{\mathbb{R}^D} g(\underline{x}) \cdot f(\underline{x}) \, \mathrm{d}\underline{x},$$

where $\underline{\boldsymbol{x}} \sim f(\underline{x})$, and $g(\cdot)$ is a smooth nonlinear function. Numerical approximation of this integral with unweighted samples $\hat{\underline{x}}_i, \ i \in \{1, \ldots, L\}$ goes like

$$\begin{aligned} \hat{\mathrm{E}}\left\{ g(\underline{\boldsymbol{x}}) \right\} &= \int_{\mathbb{R}^D} g(\underline{x}) \cdot \hat{f}(\underline{x}) \, \mathrm{d}\underline{x} \\ &= \int_{\mathbb{R}^D} g(\underline{x}) \cdot \frac{1}{L} \sum_{i=1}^{L} \delta(\underline{x} - \hat{\underline{x}}_i) \, \mathrm{d}\underline{x} \\ &= \frac{1}{L} \sum_{i=1}^{L} g(\hat{\underline{x}}_i) \ . \end{aligned}$$

To obtain a representative comparison, we take into account all possible smooth functions $g(\underline{x})$ by using a Fourier basis

$$\begin{aligned} g_{\underline{t}}(\underline{x}) &= \exp\{\mathrm{i}\,\underline{t}\cdot\underline{x}\} \\ &= \cos(\underline{t}\cdot\underline{x}) + \mathrm{i}\sin(\underline{t}\cdot\underline{x}) \ . \end{aligned} \tag{22}$$

The true expected value of a harmonic function of $\boldsymbol{x}$,

$$\begin{aligned} \mathrm{E}\{g_{\underline{t}}(\underline{\boldsymbol{x}})\} &= \int_{\mathbb{R}^D} g_{\underline{t}}(\underline{x})\cdot f(\underline{x})\,\mathrm{d}\underline{x} \\ &= \int_{\mathbb{R}^D} \exp\{\mathrm{i}\,\underline{t}\cdot\underline{x}\}\cdot f(\underline{x})\,\mathrm{d}\underline{x} \\ &= F(\underline{t}) \ , \end{aligned}$$

is the characteristic function $F(\underline{t})$ of the density $f(\underline{x})$. Furthermore, the approximated expectation

$$\begin{aligned} \hat{\mathrm{E}}\{g_{\underline{t}}(\underline{\boldsymbol{x}})\} &= \frac{1}{L}\sum_{i=1}^{L} g_{\underline{t}}(\underline{\hat{x}}_i) \\ &= \frac{1}{L}\sum_{i=1}^{L} \exp\{\mathrm{i}\,\underline{t}\cdot\underline{\hat{x}}_i\} \\ &= \hat{F}(\underline{t}) \end{aligned}$$

is the characteristic function of $\hat{f}(\underline{x})$.

2) Distance Measure: Now by averaging over all spatial frequencies in domain $\mathcal{T}$

$$\begin{aligned} \Theta^2 &= \int_{\mathcal{T}} \left|\mathrm{E}\{g_{\underline{t}}(\underline{\boldsymbol{x}})\} - \hat{\mathrm{E}}\{g_{\underline{t}}(\underline{\boldsymbol{x}})\}\right|^2 \mathrm{d}\underline{t} \\ &= \int_{\mathcal{T}} \left|F(\underline{t}) - \hat{F}(\underline{t})\right|^2 \mathrm{d}\underline{t} \ , \end{aligned} \tag{23}$$

we obtain distance measure Θ that quantifies the approximation error of expectation values in the function space of band-limited nonlinear functions. In other words, Θ quantifies the average accuracy of an expectation value estimate of a sample set for band-limited functions. Note that (23) computes the square sum of the real and imaginary parts from (22), representing cosine and sine parts appropriately, according to

$$|a + \mathrm{i}b|^2 = (a+ib)(a-ib) = a^2 + b^2 \ .$$

We consider spatial frequencies $\underline{t}$ in a certain domain $\mathcal{T}$,

$$\mathcal{T} = \left\{ \underline{t} \;\middle|\; \bigcap_{d=1}^{D} t^{(d)} \in [-\tau, \tau] \right\} \ , \tag{24}$$

around the origin, i.e., we focus on all sufficiently smooth functions $g(\underline{x})$, where the Fourier transform $G(\underline{x})$ does not have significant energy outside $\mathcal{T}$. From the Nyquist–Shannon sampling theorem, we can try to derive a suitable bound for τ. Given L optimal deterministic uniform samples on $[0, 1]$, their spacing is about L^{-1}. When transforming these to arbitrary densities by inverse transform sampling, the spacing of the transformed samples at the point of maximum density is $(L\cdot f_{\mathrm{mode}})^{-1}$, where f_{mode} is the maximum derivative of the cumulative density, i.e., the density value at the mode. Therefore, the "sampling rate" is $L\cdot f_{\mathrm{mode}}$ at the mode, and lower elsewhere. Thus, according to Nyquist–Shannon, we may conclude that the maximum spatial frequency that can be represented by these samples is $\tau = L\cdot f_{\mathrm{mode}}/2$. In higher dimensions, the hypercubic cell representing one sample has volume L^{-1}. Using the inverse function theorem, the volume of this cell is transformed by the inverse of the Jacobian determinant $1/\det(\mathbf{J}(\underline{x}))$ of the mapping function that maps $f(\underline{x})$ to a uniform density. As a result, the new side length is $(L\cdot\det(\mathbf{J}(\underline{x})))^{-1/D}$, and we can choose $\tau = (L\cdot\det(\mathbf{J}(\underline{x})))^{1/D}/2$.

In our application, $f(\underline{x})$ is a Gaussian density

$$\begin{aligned} f(x) &= \mathcal{N}\left(\underline{x};\,\underline{\mu},\,\mathbf{C}\right) \\ &= \frac{1}{\sqrt{|2\pi\,\mathbf{C}|}}\exp\left\{-\frac{1}{2}\left(\underline{x}-\underline{\mu}\right)^{\top}\mathbf{C}^{-1}\left(\underline{x}-\underline{\mu}\right)\right\} \ , \end{aligned}$$

with $\underline{\mu}\in\mathbb{R}^D$, and positive definite component covariance matrix $\mathbf{C}\in\mathbb{R}^{D\times D}$. The characteristic function of $f(\underline{x})$ is

$$F(\underline{t}) = \exp\left\{\mathrm{i}\,\underline{\mu}\cdot\underline{t} - \frac{1}{2}\underline{t}^{\top}\mathbf{C}\,\underline{t}\right\} \ .$$

To simplify the solution of (23), we choose

$$\begin{aligned} \underline{\mu} &= \underline{0} \ , \\ \mathbf{C}_k &= \mathrm{diag}(\sigma_1^2,\,\sigma_2^2,\,\ldots,\,\sigma_D^2) \end{aligned} \tag{25}$$

without restriction of generality, as the basis system and origin of the coordinate system can always be chosen appropriately. We obtain

$$\begin{aligned} \Theta^2 &= \int_{\mathcal{T}} \left|\exp\left\{-\frac{1}{2}\underline{t}^{\top}\mathbf{C}\,\underline{t}\right\} - \frac{1}{L}\sum_{i=1}^{L}\exp\{\mathrm{i}\,\underline{t}\cdot\underline{\hat{x}}_i\}\right|^2 \mathrm{d}\underline{t} \\ &= \Theta_{xx} - 2\Theta_{xy} + \Theta_{yy} \ , \end{aligned}$$

where

$$\begin{aligned} \Theta_{xx} &= \int_{\mathcal{T}} \exp\{-\underline{t}^{\top}\mathbf{C}\,\underline{t}\}\,\mathrm{d}\underline{t} \ , \\ \Theta_{xy} &= \frac{1}{L}\int_{\mathcal{T}} \exp\left\{-\frac{1}{2}\underline{t}^{\top}\mathbf{C}\,\underline{t}\right\}\cdot\sum_{i=1}^{L}\cos(\underline{t}\cdot\underline{\hat{x}}_i)\,\mathrm{d}\underline{t} \ , \\ \Theta_{yy} &= \frac{1}{L^2}\int_{\mathcal{T}}\sum_{i=1}^{L}\sum_{j=1}^{L}\exp\left\{\mathrm{i}\,\underline{t}\cdot(\underline{\hat{x}}_i - \underline{\hat{x}}_j)\right\}\mathrm{d}\underline{t} \ . \end{aligned}$$

For diagonal covariances (25) and $\mathcal{T}$ according to (24), this can be simplified to

$$\begin{aligned} \Theta_{xx} &= \pi^{D/2}\prod_{d=1}^{D}\frac{\mathrm{erf}(\tau\cdot\sigma_d)}{\sigma_d} \ , \\ \Theta_{xy} &= \frac{(2\pi)^{D/2}}{L}\sum_{i=1}^{L}\prod_{d=1}^{D}\frac{1}{\sigma_d}\exp\left\{-\frac{1}{2}\frac{x_{i,d}^2}{\sigma_d^2}\right\} \ , \end{aligned}$$

$$\cdot \operatorname{real}\left\{\operatorname{erf}\left(\frac{\tau\sigma_d^2 + \mathrm{i}\cdot x_{i,d}}{\sqrt{2}\,\sigma_d}\right)\right\} \quad ,$$

$$\Theta_{yy} = \frac{2^D}{L^2}\sum_{i=1}^{L}\sum_{j=1}^{L}\prod_{d=1}^{D}\frac{\sin\big(\big(x_{i,d} - x_{j,d}\big)\cdot\tau\big)}{x_{i,d} - x_{j,d}} \quad .$$

The integration domain could be chosen as

$$\tau = \sqrt{\frac{\pi}{2}}\cdot\left(L\cdot\prod_{d=1}^{D}\sigma_d\right)^{1/D}$$

according to considerations relating to the sampling theorem. However, we choose τ constant to make the optimality measure better comparable over different L.

C. Gaussian Sampling Comparison

Now we will compare various Gaussian sampling methods for their suitability for numerical approximation of expectation values of band-limited nonlinear functions of Gaussian densities. The state-of-the-art we compare against includes LCD samples from the nonlinear estimation toolbox [54] and the Halton sequence [55], a well-known low-discrepancy sequence, that is transformed from uniform to Gaussian just as explained for the proposed Fibonacci grids in (16).

In Fig. 11(a), we show how well the different methods can approximate the three-dimensional standard normal density, i.e., the isotropic case. We can see there that the LCD method (yellow) generally provides the best results, closely followed by the Fibonacci methods with covariance matching (cyan, green). Note that $L = 10$ LCD or Fibonacci samples with moment correction are as effective as $L = 200$ samples from the Halton sequence, or more than 1 000 iid samples.

Fig. 11(b) shows the same for an anisotropic Gaussian with covariance $\mathbf{C} = \operatorname{diag}([1, 0.1, 0.01])^2$. Here we see that the Fibonacci and Halton samples generally provide the best results, closely followed by LCD samples. This difference compared to the standard normally distributed case is because these LCD samples are produced by anisotropic transformation of standard normal ones, where some optimality is lost. Fibonacci grids, however, can be rescaled without any quality loss.

Fig. 11(c) visualizes the computational effort to compute $L = 1\,000$ LCD samples and Fibonacci samples computed via the enclosing hypercube enumeration from Section IV-A and linear programming enumeration from Section IV-B, respectively, for various dimensions. For dimensions $D < 6$, the enclosing hypercube method is fastest; for dimensions $6 \leq D \leq 15$, the linear programming counter; and for $D > 15$, the LCD samples. Note that for all three methods, samples can be computed beforehand and stored for later real-time use. LCD samples have to be generated for every desired dimension D and number of samples L, respectively, while Fibonacci grids have to be generated separately for every dimension D only, because subsets of Fibonacci grids can easily be used; see Section IV-C.

Fig. 11(d) shows the overall fastest sampling method out of LCD, Fibonacci linear programming, and Fibonacci hypercube for various dimensions and various numbers of samples. Again, we find that the Fibonacci enclosing hypercube method is fastest for smaller dimensions, and LCD for higher dimensions, and Fibonacci linear programming in between. Note that for $D = 7$ and $D = 10$, where $2D + 1$ is not prime, suitable generalized Fibonacci matrices are not yet known; therefore, LCD is available only. Note also that LCD with symmetric samples from the nonlinear estimation toolbox [54] requires $L \geq 2D$.

VII. CONCLUSION

We presented a new enumeration method for Fibonacci grids that is based on linear programming. It is faster than the existing enclosing hypercube method for dimensions $D \geq 6$. Furthermore, we introduced different methods for covariance correction, including a simple and fast Cholesky correction and a slower discrepancy-preserving method. We have also investigated the possible range of enumerated points given a certain scaling factor. The evaluation that was performed for a broad function class suggests that Fibonacci samples, together with the state-of-the-art LCD samples, yield the best approximations of nonlinear Gaussian expectations.

In the future, we will look for generalized Fibonacci matrices for dimensions where $(2D + 1)$ is prime. We will also look for lattice rule versions of the generalized Fibonacci grid, as they are faster to compute, and the number of resulting samples is exactly known beforehand.

The authors acknowledge support by the state of Baden–Württemberg through bwHPC.

REFERENCES

[1] D. Frisch and U. D. Hanebeck
"Deterministic Gaussian sampling with generalized fibonacci grids,"
in *Proc. 24th Int. Conf. Inf. Fusion*, 2021, pp. 1–8.

[2] J. M. Hammersley and K. W. Morton
"A new Monte Carlo technique: Antithetic variates,"
Math. Proc. Cambridge Philos. Soc., vol. 52, no. 3, pp. 449–475, 1956.

[3] R. Y. Rubinstein and R. Marcus
"Efficiency of multivariate control variates in Monte Carlo simulation,"
Operations Res., vol. 33, no. 3, pp. 661–677, 1985.

[4] S. T. Tokdar and R. E. Kass
"Importance sampling: A review,"
WIREs Comput. Statist., vol. 2, no. 1, pp. 54–60, 2010.

[5] V. L. Parsons
Stratified Sampling. New York, NY: Wiley, 2017, pp. 1–11.

[6] I. H. Sloan
"Lattice methods for multiple integration,"
J. Comput. Appl. Math., vol. 12, pp. 131–143, 1985.

[7] J. Dick, F. Y. Kuo, and I. H. Sloan
"High-dimensional integration: The quasi-Monte Carlo way,"
Acta Numerica, vol. 22, pp. 133–288, 2013.

[8] S. J. Julier and J. K. Uhlmann
"New extension of the Kalman filter to nonlinear systems,"
in *Proc. Signal Process., Sensor Fusion, Target Recognit. VI*, 1997, pp. 182–193.

[9] S. Julier and J. Uhlmann
"Unscented filtering and nonlinear estimation,"
Proc. IEEE, vol. 92, no. 3, pp. 401–422, Mar. 2004.

[10] M. Roth, G. Hendeby, and F. Gustafsson
"Nonlinear Kalman filters explained: A tutorial on moment computations and sigma point methods,"
J. Adv. Inf. Fusion, vol. 11, no. 1, pp. 47–70, 2016.

[11] U. D. Hanebeck and V. Klumpp
"Localized cumulative distributions and a multivariate generalization of the Cramér-von Mises distance,"
in *Proc. IEEE Int. Conf. Multisensor Fusion Integration Intell. Syst.*, 2008, pp. 33–39.

[12] J. Steinbring and U. D. Hanebeck
"LRKF revisited: The smart sampling Kalman filter (S2KF),"
J. Adv. Inf. Fusion, vol. 9, no. 2, pp. 106–123, Dec. 2014. Available: https://confcats_isif.s3.amazonaws.com/web-files/journals/entries/441_1_art_11_17020.pdf

[13] U. D. Hanebeck
"Deterministic sampling of multivariate densities based on projected cumulative distributions,"
in *Proc. 54th Annu. Conf. Inf. Sci. Syst.*, 2020, pp. 1–6.

[14] D. Prossel and U. D. Hanebeck
"Dirac mixture reduction using Wasserstein distances on projected cumulative distributions,"
in *Proc. 25th Int. Conf. Inf. Fusion*, 2022, pp. 1–8.

[15] J. Steinbring, M. Pander, and U. D. Hanebeck
"The smart sampling Kalman filter with symmetric samples,"
J. Adv. Inf. Fusion, vol. 11, no. 1, pp. 71–90, Jun. 2016.

[16] G. E. P. Box and M. E. Muller
"A note on the generation of random normal deviates,"
Ann. Math. Statist., vol. 29, no. 2, pp. 610–611, 1958.

[17] W. Feller
An Introduction to Probability Theory and Its Applications, vol. 1, 3rd ed. New York, NY: Wiley. [Online]. Available: https://www.wiley.com/en-us/An+Introduction+to+Probability+Theory+and+Its+Applications%2C+Volume+1%2C+3rd+Edition-p-9780471257080

[18] R. E. Caflisch
"Monte Carlo and quasi-Monte Carlo methods,"
Acta Numerica, vol. 7, pp. 1–49, Jan. 1998, . Available: https://www.cambridge.org/core/journals/acta-numerica/article/monte-carlo-and-quasimonte-carlo-methods/FE7C779B350CFEA45DB2A4CCB2DA9B5C#

[19] N. Steen, G. Byrne, and E. Gelbard
"Gaussian quadratures for the integrals $\int_0^\infty \exp(-x^2) f(x)\,\mathrm{d}x$ and $\int_0^b \exp(-x^2) f(x)\,\mathrm{d}x$,"
Math. Computation, vol. 23, no. 107, pp. 661–671, 1969.

[20] K. Ito and K. Xiong
"Gaussian filters for nonlinear filtering problems,"
IEEE Trans. Autom. Control, vol. 45, no. 5, pp. 910–927, May 2000.

[21] P. Jäckel
A Note on Multivariate Gauss–Hermite Quadrature.
London: ABN-Amro. Re, 2005.

[22] I. Arasaratnam, S. Haykin, and R. J. Elliott
"Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature,"
Proc. IEEE, vol. 95, no. 5, pp. 953–977, May 2007.

[23] M. F. Huber and U. D. Hanebeck
"Gaussian filter based on deterministic sampling for high quality nonlinear estimation,"
in *Proc. 17th IFAC World Congr.*, vol. 41, no. 2, pp. 13527–13532, 2008.

[24] S. J. Julier
"The scaled unscented transformation,"
in *Proc. Amer. Control Conf.* 2002, pp. 4555–4559.

[25] I. Arasaratnam and S. Haykin
"Cubature Kalman filters,"
IEEE Trans. Autom. Control, vol. 54, no. 6, pp. 1254–1269, Jun. 2009.

[26] B. Jia, M. Xin, and Y. Cheng
"High-degree cubature Kalman filter,"
Automatica, vol. 49, no. 2, pp. 510–518, 2013.

[27] R. Cools and P. Rabinowitz
"Monomial cubature rules since "stroud": A compilation,"
J. Comput. Appl. Math., vol. 48, no. 3, pp. 309–326, 1993. Available: https://www.sciencedirect.com/science/article/pii/0377042793900279

[28] G. Phillips
"A survey of one-dimensional and multidimensional numerical integration,"
Comput. Phys. Commun., vol. 20, no. 1, pp. 17–27, 1980. Available: https://www.sciencedirect.com/science/article/pii/0010465580901022

[29] S. Julier and J. Uhlmann
"Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations,"
in *Proc. Amer. Control Conf.*, 2002, pp. 887–892.

[30] K. G. Papakonstantinou, M. Amir, and G. P. Warn
"A scaled spherical simplex filter (S3F) with a decreased $n+2$ sigma points set size and equivalent $2n+1$ unscented Kalman filter (UKF) accuracy,"
Mech. Syst. Signal Process., vol. 163, 2022, Art. no. 107433. Available: https://www.sciencedirect.com/science/article/pii/S0888327020308190

[31] J. Kotecha and P. Djuric
"Gaussian particle filtering,"
IEEE Trans. Signal Process., vol. 51, no. 10, pp. 2592–2601, Oct. 2003.

[32] V. Elvira, L. Martino, and P. Closas
"Importance Gaussian quadrature,"
IEEE Trans. Signal Process., vol. 69, pp. 474–488, 2021.

[33] U. D. Hanebeck Closas
"PGF 42: Progressive Gaussian filtering with a twist,"
in *Proc. 16th Int. Conf. Inf. Fusion*, 2013, pp. 1103–1110.

[34] J. Steinbring and U. D. Hanebeck
"Progressive Gaussian filtering using explicit likelihoods,"
in *Proc. 17th Int. Conf. Inf. Fusion (FUSION)*, 2014, pp. 1–8.

[35] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp
"A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,"
IEEE Trans. Signal Process., vol. 50, no. 2, pp. 174–188, Feb. 2002.

[36] U. D. Hanebeck, M. F. Huber, and V. Klumpp
"Dirac mixture approximation of multivariate Gaussian densities,"
in *Proc. IEEE Conf. Decis. Control*, 2009, pp. 3851–3858.

[37] J. Steinbring and U. D. Hanebeck
"S2KF: The smart sampling Kalman filter,"
in *Proc. 16th Int. Conf. Inf. Fusion*, 2013, pp. 2089–2096.

[38] D. Frisch and U. D. Hanebeck
"Efficient deterministic conditional sampling of multivariate gaussian densities,"
in *Proc. IEEE Int. Conf. Multisensor Fusion Integration Intell. Syst.*, 2020, pp. 75–81.

[39] D. Guo and X. Wang
"Quasi-Monte carlo filtering in nonlinear dynamic systems,"

IEEE Trans. Signal Process., vol. 54, no. 6, pp. 2087–2098, Jun. 2006.

[40] A. Rahimnejad, S. A. Gadsden, and M. Al-Shabi
"Lattice Kalman filters,"
IEEE Signal Process. Lett., vol. 28, pp. 1355–1359, 2021.

[41] C. Kacwin, J. Oettershagen, M. Ullrich, and T. Ullrich
"Numerical performance of optimized Frolov lattices in tensor product reproducing Kernel Sobolev spaces,"
Found. Comput. Math., vol. 21, no. 3, pp. 849–889, Jun. 2021.

[42] R. J. Purser
"Generalized Fibonacci grids; A new class of structured, smoothly adaptive multi-dimensional computational lattices" Office note (National Centers for Environmental Prediction (U.S.)), vol. 455, pp. 1–38, May 2008.

[43] D. Frisch and U. D. Hanebeck
"Rejection sampling from arbitrary multivariate distributions using generalized Fibonacci lattices,"
in *Proc. 25th Int. Conf. Inf. Fusion*, 2022, pp. 1–7.

[44] S. K. Zaremba
"A remarkable lattice generated by Fibonacci numbers,"
Fibonacci Quart., vol. 8, no. 2, pp. 185–198, 1970.

[45] H. Niederreiter
et al., "Integration of nonperiodic functions of two variables by Fibonacci lattice rules,"
J. Comput. Appl. Math., vol. 51, no. 1, pp. 57–70, 1994.

[46] R. L. Graham
Concrete Mathematics: A Foundation for Computer Science. Reading, MA, USA: Addison-Wesley, 1994.

[47] H. W. Gould
"A history of the Fibonacci Q-matrix and a higher-dimensional problem,"
Fibonacci Quart., vol. 19, no. 3, pp. 250–257, 1981.

[48] A. Makhorin
"GLPK (GNU linear programming kit)," [Online]. Available: http://www.gnu.org/software/glpk/glpk.html

[49] J. Dick and F. Pillichshammer
Explicit Constructions of Point Sets and Sequences with Low Discrepancy. Berlin, Germany: De Gruyter, 2014, pp. 63–86.

[50] D. Bilyk
"Discrepancy theory and harmonic analysis,"
Uniform Distribution and Quasi-Monte Carlo Methods: Discrepancy, Integration and Applications. Berlin, Boston: De Gruyter, 2014, pp. 45–62.

[51] E. Levina and P. Bickel
"The Earth Mover's distance is the mallows distance: Some insights from statistics,"
in *Proc. Eighth IEEE Int. Conf. Comput. Vision*, 2001, vol. 2, Jul. 2001, pp. 251–256.

[52] D. V. Murthy and R. T. Haftka
"Derivatives of eigenvalues and eigenvectors of a general complex matrix,"
Int. J. Numer. Methods Eng., vol. 26, no. 2, pp. 293–311, 1988, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620260202. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620260202

[53] P. Lancaster
"On eigenvalues of matrices dependent on a parameter,"
Numerische Mathematik, vol. 6, no. 1, pp. 377–387, Dec 1964. [Online]. Available: https://doi.org/10.1007/BF01386087

[54] J. Steinbring
"Nonlinear estimation toolbox," [Online]. Available: https://bitbucket.org/nonlinearestimation/toolbox

[55] J. H. Halton
"On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals,"
Numerische Mathematik, vol. 2, no. 1, pp. 84–90, 1960.

Daniel Frisch received the Master's degree in electrical engineering from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2017. Afterwards, he was a Ph.D. student at the Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Karlsruhe Institute of Technology (KIT), Germany. His research interests include deterministic sampling, nonlinear filtering, and multilateration.

Uwe D. Hanebeck received the Ph.D. and Habilitation degrees in electrical engineering from the Technical University in Munich, München, Germany, in 1997 and 2003, respectively. He is a Chaired Professor of Computer Science with the Karlsruhe Institute of Technology (KIT), Germany, and the Director of the Intelligent Sensor Actuator-Systems Laboratory, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. His research interests are in the areas of information fusion, nonlinear state estimation, stochastic modeling, system identification, and control with a strong emphasis on theory-driven approaches based on stochastic system theory and uncertainty models. He is an author and coauthor of more than 550 publications in various high-ranking journals and conferences and is an IEEE Fellow.

Repeated Filtering for Smoothing Particle Filters

STEPHEN L. ANDERSON
LAWRENCE D. STONE
SIMON MASKELL

This paper presents the repeated filtering method for finding a smoothed, Bayesian estimate of the path of a stochastic process over a time interval [0, *T*] when one has used a particle filter to estimate the state of the process. It provides good resolution over [0, *T*], is easy to implement, and can be used with any sequential importance resampling particle filter regardless of the probabilistic model employed by the stochastic process. Repeated filtering is general, powerful, and simple. It does not require the restrictive assumptions or complex calculations of other methods. It is suitable for real-time operational use in complex situations. We demonstrate the method on two single-target tracking examples. The second of these tracking examples is very difficult to solve by any other method known to us. We then apply repeated filtering to a standard nonlinear time series model that has been used extensively for testing numerical filtering techniques. To further illustrate the power of repeated filtering, we show how adding reflecting boundaries to this time series creates a process that is difficult to smooth with existing techniques but simple with repeated filtering.

Manuscript received October 6, 2022; revised March 22, 2023; released for publication May 29, 2023.

Associate Editor: Florian Meyer.

S. L. Anderson and L. D. Stone are with Metron, Reston VA, USA (e-mail: anderson@metsci.com; stone@metsci.com).
S. Maskell is with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, UK (e-mail: s.maskell@liverpool.ac.uk).

1557-6418/23/$17.00

I. INTRODUCTION

Particle filters are powerful and general tools for performing nonlinear, non-Gaussian filtering. For target tracking, they provide an estimate of the distribution on target state at the time of the last measurement. However, it is often desirable to compute the posterior distribution on the target's path over an interval of time $[0, T]$ given the measurements received in that interval. The process of computing this distribution is called fixed interval smoothing.

We present the repeated filtering method for smoothing in the context of surveillance and tracking, but the method is applicable to very general situations where one can use a sequential importance resampling (SIR) particle filter to estimate the history of the state of a stochastic system. To illustrate this, we apply the repeated filtering method to smooth the nonlinear time series analyzed in Example 1 of [5]. According to [5], this series has been used extensively for testing numerical filtering techniques. In the final example, we add reflecting boundaries to this time series. We show in Section IV-D that this process is difficult to smooth using the methods of [5], but simple with repeated filtering.

Repeated filtering is conceptually simple. Once one has developed a particle filter for the problem of interest, they have done the hard part. Repeated filtering proceeds as follows. Run the particle filter on the measurements received in $[0, T]$ while preserving the path histories of the particles. Choose a smoothed path from the filtered result at the end of the time interval $[0, T]$. Repeat the filtering process using the same measurements as in the first run of the filter but using independent random numbers to generate the particle paths. Choose a smoothed path as before. Repeat this process to obtain M independent draws from the posterior distribution on the paths of the process, given the measurements received in $[0, T]$. This produces a discrete sample path approximation of the posterior distribution.

Repeated filtering is simple and general. It can be incorporated into operational systems and used by operators who are not experts in tracking or data fusion. Many operational problems require motion models that are not Markovian or do not have a closed-form transition function as required by other particle filter smoothing methods. Repeated filtering produces smoothed paths, not just smoothed marginal distributions at discrete times, as many smoothing techniques do. Moreover, repeated filtering can be applied to both discrete and continuous time motion models. For continuous-time models, the smooth paths are continuous time paths.

Despite the conceptual simplicity of repeated filtering, we have not been able to find a reference to it. The first two authors spent over a year trying unsuccessfully to solve a smoothing problem conceptually similar to the surveillance problem in Example 2. The existing methods for smoothing particle filters, which are referenced

below, require assumptions that do not fit this problem. We tried a method involving Markov Chain Monte Carlo (MCMC) sampling, which was technically correct but computationally complex and delicate. In addition, it was unstable, producing qualitatively different results from different starting paths.

Perhaps by adjusting some of the tuning parameters of the MCMC, such as the acceptance probability or the proposal distribution, we could have made the MCMC method work. However, we decided that, even if we could make the method work, it would not be suitable for an operational system. By contrast, the repeated filtering method solved the problem easily and quickly and has been incorporated into an operational system. Its simplicity and robustness suggest that one might want to consider this method for some particle filter smoothing problems that can be solved by other methods.

A. Smoothing Particle Filters

In tracking situations, one is often faced with nonlinear measurements, such as lines-of-bearing and non-Gaussian motion models. The combination of nonlinear measurements and non-Gaussian motion models means that the traditional Kalman filter approach to tracking does not work well in these situations. For bearings-only tracking, particle filters have been shown [9] to outperform a Kalman filter as well as numerous nonlinear extensions of it.

In the case of a Kalman filter, there are efficient methods for smoothing, for example, the Rauch–Tung–Striebel smoother described in Section 3.2.3 of [12] or in [10]. Smoothing a particle filter is more difficult. If the particle filter preserves the full target path as the particles are split and reweighted during the resampling process, then the surviving paths and their posterior weights provide an estimate of the posterior distribution on the target paths. The difficulty with this smoother is that resampling particles usually leads to a set of surviving particles (paths) that descend from a small number of initial paths, and in some cases, only one initial path. As a result, this estimate loses resolution as one proceeds backward in time.

This generates the need for a better method of estimating the smoothed (posterior) distribution on the paths of a particle filter. Reference [10] provides a succinct review of Bayesian smoothing methods and, along with [5] and [8], provides an excellent overview of smoothing methods for particle filters.

Forward–backward smoothing, as described in Section 3.1.4 of [12] is a general solution to the smoothing problem. The difficulty with this solution is that, except in the case of Kalman filtering, one cannot evaluate the integrals involved explicitly. As a result, numerical methods are required for problems such as smoothing the output of a particle filter.

References [8], [6], and [3] present numerical methods for smoothing discrete-time particle filters that are aimed at producing marginal distributions on the state of the smoothed process at the discrete times of the process. These methods assume that the process is Markovian with an explicit functional form for the transition density.

Under these assumptions, Godsill et al. [5] developed a numerical approach to forward–backward smoothing called backward simulation. As with repeated filtering, backward simulation begins with the output from a particle filter with particles that preserve the full path of the particle. Backward smoothing produces a discrete set of independent sample paths from the posterior distribution on sample paths given the measurements received in $[0, T]$. By construction, the state of a smoothed path at time t is equal to one of the states in the particle filter approximation to the distribution at time t. References [1] and [2] remove this restriction to provide improved diversity and accuracy of the smoothing approximation.

Unfortunately, the smoothing problem that we wished to solve concerned a surveillance tracking system that used a quite natural but somewhat complex, continuous-time motion model that did not have a closed-form transition function. Although we could not use the methods referenced above, the structure of the problem allowed us to apply an MCMC method for generating the posterior distribution on target paths. In Example 2 of [11], we applied this method to a simplified version of this problem. Even for the simplified problem, the procedure was difficult and complex. Because of the nature of the motion model, a reversible-jump MCMC was required, which is even more complex than a standard MCMC. See the Appendix of [11]. However, the method obtained reasonable results on this difficult problem. As part of the further analysis and testing of this smoother, we examined the stability of the results. To do this, we ran the MCMC for 1 million iterations to estimate the posterior distribution on paths. This process took 4 h or more on a modest laptop. To test the stability of the procedure, we made a second MCMC run with 1 million iterations using the same inputs as the first run but with a different starting path for the iterations. The results were qualitatively different. The MCMC process had not converged even after 1 million iterations.

As mentioned above, we decided that the MCMC approach is too complex and delicate for an operational system. In its place, we developed the much simpler, faster, more robust, and more general repeated filtering approach presented here. Repeated filtering can be used with any stochastic process model for which one can generate independent sample paths from the process distribution. For measurements, the only requirement is that one be able to calculate likelihood functions for them. In Example IV.B, the repeated filtering method is applied to the surveillance problem mentioned above, where it performs well.

B. Outline of Paper

Section II presents a quick overview of Bayesian particle filtering to establish the notation and terminology used in this paper. Section III describes the repeated filtering method of smoothing, and Section IV presents four examples of the application of repeated filtering. Examples 1 and 2 have the same settings as Examples 1 and 2 in [11], but the results are obtained by repeated filtering. In Example 1, we apply repeated filtering to a problem that has a Kalman smoother solution and show that it produces a good approximation to that solution. Example 2 is the surveillance problem to which we applied the MCMC method in [11]. We have no yardstick by which to measure the accuracy of the solution we obtained. However, by comparing the repeated filtering solution to the actual target path, we show that this method provides a reasonable and stable solution for this problem.

Example 3 applies repeated filtering to smooth a nonlinear time series not related to tracking. This is the same problem as in Example 1 of [5], which obtained a smoothed solution to the time series using methods that require a discrete-time Markov process with a closed-form transition density. We apply repeated filtering to this problem and obtain results comparable to those in [5]. We then modified the stochastic process by adding reflecting boundaries, which produces a problem that is very difficult to solve with the methods of [5] but is simple to solve using repeated filtering.

II. BAYESIAN PARTICLE FILTERING

For this discussion, Bayesian particle filtering begins with a prior distribution on a time-varying parameter (e.g., target state) in the form of a stochastic process X on the state space S. Time is continuous, running over $[0, T]$, and the state space S can be continuous, discrete, or a combination of the two. The modifications when time is discrete will be obvious.

We approximate the prior stochastic process X (target motion model) by making a large number N of independent draws from the sample paths of the process. These sample paths form a discrete path approximation to the process. There may be times when it is more efficient to use a proposal distribution for obtaining independent sample paths from the process prior and to weight these appropriately to obtain an approximation of the stochastic process X. However, we do not consider that possibility here.

Let $\{x_n, n = 1, \ldots, N\}$ be the set of N sample paths that we have drawn from the process X. Each x_n specifies a possible target path with $x_n(t) \in S$ being the target state at time t for $t \in [0, T]$. We call x_n a *particle path* and $x_n(t)$ a *particle state* at time t. We assign probability $p(n) = 1/N$ to x_n for $n = 1, \ldots, N$ and define

$$P_N = \left\{ (x_n, p(n)), n = 1, \ldots, N \right\}$$

to be the prior *particle path distribution*. This distribution is a discrete sample path approximation to the prior distribution on the process X. The distribution P_N produces a prior *particle state distribution* for each $t \in [0, T]$ by

$$P_N^t = \left\{ (x_n(t), p(n)), n = 1, \ldots, N \right\}.$$

Bayesian particle filtering computes the Bayesian posterior distribution on this discrete particle state approximation at time t given the measurements received by time t.

In performing Bayesian filtering on this discrete sample path approximation, we obtain a solution to the particle filtering problem that is an approximation to the filtering problem on X. Thus, we find an exact solution to a problem that approximates the problem we wish to solve. The quality of this solution will depend on the quality of the discrete sample path approximation used to represent X.

A. Bayesian Recursion

We receive measurements at a discrete sequence of possibly random times $0 \leq t_1 < t_2 \cdots < t_K \leq T$. Let $L_k(y_k|\cdot)$ be the likelihood function for the measurement $Y_k = y_k$ received at t_k. Specifically,

$$L_k(y_k|s) = \Pr\left\{ Y_k = y_k | X(t_k) = s \right\} \text{ for } s \in S. \tag{1}$$

Note, we use Pr to indicate probability or probability density as appropriate.

Suppose we have received the measurement $Y_1 = y_1$ at time t_1. We compute

$$p(n|y_1) = \frac{L_1(y_1|x_n(t_1))\, p(n)}{\sum_{m=1}^{N} L_1(y_1|x_m(t_1))\, p(m)} \text{ for } n = 1, \ldots, N \tag{2}$$

to obtain

$$P_N(y_1) = \left\{ (x_n, p(n|y_1)), n = 1, \ldots, N \right\}, \tag{3}$$

which is the posterior particle path distribution given $Y_1 = y_1$.

Define $y_{1:k} = \{y_1, \ldots, y_k\}$ and $Y_{1:k} = \{Y_1, \ldots, Y_k\}$ for $k = 1, \ldots, K$. Suppose

$$P_N(y_{1:k-1}) = \left\{ (x_n, p(n|y_{1:k-1})), n = 1, \ldots, N \right\}$$

is the posterior particle path distribution given $Y_{1:k-1} = y_{1\,k-1}$, and we receive the measurement $Y_k = y_k$ at time t_k. We compute

$$p(n|y_{1:k}) = \frac{L_k(y_k|x_n(t_1))\, p(n|y_{1:k-1})}{\sum_{m=1}^{N} L_k(y_k|x_m(t_1))\, p(m|y_{1:k-1})} \tag{4}$$

for $n = 1, \ldots, N$ to obtain

$$P_N(y_{1:k}) = \left\{ (x_n, p(n|y_{1:k})), n = 1, \ldots, N \right\}, \tag{5}$$

which is the posterior particle path distribution given $Y_{1:k} = y_{1:k}$.

B. Resampling

A common problem with particle filters is that as measurements are received and processed into the filter, the posterior probability distribution tends to become concentrated on a small number of particles, causing the filter to lose resolution. This problem can be solved by resampling.

When one must resample, the filtering process becomes more complicated than described above. Instead of generating a set of N complete paths at the beginning of the filter, one must generate the paths sequentially in time so that at the time of kth measurement, one has a set of N paths (particles) over $[0, t_k]$ that provides good resolution for the distribution on system state at time t_k.

One method of resampling, described in Sections 3.3.3 and 3.3.4 of [12], splits high-probability particles into multiple (almost identical) particles and "kills off" low-probability particles in a manner that produces exactly N particles. Each child particle inherits the path history of its parent but has a slightly different state at time t_k. The resulting set of particles have probability $p(n|y_{1:l}) = 1/N$ for $n = 1, \ldots, N$.

The paths of the resampled particles are then extended to the time t_{k+1} of the next measurement to obtain $P_N^{t_{k+1}}(y_{1:k})$, the particle state distribution at time t_{k+1} given the measurements $y_{1:k}$. When $Y_{k+1} = y_{k+1}$ is received, we compute the posterior distribution on the particle paths using (4), with k replaced by $k + 1$. Alternatively, one may want to use a proposal distribution in place of $P_N^{t_{k+1}}(y_{1:k})$ to compute the posterior distribution on the particle paths.

C. The Problem With Resampling

The above procedure is a bootstrap version of the SIR particle smoother of Kitagawa [7]. This works well to provide a high-resolution estimate of the posterior distribution of the present target state at the time of the last measurement. The difficulty is that the surviving resampled particles tend to originate from a small number of the original particles, so the posterior distribution on sample paths lacks resolution as one moves from present time back to time 0. The set of particle paths obtained by time T in this fashion form an estimate of the smoothed distribution on sample paths. However, it is not a very good estimate. See [3] and [4]. Simply increasing the initial number of sample paths is not an effective solution to this problem in most cases; see [8]. Repeated filtering was developed to solve this problem without the Markov or discrete-time assumptions required by other methods.

III. REPEATED FILTERING

The increasing speed, memory capacity, and capability of present-day computers allow us to propose the following method, which would not have been practical a few years ago. The method is called *repeated filtering*. It is implemented by the following recursion, which produces M independent sample paths from the smoothed distribution on sample paths.

A. Repeated Filtering Recursion

- **Step 1**. Make an initial run of the particle filter, processing the measurements received over the time interval $[0, T]$ and resampling as necessary.
- **Step 2**. Resample the particles at time T to obtain N equal probability particle paths $\{x_n; n = 1, \ldots, N\}$. Choose one of these paths at random, with each path having probability $1/N$ of being chosen. Save the chosen sample path $\bar{x}$.
- **Step 3**. Rerun the particle filter with the same measurements as in Step 1, but drawing particles that are independent of those drawn in Step 1. This will ensure that we choose new and independent samples of the target state at time 0 and at the measurement times.
- **Step 4**. Make a random draw to choose one of the sample paths as in Step 2. Save this sample path.
- **Step 5**. Repeat Steps 3 and 4, using particles that are independent of those drawn previously, until one obtains M smoothed sample paths $\bar{x}_m$ for $m = 1, \ldots, M$. Define the particle path distribution

$$\bar{P}_N(y_{1:K}) = \left\{(\bar{x}_m, 1/M); m = 1, \ldots, M\right\}. \quad (6)$$

Then $\bar{P}_N(y_{1:K})$ is a discrete path approximation to the posterior distribution on sample paths given the measurements received in $[0, T]$.

The solution in (6) gives each smoothed path an equal weight. We hypothesize that an alternate weighting scheme applied to the smoothed paths in (6) would produce a better solution. However, none of the methods we have tried have done this. This is an area for further investigation.

B. Marginal Distributions

For any $t \in [0, T]$, we can obtain from $\bar{P}_N(y_{1:K})$ a particle state estimate for the smoothed marginal distribution at time t as follows:

$$\bar{P}_N^t(y_{1:K}) = \left\{(\bar{x}_m(t), 1/M); m = 1, \ldots, M\right\}. \quad (7)$$

We often provide a visual representation of such a distribution by imposing a grid of cells on the state space, summing the probability of the points in each cell, and color coding the cells to represent the probabilities in the cells.

The ability to estimate the distribution of the state of the smoothed process at a time between measurements can be particularly important in situations where there are large time gaps between measurements, as occurs in some surveillance problems. In addition, having a set of smoothed paths can be helpful in determining patterns of motion. Moreover, as noted in [5], having sample

paths also allows one to explore relationships between the state of the process at different times.

C. Resolution of the Smoothed Solution

Resampling within each run of the particle filter is necessary to preserve the resolution of the estimate of the posterior distribution as the number of measurements increases. Making independent runs of the particle filter in Step 3 to obtain M posterior sample paths is necessary to preserve the resolution of the estimate of the posterior as one goes back in time toward 0. Determining the number of particle filter runs required and the number of particles for a run generally requires some experimentation.

We expect that there is some limitation on the length of the interval $[0, T]$ over which this process produces solutions with good resolution, or more likely, as the time interval gets longer, the number of particle filter runs M may need to get larger. We have not explored this question. Another possibility is to break the interval $[0, T]$ into two or more subintervals and splice the solutions from the subintervals together in some fashion. We have not explored this possibility either.

D. Computation Time

The computation time to obtain a repeated filtering solution depends on the time to perform one filter run, which depends on the complexity of the problem. Generating M independent samples from the posterior will take M times as long as a single filter run. If time becomes a problem, one can easily apply coarse grain parallel processing by allocating the repetition of Steps 3 and 4 across a number of processors.

IV. EXAMPLES

This section presents four examples of estimating the posterior distribution on sample paths in $[0, T]$ using repeated filtering. The first example compares repeated filtering to a Kalman smoother where the exact solution is known. The second example involves a simplified surveillance situation where the target is moving through an area in which it has to avoid certain regions. Even though this is a simplified situation, it is still a challenge for smoothing. We use a motion model called a generalized random tour (GRT), see [11] or Section 1.3.3 of [12], which is a special case of a variable rate particle filter. We incorporate avoidance regions to provide a more realistic and challenging motion model. The last two examples smooth a standard nonlinear time series used to test particle filters.

A. Example 1: Comparison to a Kalman Smoother

For this comparison, the motion model is the almost-constant velocity model described below, and the measurements are position measurements with additive circular normal errors. We find a repeated filtering solution for this example and compare it to the solution from the Rauch–Tung–Striebel smoother [10, p. 135].

1) Almost Constant Velocity Model: The target state is given by a position–velocity pair (x, v). The state at time 0 is

$$(x_0, v_0) \sim \eta\left(\cdot, (\bar{x}, \bar{v}), \Sigma_0\right), \tag{8}$$

where we use $\eta(\cdot, \mu, \Sigma)$ to denote a normal density function with mean μ and covariance Σ. Let Δ be a fixed time increment. There are I time increments and $T = I\Delta$. The target proceeds at velocity v_0 until time $t_1 = \Delta$ at which time a new velocity v_1 is obtained by adding a small, independent, mean-zero, Gaussian distributed variation to v_0 to obtain v_1. The target continues at this velocity until the next time increment. We may express this mathematically as follows. Let (x_i, v_i) be the target state at time $t_i = i\Delta$. Then

$$\begin{pmatrix} x_i \\ v_i \end{pmatrix} = \begin{pmatrix} x_{i-1} + \Delta v_{i-1} \\ v_{i-1} + w_i \end{pmatrix} \text{ for } i = 1, 2 \ldots, \tag{9}$$

where $\{w_i : i = 1, \ldots, I\}$ are independent, identically distributed random variables with $w_i \sim \eta(\cdot, (0, 0), Q)$ and Q is a "small" covariance matrix.

Let $\mathbf{I}_m$ be the m-dimensional identity matrix. The parameters of the motion model are $\Delta = 1$ hr,

$$\bar{x} = (0, 0)\,, \bar{v} = 7 \text{ kn } (\cos(\theta), \sin(\theta)) \text{ where } \theta = \frac{\pi}{6}$$

$$\Sigma_0 = \begin{bmatrix} \sigma_x^2 \mathbf{I}_2 & 0 \\ 0 & \sigma_v^2 \mathbf{I}_2 \end{bmatrix} \text{ where } \sigma_x = 4 \text{ nm}, \sigma_v = 1 \text{ kn}$$

$$Q = \sigma_w^2 \mathbf{I}_2, \text{ where } \sigma_w = 1 \text{ kn}. \tag{10}$$

We use the abbreviations nm for nautical miles and kn for knots. A knot equals 1 nm/h.

The target follows a slightly curved path starting at the origin, as shown by the black line in Fig. 1. The time duration is 10 h. There are eleven position measurements received at 1-h increments over the duration of the path. The 2-sigma uncertainty ellipses for the measurements are shown in black. The measurements have circular Gaussian errors with a standard deviation of 4 nm.

2) Comparison of Kalman and Repeated Filtering Smoothers: The Kalman smoother provides an analytic solution to this problem. The red ellipses are the 2-sigma ellipses from the Kalman smoother solution at equally spaced times on the path.

Repeated filtering was applied to this problem by drawing 10 000 particle paths from the almost-constant velocity motion model and performing the recursion in Section III to obtain 400 samples from the posterior distribution on target paths. The green ellipses are the 2-sigma ellipses derived from the empirical means and covariances of the path positions at the same times as the Kalman smoother ellipses. As one can see, there is good agreement between the two plots of 2-sigma ellipses. Note that agreement improves as time increases

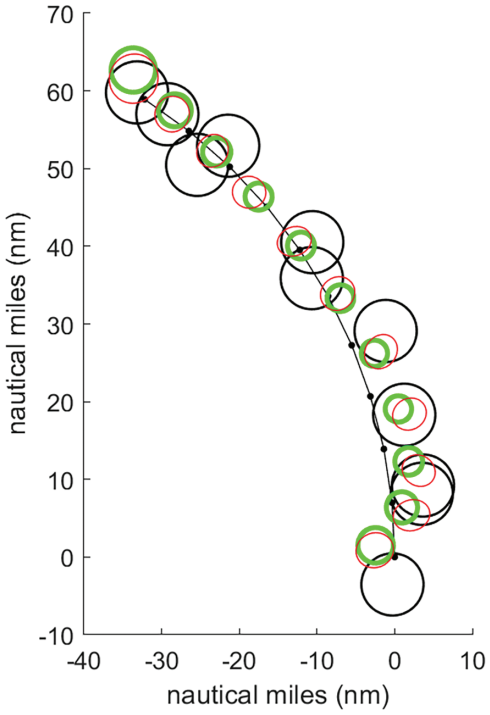


Fig. 1. Comparison of Kalman and repeated filtering smoothers. The black line is the target's path, which starts at (0,0); black circles are 2-σ uncertainty circles for measurements; red ellipses are 2-σ ellipses for the Kalman smoother solution; green ellipses are 2-σ ellipses for the repeated filtering smoother solution.

because the smoother has more history to use for the smoothing.

We made the following computation to measure how well repeated filtering approximated the optimal solution from the Kalman smoother. At each of the eleven equally spaced points on the target track in Fig. 1, we computed the mean squared distance from the point to the bivariate normal distribution at that point computed by the Kalman smoother and to the bivariate normal distribution corresponding to the $2-\sigma$ ellipse for the repeated filtering result. We averaged these results over the eleven points and took the square root of this average to obtain the square root of the average mean squared missed distance for the Kalman and repeated filtering smoothers.

The results were 3.97 nm for the Kalman smoother and 4.61 nm for repeated filtering. The repeated filtering result is only 16% larger than the Kalman result demonstrating that repeated filtering provides a good approximation to the posterior path distribution in a case where we can calculate the exact distribution. We produced only 400 smoothed paths for this example. Using more paths would improve the approximation.

B. Example 2: Surveillance Problem

As before, we use a target state space that is 2-dimensional in position and velocity and use (x, v) to represent a position and velocity pair in this space. A

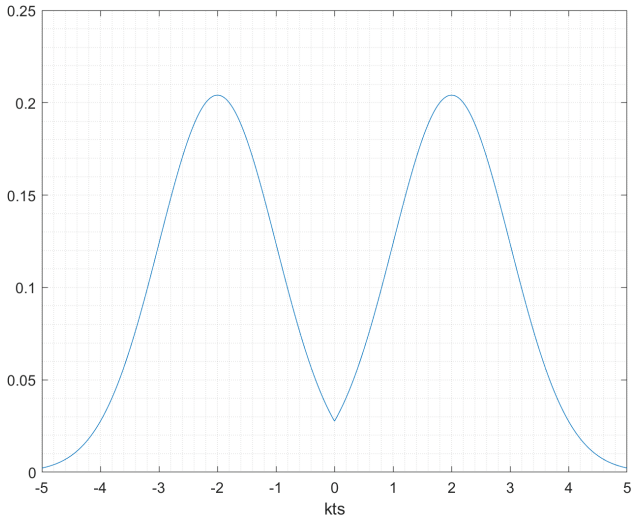


Fig. 2. Speed change distribution.

GRT motion model is specified by first specifying a probability (density) function $p_0(x, v)$ on the position and velocity (x_0, v_0) of the target at time 0. As time progress, the target changes velocity (instantaneously) at the event times of a Poisson process with rate λ.

Between velocity changes, the target follows a constant velocity path at the previously chosen velocity. When the target changes velocity, its new velocity v_i is drawn from a probability (density) function $p(\cdot|v_{i-1})$, where v_{i-1} is the velocity just prior to the change. For many tracking problems, the GRT motion model is more operationally realistic than the almost-constant velocity motion model or other often-used Gaussian-diffusion motion models.

For this example, we set $\lambda = 0.25/\text{h}$ and

$$p_0(x_0, v_0) = \eta\left(x_0, (0,0), (15\,\text{nm})^2\mathbf{I}_2\right) \times \eta\left(v_0, (0,0), (10\,\text{kn})^2\mathbf{I}_2\right).$$

When a velocity change occurs, the new velocity is chosen by making independent draws to determine the changes to the speed and course of the target. The speed change distribution is symmetric about zero. On each side of zero, the distribution is proportional to that of a truncated Gaussian whose mean has an absolute value of 2 kn and a standard deviation of 1 kn, as shown in Fig. 2. Similarly, the course change distribution is symmetric about zero, with each side being proportional to a truncated Gaussian whose mean has an absolute value of 60°and a standard deviation of 30°.

As the sample paths are generated, we ensure that they stay clear of avoidance regions, as follows: When a velocity change takes place, the time on that leg is drawn as well as the new velocity. If the resulting leg hits an avoidance region, a new velocity is drawn. This process is repeated up to a maximum of 20 times until the resulting leg does not intersect an avoidance region. If no

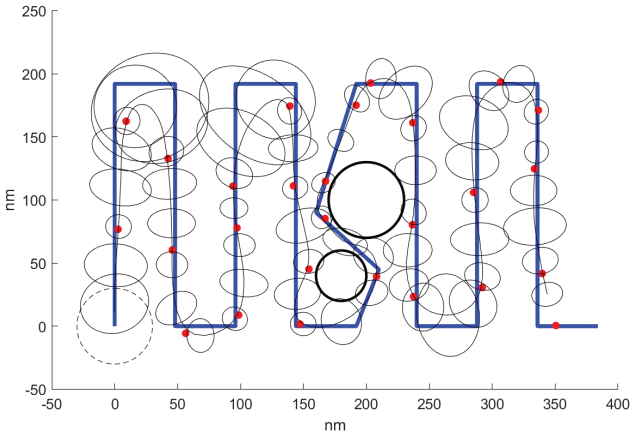


Fig. 3. Slinky plot from the first run of the repeated filtering smoother. The heavy blue line shows the target's path, which starts at (0, 0). The red dots show the measurements. The black circles show discs of the regions that the target must avoid. The red ellipses are 2-sigma ellipses generated by the repeated filtering smoother. The dashed circle shows the 2-sigma ellipse for the initial position distribution at time 0.

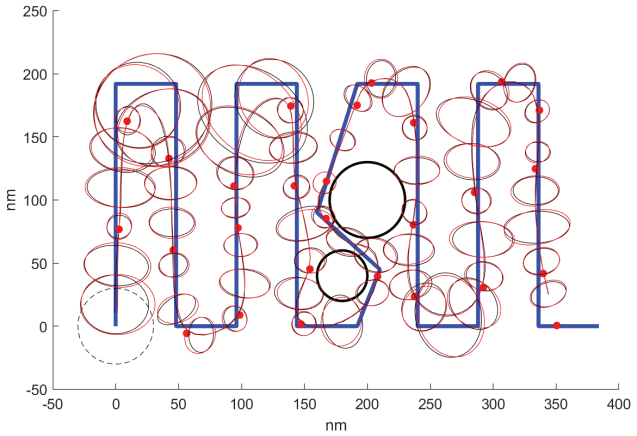


Fig. 4. Comparison of the slinky plots from two runs of the repeated filtering method using the same inputs but independent random numbers. The blue line shows the target's path.

such leg is found, then the path is discarded and a new one generated in its place.

1) Scenario Description: The actual target path, shown in blue in Fig. 3, has a fixed speed of 8 kn. It follows the ladder path with long legs of 24 h duration and short legs of 6 h duration. The black circles indicate regions that the target must avoid as it traverses its path. The target path starts near the origin. The total time is 240 h or 10 days.

The time to the first measurement is gamma-distributed with a mean of 4 h and variance of $(8/3)$ h^2. The time intervals between subsequent measurements are independent with this same gamma distribution. The measurements are of position with a circular Gaussian error distribution having a standard deviation of 10 nm. In Fig. 3, measurements are indicated by red dots, and the dashed circle shows the 2-sigma ellipse of the initial position distribution.

2) Repeated Filtering Smoother: For repeated filtering, we ran the particle filter with $N = 10\,000$ and at time T randomly chose one of the paths, with each path having an equal probability of being chosen.

We repeated Steps 3 and 4 in Section III-A to obtain $M = 400$ sample paths from the posterior (smoothed) distribution on the target paths. The 2-sigma ellipses for the position estimates were calculated every 4 h. This sequence of ellipses, called a slinky plot, is shown in Fig. 3. The ellipses represent normal approximations to the position distributions every 4 h. Thus, even though some ellipses intersect an avoidance region, the paths themselves do not.

To illustrate the stability of the repeated filtering method, we repeated this run a second time using the same measurements as in the first run but using random draws independent of those made for the first run. We overlaid the slinky plots for the two runs in Fig. 4. As the reader can see, there is little, if any, difference in the plots, which gives us confidence in the stability of the method.

A sample of the smoothed paths from repeated filtering is shown in Fig. 5. Note that none of the sample paths pass through the avoidance regions. Note also that there is more uncertainty in the distribution of the smoothed paths near (0,0), the target's starting point at time 0, than there is close to time T. It is not surprising that having past history as well as future information is helpful in estimating the target's smoothed path.

3) MCMC Smoothing: In [11], we applied an MCMC method to estimate the posterior distribution on the target paths for this example. The procedure was difficult and complex. In the hope of ensuring the stability of the results, we ran the MCMC for 1 million iterations to estimate the posterior distribution on paths. This process took 4 h or more on a modest laptop. The results looked reasonable, but to test the stability of the procedure, we made a second MCMC run with 1 million iterations

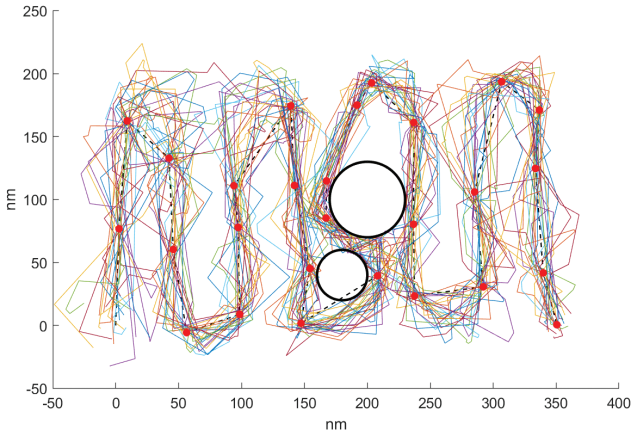


Fig. 5. Smoothed sample paths selected by random draws from the set of smoothed paths. Each path has an equal probability of being chosen. Note that none of the sample paths pass through the avoidance regions. The dashed line connects the measurements.

using the same inputs as the first run but with a different starting path for the iterations. As noted in the introduction, the results were qualitatively different. The MCMC had not converged even after 1 million iterations. At this point, we abandoned the MCMC approach and developed the repeated filtering approach.

4) Discussion: We have no analytical method with which to compare our smoothed solution in this example. The solution appears reasonable compared to the actual target path in this example, even though this is a difficult problem and there is a mismatch between the motion model and the target's motion. The smoothed paths stay out of avoidance regions, and the distribution is repeatable up to the small differences that are to be expected in Monte Carlo solutions.

We have used slinky plots to display the smoothed solution. Alternatively, one could calculate a mean smoothed path by finding the mean of the position of the paths at each time in a sequence of evenly spaced times and displaying the line connecting these means. Or, one could display both the mean path and the slinky plot.

C. Example 3: Smoothing a Nonlinear Time Series

In this example, we apply repeated filtering to smooth the output of the stochastic nonlinear time series model given in Example 1 of [5]. Reference [5] describes this model as a standard nonlinear time series model that has been used extensively for testing numerical filtering techniques.

The time series $\{X(t), t = 1, ..., 100\}$ has for its initial state $X(1) \sim \eta(\cdot, 0, 10)$ and is defined for $t \geq 2$ by

$$X(t) = \frac{X(t-1)}{2} + \frac{2.5X(t-1)}{1+X^2(t-1)} + 8\cos(1.2t) + v(t)$$
$$v(t) \sim \eta(\cdot, 0, 10), \; v(t) \text{ independent of } v(s) \text{ for } s \neq t. \tag{11}$$

The measurements $\{Y(t), t = 1, ..., 100\}$ are defined by

$$Y(t) = \frac{X^2(t)}{20} + w(t)$$
$$w(t) \sim \eta(\cdot, 0, 1), \; \mathrm{w}(t) \text{ independent of } \mathrm{w}(s) \text{ for } s \neq t. \tag{12}$$

We cannot reproduce the results in Example 1 in [5] exactly because we do not have access to the sample path [5] of the process used for their example or the series of measurements produced. While we cannot reproduce this example exactly, we are able to produce comparable results and similar figures, which leads us to conclude that the repeated filtering method produces results comparable to the method in [5], which is limited to discrete-time Markov processes with closed-form transition functions.

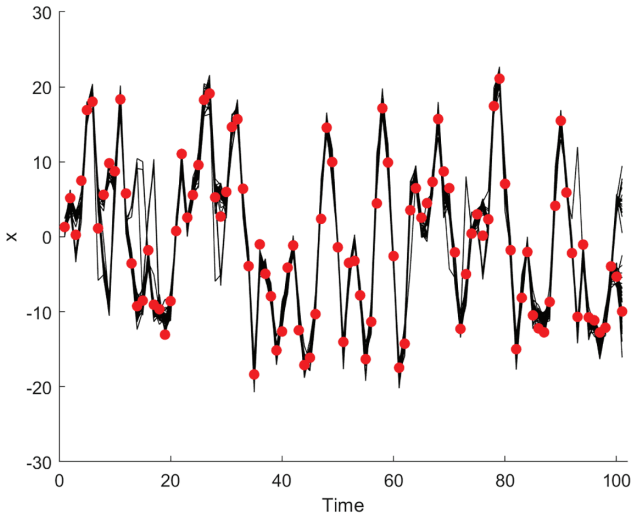


Fig. 6. Fifty smoothed paths are shown in black; the time series values are in red.

1) Repeated Filtering Approach: To apply repeated filtering, we simulated one sample path and set of measurements from the time series defined by (11) and (12). Using these as inputs, we ran a standard SIR particle filter with 1000 particles resampling as described in Section II-B. We used the stochastic process defined in (11) and (12) for our motion model for the filter. The particles were resampled at each time step so that they all had equal weight. At the conclusion of a filter run at time 100, we selected one of these paths by making a draw from this set of particles, with each particle having an equal probability of being drawn. This path was saved as a smoothed path. We repeated the filtering process 1000 times, using independent random numbers to produce the particles and drawing one of them for a smoothed path. The resulting set of 1000 independently drawn smoothed paths constitutes our estimate of the posterior distribution on the target paths given the measurements in [1,100].

Fig. 6 shows a sample of 50 smoothed paths in black and the actual values of the time series in red. Looking at the measurement equation (12), one can see that a value x of the series will produce the same measurement as $-x$. As more measurements are received, the smoother (usually) sorts out this ambiguity. This ambiguity has produced the bimodal results near time 100.

Fig. 7 shows the smoother results when restricted to the interval [0, 51]. The ambiguity in the smoothed solution near time 51 in this figure is resolved by time 100 in Fig. 6.

Figs. 8–10 below are similar to Figs. 4, 5, and 7 in [5], and the results are qualitatively similar. Since the sampled time series and measurements in our data are somewhat different from those in [5], we do not expect an exact match.

Fig. 8 shows a histogram of the smoothed distribution values at each of the 100 times. As [5] notes, one of the advantages of finding smoothed paths rather than marginal distributions at each time is the ability to analyze joint densities of values at two different times.

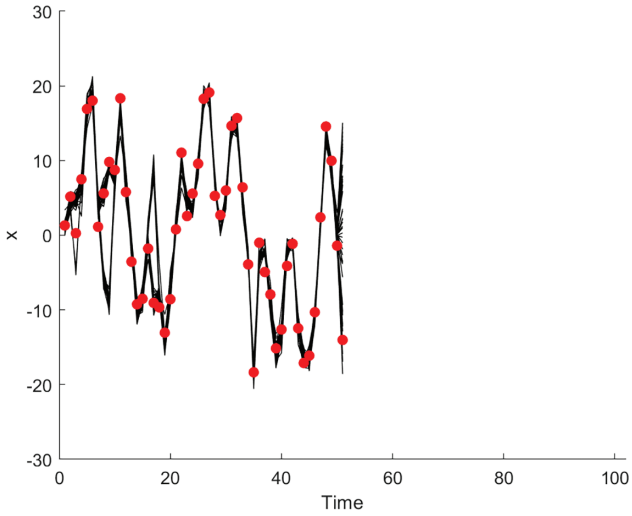


Fig. 7. Smoother results for [0,51]. Smoothed paths are in black. Time series values are in red.

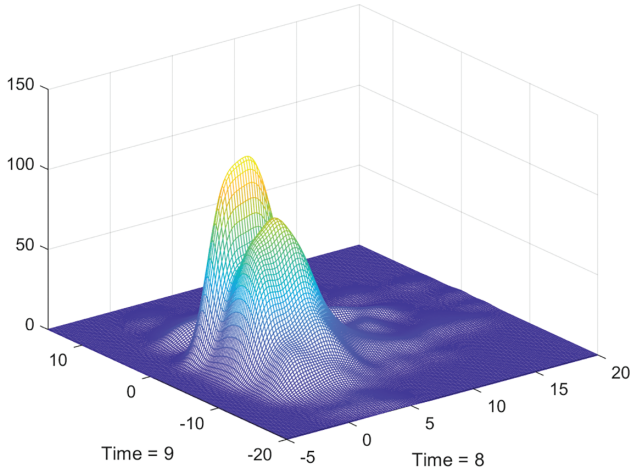


Fig. 9. Joint density plot for the values of the smoothed time series at times 8 and 9. Note the multimodal distribution.

Figures 9 and 10 show examples of these joint densities and exhibit behavior similar to that seen in [5].

D. Smothing a Nonlinear Time Series With Reflecting Boundaries

This example adds reflecting boundaries at $+15$ and -15 to the nonlinear time series example in Section IV-C. Smoothing of this process is easily performed using repeated filtering but is difficult to do using the methods of [5]. Equations (13)–(15) in the Appendix give the revision to the equation for $X(t)$ produced by the reflecting boundaries.

One can see from these equations that for each transition, one must allow for the reflection off one or more boundaries to determine the distribution of $X(t)$ given $X(t-1)$. In fact, since the term, $v(t)$ is drawn from a Gaussian distribution, the transition function involves summing an infinite number of terms to account for the number of possible reflections!

To smooth this process, we modified the particle filter in Section IV-C by adding the reflective boundaries and performed repeated filtering as above.

Fig. 11 shows 50 smoothed sample paths from this process. The smoothed paths are shown in black, and the red dots indicate the values of the process. Note the ambiguity at time 100. The value of the process is approximately -10, but the smoother shows an ambiguity about 0 because of the measurement model. This ambiguity will not be resolved until more data is received. If one truncated the time series at time 50, as in Fig. 12, then one would see a similar ambiguity that is resolved as the filter receives more data.

Fig. 13 shows the joint density of the smoothed paths at times 99 and 100. As one can see from this figure, if the series is positive (negative) at time 99, it will be positive (negative) at time 100.

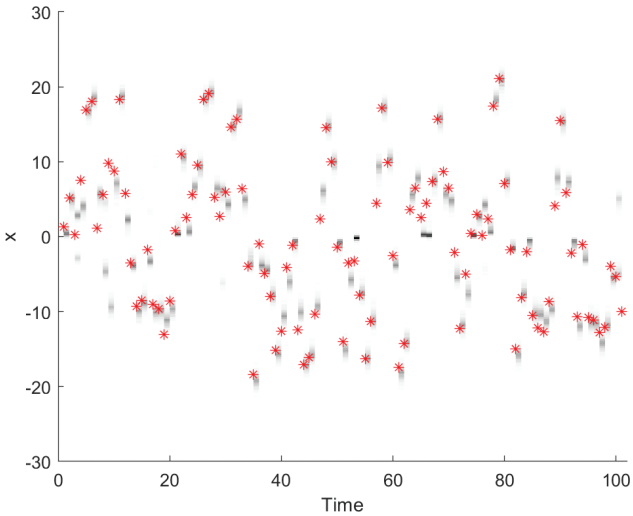


Fig. 8. Histogram of smoother values. Dark grey indicates higher-density areas. Red stars show the actual values of the time series.

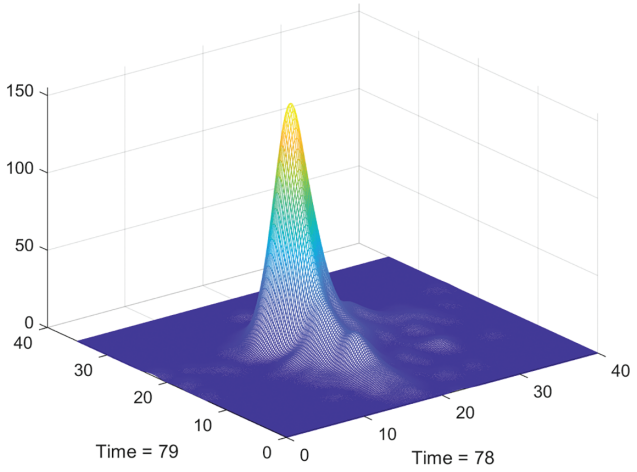


Fig. 10. Joint density plot for times 77 and 78. This density is unimodal but not Gaussian.

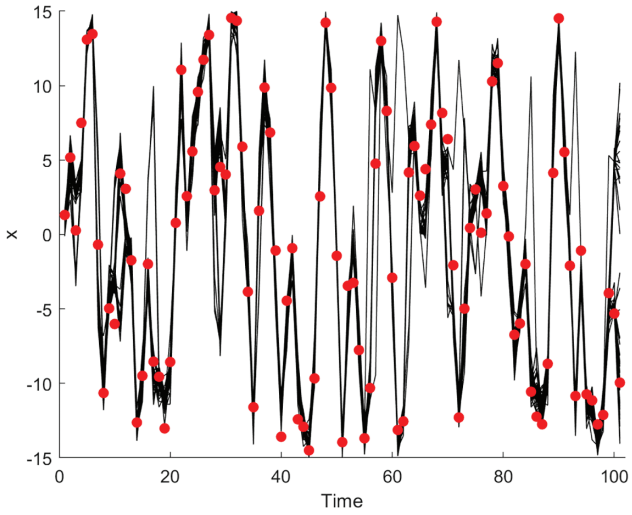


Fig. 11. Smoothed paths for time series with reflecting boundaries. Smooth paths are shown in black; times series values in red.

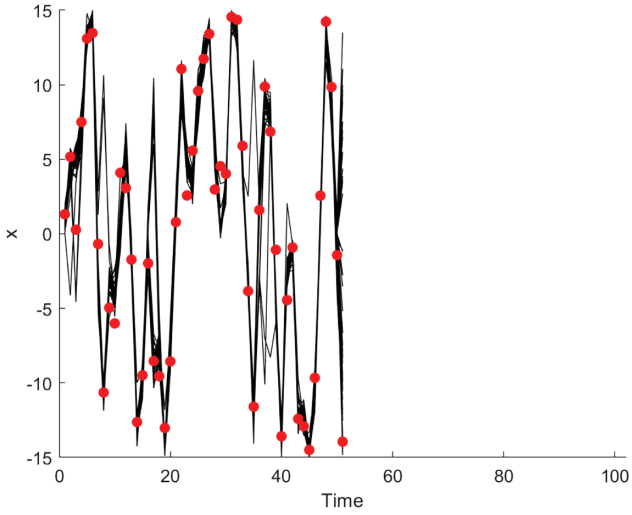


Fig. 12. Smoothed paths resulting from stopping the series at time 50.

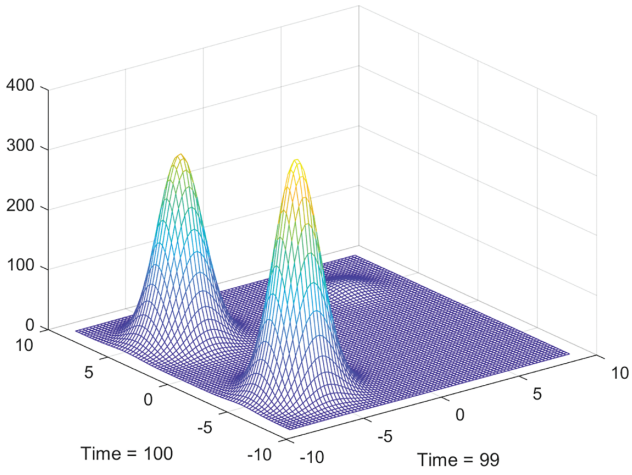


Fig. 13. Joint density at times 99 and 100.

V. CONCLUSIONS

This paper presents the repeated filtering smoother, which is a simpler method than most smoothing methods and can be applied to any SIR particle filter. Like the method in [5], the smoother produces sample paths from the smoothed distribution, allowing for more detailed analysis of path behavior than can be obtained from smoothers that produce only marginal distributions. The only restriction on the stochastic process defining the motion model used for the particle filter is that one must be able to draw independent sample paths from the process and that these paths can be produced sequentially in time. In particular, the process does not have to be Markovian in its state space.

We have demonstrated the repeated filtering method on a Kalman filter problem and shown that it produces comparable results. We demonstrated the method on a tracking problem with a motion model whose transition function does not have a closed analytical form, and which has unusual features such as avoidance areas. Next, we demonstrated the repeated filtering method on a standard nonlinear time series problem used to test many particle filters. We also performed repeated smoothing on a modification of this time series with reflecting boundaries. The smoothing was performed on this example by simply putting reflecting boundaries on the time series and applying the repeated filtering. By contrast, the method in [5] would require substantial additional effort.

In all four examples, the repeated filtering method performed well and required only modest amounts of computational effort. We stress again the simplicity and generality of this method. If one can construct a good particle filter for the process, one can easily find smoothed sample paths using repeated filtering. The computational load is easy to estimate. If you want M smoothed paths, the computational effort will be M times the effort required for a single run of the particle filter.

APPENDIX

EQUATION FOR TIME SERIES WITH REFLECTING BONDARIES

This appendix derives the modifications to (11) resulting from adding the reflecting boundaries in Example 4.

If $v(t) > 0$, let

$$Z(t) = \frac{X(t-1)}{2} + \frac{2.5X(t-1)}{1+X^2(t-1)} + 8\cos(1.2t). \quad (13)$$

If $v(t) \leq 15 - Z(t)$, then

$$X(t) = Z(t) + v(t). \quad (14)$$

If $v(t) > 15 - Z(t)$, let

$$R(t) = v(t) - (15 - Z(t))$$
$$n(t) = \lfloor R(t)/30 \rfloor$$
$$f(t) = v(t) - [30n(t) + (15 - Z(t))].$$

Then

$$X(t) = (-1)^{n(t)} (15 - f(t)). \tag{15}$$

If $v(t) \leq 0$, then one obtains a similar set of equations for $X(t)$.

REFERENCES

[1] M. Briers, A. Doucet, and S. Maskell
"Smoothing algorithms for state-space models,"
Ann. Inst. Stat. Math., vol. 62, pp. 61–89, 2010.

[2] P. Bunch and S. Godsill
"Improved particle approximations to the joint smoothing distribution using Markov Chain Monte Carlo,"
IEEE Trans. Signal Process., vol. 61, no. 4, pp. 956–963, Feb. 2013.

[3] A. Doucet, J. F. G De Freitas, and N. J. Gordon
Eds.
Sequential Monte Carlo Methods in Practice. New York, NY: Springer, 2001.

[4] A. Doucet, S. J. Godsill, and C. Andrieu
"On sequential Monte Carlo sampling methods for Bayesian filtering,"
Stat. Comput., vol. 10, no. 3, pp. 197–208, 2000.

[5] S. J. Godsill, A. Doucet, and M. West
"Monte Carlo smoothing for nonlinear time series,"
J. Am. Stat. Assoc., vol. 99, no. 465, pp. 156–168, 2004.

[6] M. Hürzler and H. R. Künsch
"Monte Carlo approximations for general state space models,"
J. Comput. Graph. Stat. vol. 7, no. 2, pp. 175–193, 1998

[7] G. Kitagawa
"Monte Carlo filter and smoother for non-Gaussian, nonlinear state space models,"
J. Comput. Graph. Stat., vol. 5, no. 1, 1–25, 1996.

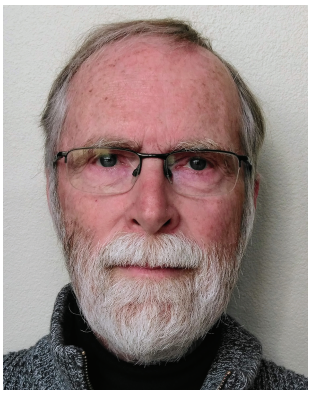
[8] M. Klaas, M. Briers, N. de Freitas, S. Maskell, and D. Lang
"Fast particle smoothing: If I had a million particles,"
in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 481–488.

[9] B. Ristic, S. Arulampalam, and N. Gordon
Beyond the Kalman Filter. Boston, MA: Artech House, 2004.

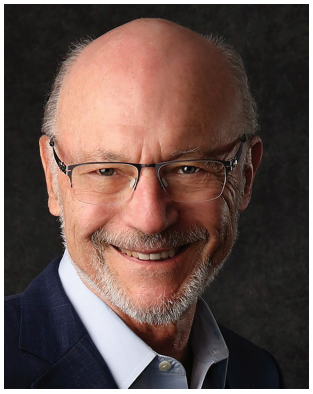
[10] S. Särkä
Bayesian Filtering and Smoothing. New York, NY: Cambridge University Press, 2013.

[11] L. D. Stone, S. L Anderson, and D. Lo
"MCMC smoothing for generalized random tour particle filters,"
in *Proc. FUSION 2018, 21st Int. Conf. Inf. Fusion*, 2018, pp. 142–150.

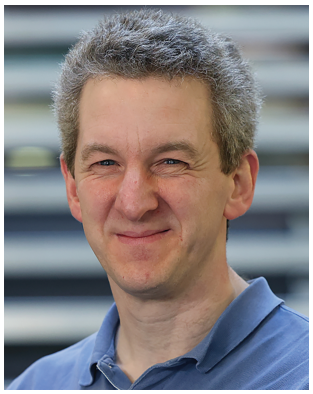
[12] L. D. Stone, R. L. Streit, T. L. Corwin, and K. L. Bell Bayesian
Multiple Target Tracking, 2nd ed. Boston, MA: Artech House, 2014.



Stephen L. Anderson received an B.A. (Honors) degree in mathematics from the University of Utah, Salt Lake City, UT, USA, in 1975. He earned a Ph.D. degree in mathematics from Brown University, Providence, RI, USA, in 1980. From 1980 to 1983, he was Acting Assistant Professor of Mathematics with the University of Washington, Seattle, WA, USA, and from 1983 to 1986, he was Assistant Professor of Mathematics and Computer Science at Wilkes University, Wilkes-Barre, PA, USA. From 1987 to 1993, he worked at Daniel H. Wagner, Associates, primarily on multitarget tracking applications. He joined Metron, Inc., in 1993, where he is currently Senior Research Scientist. At Metron, his work has focused primarily on target tracking and search applications.



Lawrence D. Stone is Chief Scientist at Metron Inc., Reston VA, USA. He is a member of the National Academy of Engineering and a fellow of the Institute for Operations Research and Management Science (INFORMS). He is a recipient of the Jacinto Steinhardt Award from the Military Applications Section of INFORMS in recognition of his applications of Operations Research to military problems. In 1975, the Operations Research Society of America awarded the Lanchester Prize to his text, *Theory of Optimal Search*. In 1986, he produced the probability maps used to locate the *S.S. Central America*, which sank in 1857, taking millions of dollars' worth of gold coins and bars to the ocean bottom one and one-half miles below. In 2010, he led the team that produced the probability distribution that guided the French to the location of the underwater wreckage of Air France Flight AF447. Recently, he used search theory methods to help guide the Canadian exploration company, Aurania, to the location of one of the lost Spanish gold cities in Ecuador. He coauthored the 2016 book *Optimal Search for Moving Targets*. He was one of the primary developers of the Search and Rescue Optimal Planning System, which has been used by the Coast Guard since 2007 to plan searches for people missing at sea. He continues to work on a number of detection and tracking systems for the United States Navy. He is a coauthor of the 2014 book, *Bayesian Multiple Target Tracking 2nd Ed.*



Simon Maskell (Member, IEEE) received the M.A., M.Eng., and Ph.D. degrees in engineering from the University of Cambridge, Cambridge, UK, in 1998, 1999, and 2003, respectively. Prior to 2013, he was a Technical Manager of command, control, and information systems with QinetiQ, UK. Since 2013, he has been a Professor of Autonomous Systems with the University of Liverpool, Liverpool, UK. His research interests include Bayesian inference applied to signal processing, multitarget tracking, data fusion, and decision support with particular emphasis on the application of sequential Monte Carlo methods in challenging data science contexts.

Absolute Calibration of Imaging Sensors

DJEDJIGA BELFADEL

Despite the efforts for precise alignment of satellite-based imaging sensors before launch, several factors may cause the values of the calibration parameters to vary between the time of ground calibration and on-orbit operation. This paper considers the problem of satellite-based imaging sensors on-board calibration while estimating the position and velocity of a target of opportunity. The pixel measurements (estimated location of the target's image in the focal-plane array) generated by these sensors are used to estimate the sensors' pointing angle biases, which is a key element of accurate tracking of a target in a space-based system. The target is assumed to be seen by the sensors from a changing direction as a function of the target position, allowing the target in this nonlinear tracking system to be observable. The evaluation of the corresponding Cramér–Rao lower bound on the covariance of the bias estimates and the statistical tests on the results of simulations show that both the target trajectory and the biases are observable and that this method is statistically efficient.

Manuscript received December 28, 2022; revised April 13, 2023; released for publication May 29, 2023.

The author is with the Electrical and Biomedical Engineering, Fairfield University, Fairfield, CT 06824, USA (e-mail: dbelfadel@fairfield.edu).

1557-6418/23/$17.00

I. INTRODUCTION

Image registration is an important research topic in many related areas, such as computer vision, automatic object detection, remote sensing, image processing, robotics, and medical imaging. Multisensor image fusion is the process of combining relevant information from several images into one image. The final output image can provide more information than any of the single images.

In the literature of computer vision, several camera calibration methods exist. These methods are classified based on the calibration object used, such as stereo calibration, plane calibration [13], line calibration [14], and self-calibration [10]. However, it is important to note that constraint conditions become weaker and precision decreases when the dimension is reduced. Thus, if high-precision results are necessary, the latter two methods may not be suitable. Furthermore, the three-dimensional calibration block required for the third method is challenging to make. Therefore, the plane calibration method is a widely used method in computer vision due to its flexibility and simplicity [13].

In order to carry out image fusion, registration error correction is crucial in multisensor systems. This requires estimation of the sensor measurement biases. Measurement bias in target tracking problems can result from a number of different sources. Some primary sources of bias errors include measurement biases due to the deterioration of initial sensor calibration over time, attitude errors caused by biases in the gyros of the inertial measurement units of (airborne or spaceborne) sensors, and timing errors due to the biases in the onboard clock of each sensor platform [9].

For angle-only sensors, imperfect registration leads to line-of-sight (LOS) angle measurement biases in azimuth and elevation. If not corrected, the registration errors can seriously degrade the global surveillance system's performance by increasing tracking errors and even introducing ghost targets. In [7], the effect of sensor and timing bias error on the tracking quality of a space-based infrared (IR) tracking system that utilizes a linearized Kalman filter (LKF) for the highly nonlinear problem of tracking a ballistic missile was presented. This was extended in [8] by proposing a method of using stars observed in the sensor background to reduce the sensor bias error. In [5] simultaneous sensors bias and targets position estimation using fixed passive sensors was proposed. A solution to the related observability issues discussed in [5] was proposed in [6] using space-based sensors. In [4], a simultaneous target state and passive sensor bias estimation were proposed.

In this paper, imaging sensor bias estimation is investigated when only a single target of opportunity is available. The tracking system consists of two or three satellites tracking a ballistic target. The sensors provide synchronized focal-plane (pixel) measurements. The data association is assumed to be correct, and the sensors'

locations are known, and we estimate their orientation biases while simultaneously estimating the state of the target (position and velocity). Our new bias estimation method is validated using a hypothetical scenario created using the System Tool Kit (STK) [1]. Two cases are considered. In the first case, we use three imaging sensors to estimate the state of a ballistic target simultaneously with the biases of the three sensors. In the second case, we estimate the position and velocity of a single target of opportunity simultaneously with the biases of two imaging sensors [3].

First, we discuss the bias estimation for synchronously biased imaging sensors in pixel coordinates. Then we evaluate the corresponding Cramér–Rao lower bound (CRLB) on the covariance of the bias estimates, which is the quantification of the available information on the sensor biases, and show via statistical tests that the estimation is statistically efficient—it meets the CRLB. Section II describes the problem formulation and solution in detail. Section III describes the simulations performed and gives the results. Finally, Section IV gives the conclusions.

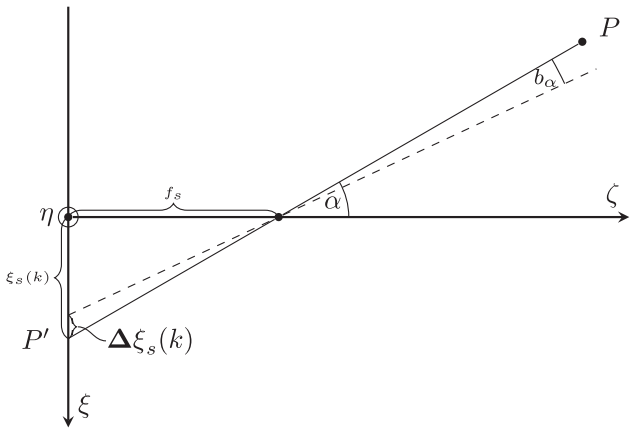


Fig. 1. Sensor coordinates and azimuth pointing bias.

II. PROBLEM FORMULATION

To locate a target in world coordinates and to estimate and correct the biases, one needs to transform the pixels on the image plane to positions in world coordinates and vice versa. Starting with a discussion on the orientation of a spaceborne sensor, this section is devoted to defining the transformations used in the formulation of the new method. The fundamental frame of reference used in this paper is the Earth-centered inertial (ECI) coordinate system. The ECI is defined by the orthogonal set of unit vectors (i_x, i_y, i_z). In a multisensor scenario, sensor platforms will typically have a sensor reference frame associated with them (measurement frame of the sensor) defined by the orthogonal set of unit vectors (i_{xs}, i_{ys}, i_{zs}). The origin of the measurement frame of the sensor is a translation of the ECI origin, and its axes are rotated with respect to the ECI axes. The rotation between these frames can be described by a set of Euler angles. We will refer to these angles $\alpha+\alpha^{\mathrm{n}}$, $\epsilon+\epsilon^{\mathrm{n}}$, $\rho+\rho^{\mathrm{n}}$ of the sensor, as pitch, yaw, and roll, respectively, where α^{n} is the nominal pitch angle, α is the pitch bias, etc. Each angle defines a rotation about a prescribed axis in order to align the sensor frame axes with the ECI axes. The xyz rotation sequence is chosen, which is accomplished by first rotating about the x axis by α^{n}, then rotating about the y axis by ϵ^{n}, and finally rotating about the z axis by ρ^{n}. The rotation sequence can be expressed as

$$
\begin{aligned}
T(\rho^{\mathrm{n}}, \epsilon^{\mathrm{n}}, \alpha^{\mathrm{n}}) &= T_z(\rho^{\mathrm{n}})T_y(\epsilon^{\mathrm{n}})T_x(\alpha^{\mathrm{n}}) \\
&\triangleq \begin{bmatrix} \cos\rho^{\mathrm{n}}\cos\epsilon^{\mathrm{n}} & \cos\rho^{\mathrm{n}}\sin\epsilon^{\mathrm{n}}\sin\alpha^{\mathrm{n}} - \sin\rho^{\mathrm{n}}\cos\alpha^{\mathrm{n}} & \cos\rho^{\mathrm{n}}\sin\epsilon^{\mathrm{n}}\cos\alpha^{\mathrm{n}} + \sin\rho^{\mathrm{n}}\sin\alpha^{\mathrm{n}} \\ \sin\rho^{\mathrm{n}}\cos\epsilon^{\mathrm{n}} & \sin\rho^{\mathrm{n}}\sin\epsilon^{\mathrm{n}}\sin\alpha^{\mathrm{n}} + \cos\rho^{\mathrm{n}}\cos\alpha^{\mathrm{n}} & \sin\rho^{\mathrm{n}}\sin\epsilon^{\mathrm{n}}\cos\alpha^{\mathrm{n}} - \cos\psi_s\sin\alpha^{\mathrm{n}} \\ -\sin\epsilon^{\mathrm{n}} & \cos\epsilon^{\mathrm{n}}\sin\alpha^{\mathrm{n}} & \cos\epsilon^{\mathrm{n}}\cos\alpha^{\mathrm{n}} \end{bmatrix}. \qquad (1)
\end{aligned}
$$

Assume there are N_S synchronized passive sensors with known positions in ECI coordinates, $\mathbf{sp}_s(k) = [e_s(k), n_s(k), u_s(k)]'$, $s = 1, 2, \ldots, N_S$, $k = 0, 1, 2, \ldots, K$, tracking a single target at unknown positions $\mathbf{x}(k) = [x(k), y(k), z(k)]'$, also in ECI coordinates. With the previous convention, the operations needed to transform the position of the target location expressed in ECI coordinates into the sensor s coordinate system (based on its nominal orientation) is

$$
\begin{aligned}
\mathbf{x}_s^{\mathrm{n}}(k) &= T(\omega_s(k))(\mathbf{x}(k) - \mathbf{sp}_s(k)) \\
&\qquad s = 1, 2, \ldots, N_S, \quad k = 0, 1, 2, \ldots, K, \qquad (2)
\end{aligned}
$$

where $\omega_s(k) = [\alpha_s^{\mathrm{n}}(k), \epsilon_s^{\mathrm{n}}(k), \rho_s^{\mathrm{n}}(k)]'$ is the nominal orientation of sensor s, $T(\omega_s(k))$ is the appropriate rotation matrix, and the translation $(\mathbf{x}(k) - \mathbf{sp}_s(k))$ is the difference between the vector position of the target and the vector position of the sensor s, both expressed in ECI coordinates. The superscript "n" in (2) indicates that the rotation matrix is based on the nominal sensor orientation.

A. Measurement Model

In the process of optical imaging, a simplified model of image formation is shown in Fig. 1. In this so-called pinhole camera model, the lens is a single point. A given scene is mapped onto the image plane by projection through the optical center of the imaging lens. We shall define the sensor coordinate system as having the horizontal and vertical axes of the image plane labeled ξ

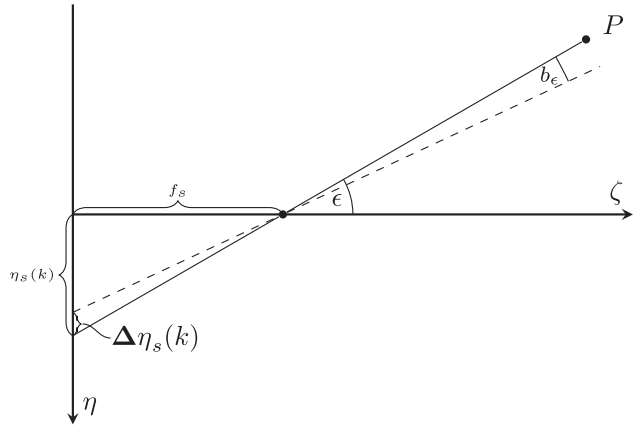


Fig. 2. Sensor coordinates and elevation pointing bias.

and η, respectively, and the "optical axis", labeled ζ (as shown in Figs. 1 and 2). Assume that the origin of the sensor coordinate system is the lens center, and f_s is the focal length of the optical sensor, the distance from the lens focal point to the image plane. The three-dimensional coordinates $(x_s(k), y_s(k), z_s(k))$ of a point target are transformed to the image coordinates $(\xi_s(k), \eta_s(k))$ under perspective projection. Then using the two similar triangles, we can write the $\xi_s(k)$ image coordinate in the focal plane as

$$\xi_s(k) = -f_s \frac{x_s(k)}{z_s(k)}, \tag{3}$$

where f_s is the focal length of sensor s and the negative sign is due to the reversing of the image. Similarly, as shown in Fig. 2, the $\eta_s(k)$ coordinate of the image is given by

$$\eta_s(k) = -f_s \frac{y_s(k)}{z_s(k)}. \tag{4}$$

For our bias estimation algorithm, the target measurements will be generated in pixels $\xi_s(k)$ and $\eta_s(k)$. For convenience, the xzy coordinate system is used, the azimuth angle $\beta_s(k)$ is taken in the sensor xz plane between the sensor z axis and the line of sight to the target, while the elevation angle $\gamma_s(k)$ is the angle taken in the Cartesian plane yz between the z axis and the line of sight to the target, that is,

$$\begin{bmatrix} \beta_s(k) \\ \gamma_s(k) \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{x_s(k)}{z_s(k)}\right) \\ \tan^{-1}\left(-\frac{y_s(k)}{\sqrt{x_s^2(k)+z_s^2(k)}}\right) \end{bmatrix}. \tag{5}$$

Assuming a small clockwise roll of b_ρ about the ζ axis, the resulting tilted (rotated) image has the focal plane coordinates

$$\xi_s'(k) = \xi_s(k)\cos b_{\rho_s} + \eta_s(k)\sin b_{\rho_s}, \tag{6}$$

$$\eta_s'(k) = -\xi_s(k)\sin b_{\rho_s} + \eta_s(k)\cos b_{\rho_s}, \tag{7}$$

where $\xi_s'(k)$ and $\eta_s'(k)$ are the resulting pixel positions after the rolling, and $\xi_s(k)$ and $\eta_s(k)$ are the ideal image positions. As shown in Fig. 1, the azimuth bias b_{α_s} of sensor s results in a horizontal bias in pixels of

$$\begin{aligned} \Delta\xi_s(k) &= \frac{f_s \sin b_{\alpha_s}}{\cos\alpha_s(k)\cos(\alpha_s(k) - b_{\alpha_s})} \\ &= \frac{f_s \sin b_{\alpha_s}}{\cos\alpha_s(k)(\cos\alpha_s(k)\cos b_{\alpha_s} + \sin\alpha_s(k)\sin b_{\alpha_s})} \\ &= \frac{f_s \sin b_{\alpha_s}}{\cos^2\alpha_s(k)\cos b_{\alpha_s} + \cos\alpha_s(k)\sin\alpha_s(k)\sin b_{\alpha_s}} \\ &= \frac{f_s}{\cos^2\alpha_s(k)\frac{\cos b_{\alpha_s}}{\sin b_{\alpha_s}} + \cos\alpha_s(k)\sin\alpha_s(k)} \\ &= \frac{f_s}{\cos^2\alpha_s(k)\cot b_{\alpha_s} + \cos\alpha_s(k)\sin\alpha_s(k)} \\ &\qquad s = 1, 2, \ldots, N_s. \end{aligned} \tag{8}$$

Similarly, as shown in Fig. 2, the elevation bias b_{ϵ_s} results in a vertical bias in pixels of

$$\begin{aligned} \Delta\eta_s(k) &= \frac{f_s \sin b_{\epsilon_s}}{\cos\epsilon_s(k)\cos(\epsilon_s(k) - b_{\epsilon_s})} \\ &= \frac{f_s}{\cos^2\epsilon_s(k)\cot b_{\epsilon_s} + \cos\epsilon_s(k)\sin\epsilon_s(k)}. \end{aligned} \tag{9}$$

The focal length is related to the horizontal field of view $2\alpha_{\max}$ and N_ξ, the number of pixels along the horizontal ξ axis.

$$f_s = \frac{1}{2} N_\xi \frac{1}{\tan\alpha_{\max}}. \tag{10}$$

Combining (7)–(10), the noiseless biased measurements of the target from sensor s in pixels are

$$\begin{aligned} \xi_s^b(k) = {} & \xi_s(k)\cos b_{\rho_s} \\ & - \frac{f_s}{\cos^2\alpha_s(k)\cot b_{\alpha_s} + \cos\alpha_s(k)\sin\alpha_s(k)} \\ & + \eta_s(k)\sin b_{\rho_s}, \end{aligned} \tag{11}$$

$$\begin{aligned} \eta_s^b(k) = {} & \eta_s(k)\cos b_{\rho_s} \\ & - \frac{f_s}{\cos^2\epsilon_s(k)\cot b_{\epsilon_s} + \cos\epsilon_s(k)\sin\epsilon_s(k)} \\ & - \xi_s(k)\sin b_{\rho_s}, \end{aligned} \tag{12}$$

where $\xi_s(k)$ and $\eta_s(k)$ are the ideal image pixel positions. b_{α_s}, b_{ϵ_s}, and b_{ρ_s} are the azimuth, elevation, and roll biases, respectively. The model for the biased noise free focal-plane measurements expressed in pixels is

then

$$\mathbf{h}\left(\mathbf{x}_s(k), \mathbf{b}_s\right) = \begin{bmatrix} \boldsymbol{\xi}_s^b(k) \\ \eta_s^b(k) \end{bmatrix} = \begin{bmatrix} \xi_s(k)\cos b_{\rho_s} - \frac{f_s}{\cos^2\alpha_s(k)\cot b_{\alpha s}+\cos\alpha_s(k)\sin\alpha_s(k)} + \eta_s(k)\sin b_{\rho_s} \\ \eta_s(k)\cos b_{\rho_s} - \frac{f_s}{\cos^2\epsilon_s(k)\cot b_{\epsilon_s}+\cos\epsilon_s(k)\sin\epsilon_s(k)} - \xi_s(k)\sin b_{\rho_s} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{f_s x_s(k)}{z_s(k)}\cos b_{\rho_s} - \frac{f_s}{\frac{z_s^2(k)}{x_s^2(k)+z_s^2(k)}\cot b_{\alpha_s}+\frac{x_s(k)z_s(k)}{x_s^2(k)+z_s^2(k)}} - \frac{f_s y_s(k)}{z_s(k)}\sin b_{\rho_s} \\ -\frac{f_s y_s(k)}{z_s(k)}\cos b_{\rho_s} - \frac{f_s}{\frac{x_s^2(k)+z_s^2(k)}{x_s^2(k)+y_s^2(k)+z_s(k)}\cot b_{\epsilon_s}+\frac{z_s(k)\sqrt{x_s^2(k)+z_s^2(k)}}{x_s^2(k)+y_s^2(k)+z_s^2(k)}} + \frac{f_s x_s(k)}{z_s(k)}\sin b_{\rho_s} \end{bmatrix}$$

$$= f_s \begin{bmatrix} -\frac{x_s(k)}{z_s(k)}\cos b_{\rho_s} - \frac{y_s(k)}{z_s(k)}\sin b_{\rho_s} - \frac{x_s^2(k)+z_s^2(k)}{z_s^2(k)\cot b_{\alpha_s}+x_s(k)z_s(k)} \\ -\frac{y_s(k)}{z_s(k)}\cos b_{\rho_s} + \frac{x_s(k)}{z_s(k)}\sin b_{\rho_s} - \frac{x_s^2(k)+y_s^2(k)+z_s^2(k)}{(x_s^2(k)+z_s^2(k))\cot b_{\epsilon_s}+z_s(k)\sqrt{x_s^2(k)+z_s^2(k)}} \end{bmatrix}, \tag{13}$$

where $[\xi_s(k)\ \ \eta_s(k)]'$ is the focal-plane position of the image of the target seen by sensor s, $\mathbf{x}_s(k) = [x_s(k), y_s(k), z_s(k)]$ is the target position, and $\mathbf{b}_s = [b_{\alpha_s}\ b_{\epsilon_s}\ b_{\rho_s}]'$ is the bias vector of sensor s.

At time k, each sensor provides the noisy measurements

$$\mathbf{z}_s(k) = \mathbf{h}_s\left(\mathbf{x}_s(k), \mathbf{b}_s\right) + \mathbf{w}_s(k), \tag{14}$$

Let $\mathbf{z}$ be an augmented vector consisting of the batch-stacked measurements from all the sensors up to time K

$$\mathbf{z} = [z_1(1), z_2(1), \ldots, z_{N_S}(1), \ldots, z_1(K), z_2(K), \ldots, z_{N_S}(K)], \tag{15}$$

and

$$\mathbf{w}_s(k) = \left[w_s^{\xi}(k), w_s^{\eta}(k)\right]'. \tag{16}$$

The measurement noises $\mathbf{w}_s(k)$ are zero-mean, white Gaussian with

$$R_s = \begin{bmatrix} (\sigma_s^{\xi})^2 & 0 \\ 0 & (\sigma_s^{\eta})^2 \end{bmatrix} \quad s = 1, 2, \ldots, N_S \tag{17}$$

and are assumed mutually independent.

The problem is to estimate the bias vectors for all sensors and the state vector (position and velocity) of the target of opportunity, i.e.,

$$\theta = [x(K), y(K), z(K), \dot{x}(K), \dot{y}(K), \dot{z}(K), \mathbf{b}_1', \ldots, \mathbf{b}_{N_S}']' \tag{18}$$

from

$$\mathbf{z} = \mathbf{h}(\theta) + \mathbf{w}, \tag{19}$$

where

$$\mathbf{h}(\theta) = [h_{11}(\theta)', h_{21}(\theta)', \ldots, h_{N_S 1}(\theta)', \ldots, h_{1K}(\theta)', h_{2K}(\theta)', \ldots, h_{N_S K}(\theta)']', \tag{20}$$

$$\mathbf{w} = [\mathbf{w}_1(1)', \mathbf{w}_2(1)', \ldots, \mathbf{w}_{N_S}(1)', \ldots, \mathbf{w}_1(K)', \mathbf{w}_2(K)', \ldots, \mathbf{w}_{N_S}(K)']', \tag{21}$$

and the covariance of the stacked process noise (21) is the $(N_S K \times N_S K)$ block-diagonal matrix

$$R = \begin{bmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & R_{N_S} \end{bmatrix}. \tag{22}$$

We shall obtain the maximum likelihood (ML) estimate of the augmented parameter vector (18), consisting of the (unknown) target position, velocity, and sensor biases, by maximizing the likelihood function (LF) of θ based on $\mathbf{z}$

$$\Lambda(\theta; \mathbf{z}) = p\left(\mathbf{z}|\theta\right), \tag{23}$$

where

$$p\left(\mathbf{z}|\theta\right) = |2\pi R|^{-1/2} \exp\left(-\frac{1}{2}\left[\mathbf{z} - \mathbf{h}\left(\theta\right)\right]' R^{-1}\left[\mathbf{z} - \mathbf{h}\left(\theta\right)\right]\right), \tag{24}$$

and $\mathbf{h}$ is defined in (20)

The ML estimate (MLE) is then

$$\hat{\theta}(\mathbf{z})^{\mathrm{ML}} = \arg\max_{\theta} \Lambda(\theta; \mathbf{z}). \tag{25}$$

In order to find the MLE, one has to solve a nonlinear least squares problem. This will be done using a numerical search via the batch iterated least squares (ILS) technique.

B. Space Target Dynamics

The state-space model for a noiseless discrete-time system[1] is of the general form

$$\mathbf{x}(k+1) = f[\mathbf{x}(k), \mathbf{u}(k)] \quad k = 0, 1, 2, \ldots, K-1. \tag{26}$$

With small time steps ($\leq$10 s), we can approximate the motion model with the discrete-time dynamic equation

$$\mathbf{x}(k+1) = F\mathbf{x}(k) + G\mathbf{u}(k), \tag{27}$$

[1]Since we are dealing with exoatmospheric motion, it is reasonable to assume that it is noiseless.

where

$$\mathbf{x}(k) = [x(k), y(k), z(k), \dot{x}(k), \dot{y}(k), \dot{z}(k)]', \quad k = 0, 1, 2, \ldots, K \tag{28}$$

is the six-dimensional state vector at time k, F is the state-transition matrix, and $\mathbf{u}$ is a known input representing the gravitational effects acting on the target [given in (31)]. The state-transition matrix for a target with acceleration due to gravity is

$$F = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{29}$$

and the known input gain matrix (multiplying the appropriate components of the gravity vector) is

$$G = \begin{bmatrix} \Delta t^2/2 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 \\ 0 & 0 & \Delta t^2/2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix}, \tag{30}$$

where Δt is the sampling interval. The gravity term is given by

$$\mathbf{u}(k) = g\frac{\mathbf{x}_{\mathrm{p}}(k)}{a(\mathbf{x}_{\mathrm{p}}(k))}, \tag{31}$$

where $\mathbf{x}_{\mathrm{p}}$ is the position part of the state $\mathbf{x}$ in (28), $g = 9.8\ \mathrm{m/s^2}$, and

$$a = \sqrt{x(k)^2 + y(k)^2 + z(k)^2} \tag{32}$$

is the distance from the target to the origin of the coordinates system. For simplicity, we assume g to be constant. The ratio $\mathbf{x}_{\mathrm{p}}/a$ yields the time-varying components of the gravity acting on the target and provides the scaling factor for the gravity term. Note that in view of (31), the state model (27) is not linear.

C. Bias Estimability

Intuitively, the observability of a system guarantees that the sensor measurements provide sufficient information for estimating the unknown parameters. As discussed in [4], the two requirement for bias estimability are:

First Requirement for Bias Estimability: Each sensor provides a two-dimensional measurement (the two pixel positions of the target in the sensor image) at time K. We assume that each sensor sees the target at all the times $0, 1, 2, \ldots, K$. Stacking together all the measurements results in an overall measurement vector of dimension $2KN_S$. Given that the position, velocity of the target, and bias vectors of each sensor are three-dimensional, and knowing that the number of equations (size of the stacked measurement vector) has to be at least equal to the number of parameters to be estimated (target state and biases), we must have

$$2KN_S \geq 3N_S + 6. \tag{33}$$

This is a necessary condition but not sufficient because (25) has to have a unique solution, i.e., the parameter vector has to be estimable. This is guaranteed by the second requirement.

Second Requirement of Bias Estimability: This is the invertibility of the Fisher information matrix (FIM). In order to have parameter observability, the FIM must be invertible. If the FIM is not invertible (i.e., it is singular), then the CRLB (the inverse of the FIM) will not exist—the FIM will have one or more infinite eigenvalues, which means total uncertainty in a subspace of the parameter space, i.e., ambiguity [2].

For the example of bias estimability discussed in the sequel, estimate the biases of 2 sensors (6 bias components) and 6 target components (3 position and 3 velocity components), i.e., the search is in a 12-dimensional space in order to meet the necessary requirement (33). As stated previously, the FIM must be invertible, so the rank of the FIM has to be equal to the number of parameters to be estimated ($6 + 6 = 12$ in the previous example). The full rank of the FIM is a necessary and sufficient condition for estimability. There exists, however, a subtle unobservability for this example that will necessitate the use of more measurements than the strict minimum number of measurements given by (33).

D. Iterated Least Squares for Maximization of the LF of θ

Given the estimate $\hat{\theta}^j$ after j iterations, the batch ILS estimate after the $(j+1)$th iteration will be

$$\hat{\theta}^{j+1} = \hat{\theta}^j + [(H^j)'R^{-1}H^j]^{-1}(H^j)'R^{-1}[\mathbf{z} - \mathbf{h}(\hat{\theta}^j)], \tag{34}$$

where

$$\mathbf{h}(\hat{\theta}^j) = [h_{11}(\hat{\theta}^j)', h_{21}(\hat{\theta}^j)', \ldots, h_{N_s1}(\hat{\theta}^j)', \ldots, h_{1K}(\hat{\theta}^j)', h_{2K}(\hat{\theta}^j)', \ldots, h_{N_sK}(\hat{\theta}^j)'], \tag{35}$$

where

$$H^j = \left.\frac{\partial \mathbf{h}\left(\theta^j\right)}{\partial \theta}\right|_{\theta=\hat{\theta}^j} \tag{36}$$

is the Jacobian matrix of the vector consisting of the stacked measurement functions (35) w.r.t. (18) evaluated at the ILS estimate from the previous iteration j. In this case, the Jacobian matrix is, with the iteration index omitted for conciseness,

$$H = \left[\, H_{11}\ H_{21}\ H_{N_s1} \cdots H_{1K}\ H_{2K}\ H_{N_sK} \,\right]', \tag{37}$$

where

$$H_s(k) = \begin{bmatrix} \frac{\partial \xi_s(k)}{\partial x(k)} & \frac{\partial \xi_s(k)}{\partial y(k)} & \frac{\partial \xi_s(k)}{\partial z(k)} & \frac{\partial \xi_s(k)}{\partial \dot{x}(k)} & \frac{\partial \xi_s(k)}{\partial \dot{y}(k)} & \frac{\partial \xi_s(k)}{\partial \dot{z}(k)} & \frac{\partial \xi_s(k)}{\partial b_{\alpha_1}} & \frac{\partial \xi_s(k)}{\partial b_{\epsilon_1}} & \frac{\partial \xi_s(k)}{\partial b_{\rho_1}} & \cdots & \frac{\partial \xi_s(k)}{\partial b_{\alpha_{N_S}}} & \frac{\partial \xi_s(k)}{\partial b_{\epsilon_{N_S}}} & \frac{\partial \xi_s(k)}{\partial b_{\rho_{N_S}}} \\ \frac{\partial \eta_s(k)}{\partial x(k)} & \frac{\partial \eta_s(k)}{\partial y(k)} & \frac{\partial \eta_s(k)}{\partial z(k)} & \frac{\partial \eta_s(k)}{\partial \dot{x}(k)} & \frac{\partial \eta_s(k)}{\partial \dot{y}(k)} & \frac{\partial \eta_s(k)}{\partial \dot{z}(k)} & \frac{\partial \eta_s(k)}{\partial b_{\epsilon_1}} & \frac{\partial \eta_s(k)}{\partial b_{\epsilon_1}} & \frac{\partial \eta_s(k)}{\partial b_{\rho_1}} & \cdots & \frac{\partial \eta_s(k)}{\partial b_{\epsilon_{N_S}}} & \frac{\partial \eta_s(k)}{\partial b_{\epsilon_{N_S}}} & \frac{\partial \eta_s(k)}{\partial b_{\rho_{N_S}}} \end{bmatrix}, \quad (38)$$

The appropriate partial derivatives, in pixel measurements, with respect to the target position and velocity components are given in the appendix.

E. Initial Solution

Assuming that the biases are null, the LOS measurements from the first and second sensors $\alpha_1(k), \alpha_2(k)$, and $\epsilon_1(k)$ can be used to solve for each initial Cartesian target position in ECI coordinates using (39)–(41). The two Cartesian positions formed from (39) to (41) can then be differenced to provide an approximate velocity. This procedure is analogous to two-point differencing [2] and will provide a full six-dimensional state to initialize the ILS algorithm.

$$x(k)^0 = \frac{y_2(k) - y_1(k) + x_1(k)\tan\alpha_1(k) - x_2(k)\tan\alpha_2(k)}{\tan\alpha_1(k) - \tan\alpha_2(k)}, \quad (39)$$

$$y(k)^0 = \frac{\tan\alpha_1(k)\,(y_2(k) + \tan\alpha_2(k)\,(x_1(k) - x_2(k))) - y_1(k)\tan\alpha_2(k)}{\tan\alpha_1(k) - \tan\alpha_2(k)}, \quad (40)$$

$$z(k)^0 = z_1(k) + \tan\epsilon_1(k) \times \left| \frac{(y_1(k) - y_2(k))\cos\alpha_2(k) + (x_2(k) - x_1(k))\sin\alpha_2(k)}{\sin(\alpha_1(k) - \alpha_2(k))} \right| \quad k = 1, 2, \ldots, K. \quad (41)$$

The CRLB provides a lower bound on the covariance matrix of an unbiased estimator [2] as

$$E\{(\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}})(\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}})'\} \geq J(\boldsymbol{\Theta})^{-1}, \quad (42)$$

where $\boldsymbol{\Theta}$ is the true parameter vector to be estimated, $\hat{\boldsymbol{\Theta}}$ is the estimate, and J is the FIM given as

$$J(\boldsymbol{\Theta}) = E\left\{[\nabla_{\boldsymbol{\Theta}} \ln \Lambda(\boldsymbol{\Theta})]\,[\nabla_{\boldsymbol{\Theta}} \ln \Lambda(\boldsymbol{\Theta})]'\right\}\Big|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}_{\text{true}}} = \Delta'\left(R^{-1}\right)\Delta\Big|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}_{\text{true}}}, \quad (43)$$

where Δ is given by (37) and R given by (22).

F. Statistical Test for Efficiency With Monte Carlo Runs

As discussed in [2], the normalized estimation error squared (NEES) for the parameter $\boldsymbol{\Theta}$ (under the hypothesis of efficiency), defined as

$$\gamma_{\boldsymbol{\Theta}} = (\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}})' P^{-1} (\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}) = (\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}})' J(\boldsymbol{\Theta})(\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}) \quad (44)$$

is Chi-square distributed with $n_{\boldsymbol{\Theta}}$ (the dimension of $\boldsymbol{\Theta}$) degrees of freedom, assuming that estimation errors are Gaussian, that is,

$$\gamma_{\boldsymbol{\Theta}} \sim \chi^2_{n_{\boldsymbol{\Theta}}}. \quad (45)$$

The hypothesis test whether efficiency can be accepted, i.e., that $P = J^{-1}$, is discussed in [2] and outlined next. The NEES is used in simulations to check whether the estimator is efficient. In practice, to check the estimator efficiency, we use the sample average NEES from N independent Monte Carlo runs, defined as

$$\bar{\gamma}_{\boldsymbol{\Theta}} = \frac{1}{N}\sum_{i=1}^{N} \gamma^i_{\boldsymbol{\Theta}}. \quad (46)$$

The quantity $N\bar{\gamma}_{\boldsymbol{\Theta}}$ is Chi-square distributed with $Nn_{\boldsymbol{\Theta}}$ degrees of freedom.

III. SIMULATIONS

We simulate a space-based tracking system tracking a ballistic missile. The missile and satellite trajectories are generated using STK.[2] The target modeled represents a ballistic missile with a flight time of about 20 min. STK provides the target and sensor positions in three-dimensional Cartesian coordinates at 1 s intervals. The target launch time is chosen so that the satellite sensors are able to follow the missile's trajectory throughout its flight path. The missile and satellite trajectories represent 5 min of flight time (exoatmospheric).

A. Three-Sensor Case

We simulated three space-based imaging sensors at various known orbits, observing a target of opportunity at unknown locations. In this case, a 15-dimensional parameter vector is to be estimated. The horizontal and vertical fields of view of each sensor are assumed to be 60°. The measurement noise standard deviation σ_s (identical across sensors for both horizontal and vertical axes of the image plane ξ and η measurements, $\sigma_s^{\xi} = \sigma_s^{\eta} = \sigma_s$) was assumed to be 0.3 pixel. As shown in Fig. 3, these satellite orbits enabled maximum visibility of the missile trajectory from multiple angles. As discussed in the previous section, the three sensor biases are roll, pitch, and yaw angle offsets. Table I summarizes the bias values (in mrad).

[2]STK is registered trademark of Analytical Graphics, Inc.

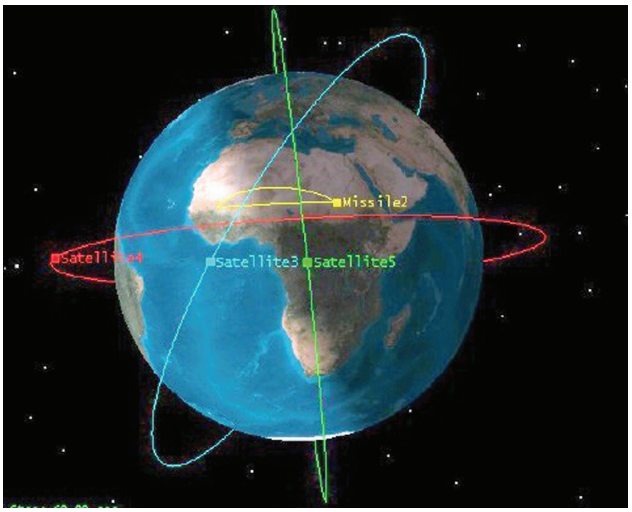


Fig. 3. Target and satellite trajectories for the three-sensor case.

B. Statistical Efficiency of the Estimates for the Three-Sensor Case

In order to test for the statistical efficiency of the estimate [of the 15-dimensional vector (18)], the NEES [2] is used, with the CRLB as the covariance matrix. The sample average NEES over 100 Monte Carlo runs calculated using the FIM evaluated at the true bias values and target locations is approximately 14.3, and the sample average NEES calculated using the FIM evaluated at the estimated biases and target locations is approximately 14.6, and both fall in the interval given below. According to the CRLB, the FIM has to be evaluated at the true parameter. Since this is not available in practice, however, it is useful to evaluate the FIM also at the estimated parameter, the only one available in real-world implementations [11], [12]. The 95% probability region for the 100-sample average NEES of the 15-dimensional parameter vector is [13.95, 16.09]. This NEES is found to be within this interval, and the MLE is therefore statistically efficient. Fig. 4 shows the individual bias component NEES. The 95% probability region for the 100-sample average single component NEES is [0.74, 1.29]. The NEES values are found to be within this interval.

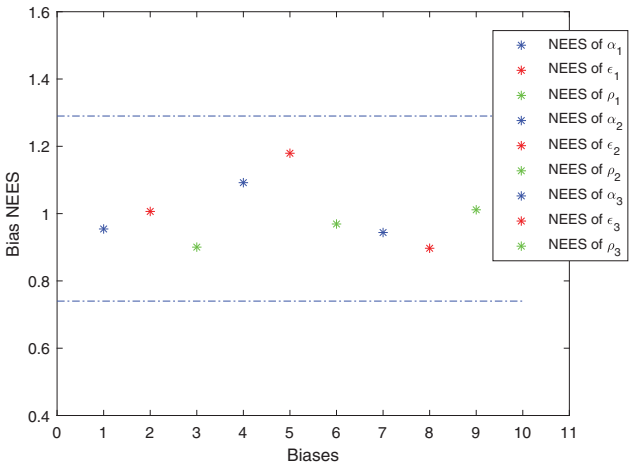


Fig. 4. Sample average bias NEES (CRLB evaluated at the estimate), for each of the nine biases, over 100 Monte Carlo runs (three-sensor case).

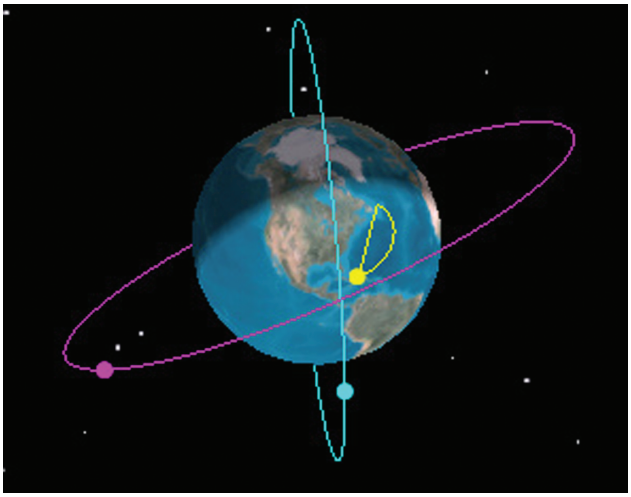


Fig. 5. Target and satellite trajectories for the two-sensor case.

Table I
Sensor Biases (mrad) for the Three-Sensor Case

	α	ϵ	ρ
Sensor 1	2.90	2.80	3.40
Sensor 2	3.33	2.90	3.00
Sensor 3	3.03	3.00	2.90

C. Two-Sensor Case

We simulated two satellite-based imaging sensors at various locations, observing a single target of opportunity. The sensor satellites are in circular orbits of 550 km and 675 km altitude with 0° and 45° inclination, respectively. The horizontal and vertical fields of view of each sensor are assumed to be 60°. The measurement noise standard deviation σ_s (identical across sensors for both horizontal and vertical axes of the image plane ξ and η measurements, $\sigma_s^{\xi} = \sigma_s^{\eta} = \sigma_s$) was assumed to be 0.3 pixel. As shown in Fig. 5, these satellite orbits enabled maximum visibility of the missile trajectory from multiple angles. As discussed in the previous section, the three sensor biases were pitch, yaw, and roll angle offsets. Table II summarizes the bias values (in mrad).

Table II
Sensor Biases (mrad) for the Two-Sensor Case

	α	ϵ	ρ
Sensor 1	2.90	2.80	3.40
Sensor 2	3.33	2.90	3.00

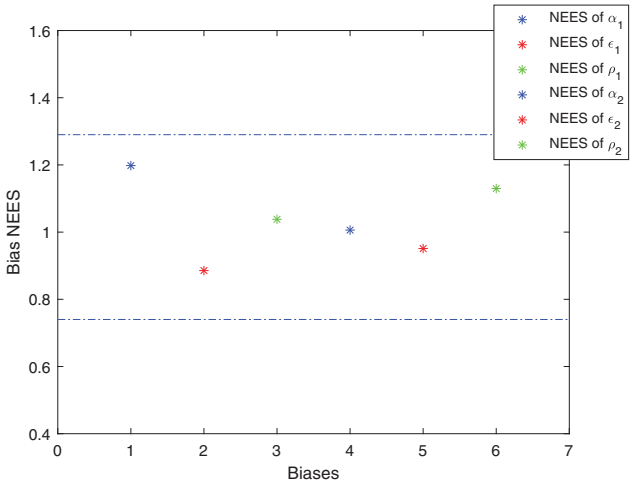


Fig. 6. Sample average bias NEES for each of the six biases.

D. Statistical Efficiency of the Estimate for the Two-Sensor Case

In order to test for the statistical efficiency of the estimate [of the 12-dimensional vector (18)], the NEES is used, with the CRLB as the covariance matrix. The sample average NEES over 100 Monte Carlo runs calculated using the FIM evaluated at the true bias values, target position, and velocity is approximately 11.24, and the sample average NEES calculated using the FIM evaluated at the estimated biases, target position, and velocity is approximately 11.45, both fall in the interval given below. According to the CRLB, the FIM has to be evaluated at the true parameter. Since this is not available in practice, however, it is useful to evaluate the FIM also at the estimated parameter, the only one available in real world implementations [12]. The results are practically identical regardless of which values are chosen for the evaluation of the FIM. The 95% probability region for the 100-sample average NEES of the 12-dimensional parameter vector is [11.20, 12.81]. This NEES is found to be within this interval, and the MLE is therefore statistically efficient. Fig. 6 and Table III show the individual bias component, NEES. The 95% probability region for the 100-sample average single component NEES is [0.74, 1.29]. These NEES are found to be within this interval.

IV. CONCLUSIONS

In this paper, we presented an algorithm that uses a single target of opportunity for the estimation of measurement biases. The first step was deriving a general bias model for synchronized imaging sensors. Based on this derivation, we formulated a nonlinear least-squares estimation scheme for concurrent estimation of the Cartesian position and the velocity in three-dimensional of the target and the angle biases of the sensors. The ILS estimate was shown to be a statistically efficient estimate, and the residual biases are negligible in view of the measurement noise. As such, the covariance matrix from the CRLB can be used as the measurement noise covariance matrix for the resulting composite measurement in a tracking filter. This composite measurement can then be used (with a linear measurement equation) for dynamic state estimation, where position measurements in Cartesian space are preferable to pixel measurements.

Table III
Sample Average Bias NEES (CRLB Evaluated at the Estimate), for Each of the Six Biases, Over 100 Monte Carlo Runs

Biases	α_1	ϵ_1	ρ_1	α_2	ϵ_2	ρ_2
NEES	1.2461	0.9891	1.2043	1.0711	1.0430	0.9734

APPENDIX

The partial derivatives of (38), in pixel measurements, with respect to the target position and velocity components are

$$\frac{\partial \xi_s(k)}{\partial x_s(k)} = -\frac{\cos b_{\rho_s}}{z_s(k)} - \frac{2x_s(k)}{q_1} + \frac{z_s(k)(x_s^2(k)+z_s^2(k))}{{q_1}^2}$$

$$\frac{\partial \xi_s(k)}{\partial y_s(k)} = -\frac{\sin b_{\rho_s}}{z_s(k)}$$

$$\frac{\partial \xi_s(k)}{\partial z_s(k)} = \frac{x_s(k)\cos b_{\rho_s} + y_s(k)\sin b_{\rho_s}}{z_s^2(k)} - \frac{2z_s(k)}{q_1} + \frac{(x_s^2(k)+z_s^2(k))(x_s(k)+2z_s(k)\cot b_{\alpha_s})}{q_1^2}$$

$$\frac{\partial \xi_s(k)}{\partial \dot{x}_s(k)} = \Delta t \frac{\partial \xi_s(k)}{\partial x_s(k)}$$

$$\frac{\partial \xi_s(k)}{\partial \dot{y}_s(k)} = 0$$

$$\frac{\partial \xi_s(k)}{\partial \dot{z}_s(k)} = \Delta t \frac{\partial \xi_s(k)}{\partial z_s(k)}$$

$$\frac{\partial \xi_s(k)}{\partial b_{\alpha_s}} = -\frac{z_s^2(k)(x_s^2(k)+z_s^2(k))}{q_1^2}$$

$$\frac{\partial \xi_s(k)}{\partial b_{\rho_s}} = \frac{x_s(k)\sin b_{\rho_s} + y_s(k)\cos b_{\rho_s}}{z_s(k)}$$

$$\frac{\partial \eta_s(k)}{\partial x_s(k)} = \frac{\sin b_{\rho_s}}{z_s(k)} - \frac{x_s(k)}{r^2 q_2} + \frac{r^2\left(2x_s(k)\cot b_{\epsilon_s} + \frac{x_s(k)z_s(k)}{\sqrt{x_s^2(k)+z_s^2(k)}}\right)}{{q_2}^2}$$

$$\frac{\partial \eta_s(k)}{\partial y_s(k)} = -\frac{\cos b_{\rho_s}}{z_s(k)} - \frac{y_s}{r^2 q_2}$$

$$\frac{\partial \eta_s(k)}{\partial z_s(k)} = \frac{y_s(k)\cos b_{\rho_s} - x_s(k)\sin b_{\rho_s}}{z_s^2(k)}$$

$$+\frac{r^2\left(2z_s(k)\cot b_{\epsilon_s}+\sqrt{x_s^2(k)+z_s^2(k)}+\frac{zs^2}{\sqrt{x_s^2(k)+z_s^2(k)}}\right)}{{q_2}^2}$$

$$-\frac{z_s(k)}{r^2 q_2}$$

$$\frac{\partial \eta_s(k)}{\partial b_{\epsilon_s}}=-\frac{r^2(x_s^2(k)+z_s^2(k))(\cot b_{\epsilon_s}^2+1)}{{q_2}^2}$$

$$\frac{\partial \eta_s(k)}{\partial b_{\rho_s}}=\frac{x_s(k)\cos b_{\rho_s}+y_s(k)\sin b_{\rho_s}}{z_s^2(k)}$$

$$\frac{\partial \eta_s(k)}{\partial \dot{x}_s(k)}=\Delta t\frac{\partial \eta_s(k)}{\partial x_s(k)}$$

$$\frac{\partial \eta_s(k)}{\partial \dot{y}_s(k)}=\Delta t\frac{\partial \eta_s(k)}{\partial y_s(k)}$$

$$\frac{\partial \eta_s(k)}{\partial \dot{z}_s(k)}=\Delta t\frac{\partial \eta_s(k)}{\partial z_s(k)},$$

where

$$q_1=\cot b_{\alpha_s}z_s^2(k)+x_s(k)z_s(k),$$

and

$$q_2=\cot b_{\epsilon_s}(x_s^2(k)+z_s^2(k))+z_s(k)\sqrt{x_s^2(k)+z_s^2(k)}$$

REFERENCES

[1] System Tool Kit, registered trademark of Analytical Graphics Inc., Dec. 2022. [Online]. Available: https://www.agi.com

[2] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan
Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software. Hoboken, NJ, USA: Wiley, 2001.

[3] D. Belfadel and Y. Bar-Shalom
"On-orbit calibration of satellite based imaging sensors,"
Proc. SPIE, vol. 10646 Apr. 2018, Art. no. 1064605.

[4] D. Belfadel, Y. Bar-Shalom, and P. Willett
"Simultaneous target state and passive sensors bias estimation,"
in *Proc. 19th Int. Conf. FUSION*, 2016, pp. 1223–1227.

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

[6] D. Belfadel, R. W. Osborne, and Y. Bar-Shalom
"Bias estimation for moving optical sensor measurements with targets of opportunity,"
J. Adv. Inf. Fusion, vol. 10, no. 2, pp. 101–112, Dec. 2015.

[7] T. M. Clemons and K.-C. Chang
"Effect of sensor bias on space-based bearings-only tracker,"
Proc. SPIE, vol. 6968, Aug. 2008, Art. no. 696809.

[8] T. M. Clemons and K.-C. Chang
"Sensor calibration using in-situ celestial observations to estimate bias in space-based missile tracking,"
IEEE Trans. Aerosp. Electron. Syst., vol. 48, no. 2, pp. 1403–1427, Apr. 2012.

[9] B. D. Kragel, S. Danford, S. M. Herman, and A. B. Poore
"Bias estimation using targets of opportunity,"
Proc. SPIE, vol. 6699, Aug. 2007, Art. no. 66991F.

[10] S. D. Ma
"A self-calibration technique for active vision systems,"
IEEE Trans. Robot. Automat., vol. 12, no. 11, pp. 114–120, Feb. 1996.

[11] R. W. Osborne, III and Y. Bar-Shalom
"Statistical efficiency of composite position measurements from passive sensors,"
in *Proc. SPIE Conf. Signal and Data Processing of Small Targets Recognition XX*, #805008, Orlando, FL, May 2011. https://doi.org/10.1117/12.883045

[12] R. W. Osborne, III and Y. Bar-Shalom
"Statistical efficiency of composite position measurements from passive sensors,"
IEEE Tans. Aerosp. Electron. Syst., vol. 49, no. 4, pp. 2799–2806, Oct. 2013.

[13] Z. Zhang
"A flexible new technique for camera calibration,"
IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[14] Z. Zhang
"Camera calibration with one-dimensional objects,"
in *Proc. Eur. Conf. Comput. Vision*, 2002, pp. 161–174.

Djedjiga Belfadel received the Ph.D. degree in electrical and computer engineering from the University of Connecticut, Storrs, CT, USA. in 2015. She is currently an Associate Professor in the Electrical and Biomedical Engineering Department, Fairfield University, Fairfield, CT, USA. Her research traverses a wide breadth of estimation theory, with a specific emphasis on practical applications such as drone navigation and target tracking. Her scholarly contributions extend to space-based infrared (IR) and electro-optical (EO) sensors, signal and image processing, machine learning, and big data. Alongside her research, she is committed to enhancing engineering education and boosting the representation of women and underrepresented groups within the engineering sector.

INTERNATIONAL SOCIETY OF INFORMATION FUSION

ISIF Website: http://www.isif.org

2023 BOARD OF DIRECTORS*

2021–2023	2022–2024	2023–2025
Alta De Waal	Felix Govaers	Gustaf Hendeby
Fredrik Gustafsson	Lyudmila Mihaylova	Wolfgang Koch
Uwe Hanebeck	Paul Thomas	Claire Laudy

*Board of Directors are elected by the members of ISIF for a three year term.

PAST PRESIDENTS

Simon Maskell, 2022	Darin Dunham, 2014	Pierre Valin, 2006
Simon Maskell, 2021	Wolfgang Koch, 2013	W. Dale Blair, 2005
Paulo Costa, 2020	Roy Streit, 2012	Chee Chong, 2004
Paulo Costa, 2019	Joachim Biermann, 2011	Xiao-Rong Li, 2003
Lyudmila Mihaylova, 2018	Stefano Coraluppi, 2010	Yaakov Bar-Shalom, 2002
Lyudmila Mihaylova, 2017	Elisa Shahbazian, 2009	Pramod Varshney, 2001
Jean Dezert, 2016	Darko Musicki, 2008	Yaakov Bar-Shalom, 2000
Darin Dunham, 2015	Erik Blasch, 2007	Jim Llinas, 1999

SOCIETY VISION

The International Society of Information Fusion (ISIF) is the premier professional society and global information resource for multidisciplinary approaches for theoretical and applied information fusion technologies.

SOCIETY MISSION

Advocate

To advance the profession of fusion technologies, propose approaches for solving real-world problems, recognize emerging technologies, and foster the transfer of information.

Serve

To serve its members and engineering, business, and scientific communities by providing high-quality information, educational products, and services.

Communicate

To create international communication forums and hold international conferences in countries that provide for interaction of members of fusion communities with each other, with those in other disciplines, and with those in industry and academia.

Educate

To promote undergraduate and graduate education related to information fusion technologies at universities around the world. Sponsor educational courses and tutorials at conferences.

Integrate

Integrate ideas from various approaches for information fusion, and look for common threads and themes– look for overall principles, rather than a multitude of point solutions. Serve as the central focus for coordinating the activities of world-wide information fusion related societies or organizations. Serve as a professional liaison to industry, academia, and government.

Disseminate

To propagate the ideas for integrated approaches to information fusion so that others can build on them in both industry and academia.

Call for Papers

The Journal of Advances in Information Fusion (JAIF) seeks original contributions in the technical areas of research related to information fusion. Authors are encouraged to submit their manuscripts for peer review http://isif.org/journal.

Call for Reviewers

The success of JAIF and its value to the research community is strongly dependent on the quality of its peer review process. Researchers in the technical areas related to information fusion are encouraged to register as a reviewer for JAIF at http://jaif.msubmit.net. Potential reviewers should notify via email the appropriate editors of their offer to serve as a reviewer.

Journal of Advances in Information Fusion Volume 18 Number 1 June 2023 ISSN 1557-6418