Sensor Bias Estimation and Compensation for Practical Multisensor-Multitarget Tracking

David Schonborn, T. Kirubarajan, R. Tharmarasa

***Abstract*—Estimating and compensating for sensor biases are essential in multisensor-multitarget tracking scenarios, where association of tracks or measurements originating from different sensors is required. Ignoring measurement bias can cause data association to fail and result in substantial performance degradation. Many approaches to estimate and compensate for biases exist, but can only be used under specific operational conditions. In contrast to the assumptions of existing literature, practical sensor networks often operate under demanding conditions including a distributed fusion architecture, time-varying bias, multiple heterogeneous sensors with nonlinear measurement models, limited availability of computational resources, and asynchronous communication without full availability. By restructuring the estimation problem to avoid sources of inconsistency and leveraging flexible tools such as the Unscented Transform and Covariance Intersection, this paper proposes a computationally efficient bias estimation algorithm that can be applied to a broad range of realistic scenarios. Results are presented for a simulated application in multisensor-multitarget terrain-aided video tracking to evaluate the proposed algorithm. These results demonstrate its effectiveness in reducing bias-compensated track RMSE to levels comparable to those of unbiased sensors. This general bias estimation algorithm can be implemented in a wide range of practical systems without significant modification.**

I. Introduction

In multisensor-multitarget tracking the objective is to estimate the state (position, velocity, etc.) of targets using information obtained from multiple sensors. In many practical systems these sensors are subject to two types of measurement errors. The first is zero-mean noise which can be removed effectively through filtering (for example, using a Kalman Filter) [1]. The other type of error is bias which is constant or slowly-varying over time, not zero-mean, and can not always be removed through standard filtering processes [2]. Although bias can negatively affect estimation performance in any tracking problem, it is particularly problematic in multisensor-multitarget tracking where association of measurements or tracks from different sensors is required. This degraded performance can take the form of missed or wrong associations [**?**], formation of ghost tracks [**?**], decreased track accuracy [**?**], inconsistent tracks [**?**], and decreased track probability of detection [**?**]. This motivates the development of algorithms to estimate and compensate for bias errors prior to fusion.

Realistic operational conditions often involve additional challenges such as using a specific fusion architecture, time-varying measurement bias, imperfect knowledge of sensor positions, highly nonlinear measurement models, heterogeneous sensors, heterogeneous state space, targets moving in varied terrain, reduced communication bandwidth or unreliable communication, limitations in computation power, a large (possibly time-varying) number of sensors, or asynchronous data updates. There are many approaches to bias estimation to be found in the existing literature, some that are even optimal under certain conditions, but most address only a subset of the realistic challenges faced by many practical systems. Furthermore, many largely-speaking practical methods are only applicable to certain specific scenarios, making applications to large, heterogeneous sensor networks an exercise in patchwork.

Bias estimation and compensation methods in the existing literature can be broadly divided into three categories. There are methods that use an augmented state vector to simultaneously estimate the state of all targets and sensor biases, others that formulate a pseudomeasurement of the sensor biases that is independent of the target state, and finally those that employ particle methods to estimate various states including sensor biases. Each of these categories is reviewed below with a summary of their strengths and limitations.

Several methods have been proposed to estimate sensor biases by creating an augmented (or stacked) state vector and using, for example, a Kalman Filter (KF) to jointly estimate the states of the targets and sensor biases [3], [4], [5]. In the literature these methods are frequently referred to as Augmented State Kalman Filter (ASKF) methods. Under certain conditions these methods are able to produce an optimal solution [3], [5]. However, if all these state variables are stacked, this can result in a very large state vector if the scenario involves many sensors, targets, and/or bias parameters. Since the KF requires inversion of the covariance matrix, which has computational complexity of approximately $\mathcal{O}(n^3)$ (where n is the state dimension), this can become computationally prohibitive for use in a real time system [3]. Related methods have been proposed which decouple the state vector in various ways, allowing for the use of several KFs of smaller dimension instead of a single larger one, reducing the computational load. In [3], [6] approximations are used to decouple the augmented state, but are subject to performance degradation. In [5] an exact formulation is proposed to decouple the augmented state, allowing the use of multiple KFs for different targets, each one stacked with the sensor bias estimates. Using their approach the state vector of each KF still grows with the number of sensors, so for large sensor networks this can become computationally expensive. A two-stage approach was proposed in [4] to decouple the target states from the sensor biases when the biases are constant, which was extended by [7] to accommodate dynamic biases with white or coloured Gaussian noises. However [7] also point out that this approach is only equivalent to the ASKF under a restrictive constraint which

is not usually satisfied in practice. Additionally, most ASKF algorithms operate under the assumption that all measurements from the sensors are available at the fusion center, and that these measurements are synchronized in time [3], [5], [7]. This makes them only applicable to time-synchronized centralized fusion networks. Extension to nonlinear systems is possible through linearization, for example using the Extended Kalman Filter (EKF), but these require situational derivations and are still sub-optimal [2]. For highly nonlinear systems it may be desirable to use a robust tool such as the Unscented Transform (see section II-B), and these ASKF methods are not directly extensible to accommodate this.

Another popular approach is to formulate a pseudomeasurement of the sensor biases that is independent of the target state by taking the difference between measurements or tracks from different sensors such that the target state cancels out [8], [9], [10], [11]. This leaves just a noisy measurement of the sensor biases that can be used to estimate the biases directly. This strategy has been successfully employed in various fusion architectures [10], [9], [11], [8] and with asynchronous data rates [8], [12]. In most cases the state vector for bias estimation is the stacked vector of the biases from the sensors used to formulate the pseudomeasurement, so it is much more computationally efficient than the ASKF which also stacks all the target states. This state vector still grows with the number of sensors, so for large sensor networks it can still become computationally expensive. Beyond just computational load, many of these methods are limited to use with a pair of sensors [10], [9], [12] so use with a large sensor network is not possible without the additional consideration of pairing sensors. Some methods, such as [8], present computationally efficient solutions with multiple sensors under specific circumstances. Another challenge with pseudomeasurement methods lies in their application to heterogeneous sensor networks. For different sensor types (or different bias models), different formulations are needed and often lead to different solution methodologies. For example, in [8] a system with radar sensors is considered and can be solved using Recursive Least Squares (RLS) or a KF (for constant and time-varying biases, respectively). In [9] a similar starting point is taken but applied to a pair of video sensors, and the resulting solution requires minimizing a nonlinear cost function using the genetic algorithm (although in [10] an alternative method is proposed that applies approximations allowing RLS to be used). Finally, highly nonlinear systems remain a challenge in a similar way as with the AKSF methods. There are solutions that can be applied, frequently using a converted measurement model [8], [9], [10], but requires situational modelling of the bias in the converted measurement model. It may be more desirable for practical implementation purposes in a heterogeneous network if a unified approach could be applied in terms of solution methodology across different models of the measurement and biases.

The final category of methods that have been applied to bias estimation are those that use particle methods such as the particle filter and other related methods that use a number of (usually weighted) particles to represent the distribution of the variables of interest. These approaches are very flexible in handling situations with nonlinear systems or non-Gaussian distributions, but there are also some major drawbacks. The computational load can be substantial, especially if the state space dimension is large, since enough particles must be used to achieve adequate coverage and this number grows exponentially in the number of dimensions. Strategies can be used to reduce the sampling dimension, such as in Rao-Blackwellised Particle Filters (RBPFs) where some of the variables of the augmented state are marginalized out, for example in [13]. However if the number of dimensions that are decidedly required in the state remains large (for example if there are many bias parameters), the problem persists. In addition to this, particle filters are known to suffer from the problems of degeneracy and sample impoverishment [14]. The degeneracy problem refers to the tendency for a large number of particles to have near-zero weights as the filter is updated, requiring a large amount of computational power to be spent updating these particles that are not very relevant to the distribution estimate [14]. Degeneracy can be partly solved by resampling, where samples are redrawn to have equal weights but still represent approximately the same distribution [14]. However this introduces a tuning parameter to set a degeneracy threshold for when resampling should occur, and can also result in worsened sample impoverishment [14]. Sample impoverishment refers to the problem of having many samples which represent either very similar points or the same points, resulting in low sample diversity. This problem is especially prominent when process noise is low, or zero as in the case of constant biases [14]. Approaches have been proposed counteract sample impoverishment by adding some artificial process noise to static parameters [15], [16] but this introduces at least one additional tuning parameter.

Another important aspect of the sensor registration problem is bias compensation. Many existing works simply assume the bias estimate to be sufficiently accurate to correct the bias, and treat the problem of bias compensation by simply adjusting the measured values by subtracting the mean of the estimated bias, assuming the bias estimates to be sufficiently accurate (for example, in [10]). In [17] a bias compensation method accounting for the covariance of the bias estimate was proposed and found to produce more accurate tracks when compared with the simple approach considering only the mean. This suggests that it is not always reasonable to ignore the uncertainty of the bias estimate during bias compensation. The method from [17] requires explicit modeling of the cross-covariance terms in the update step and simplifies them with some approximations (including approximating the cross-covariance between the bias estimate and the state estimate as zero). For methods that perform joint estimation of the states and biases (such as the full ASKF) this is not an issue since these terms are maintained already, but these remain subject to other restrictions (as described above). See section IV-C for more information about bias compensation approaches.

Given the limitations of existing methods discussed above, there remains a need for a unified, flexible bias estimation method that can be applied to large, heterogeneous sensor networks without significant limitations in terms of the fusion architecture, the arrival of incoming data, computational

feasibility, measurement and bias model, or state space model. In this paper a novel algorithm for sensor bias estimation is presented to address this need. Section II provides necessary the background on existing algorithms that are used within the proposed algorithm and simulated scenario. A mathematical formulation of the general problem is given in section III that introduces the relevant notations and concepts. Section IV-D describes the proposed algorithm for bias estimation in detail, along with a discussion of bias compensation and a computational complexity analysis. An example application with required implementation details and corresponding simulation results are presented in section V-E. Section VI discusses these results, their implications, and limitations of the proposed algorithm. A conclusion summarizes the findings of this work in section VII.

II. Background

A. Multisensor-Multitarget Tracking

In multisensor-multitarget tracking problems, a number of sensors are used to track multiple targets (estimate their state, for example their position and velocity). In the general case, both the number of sensors and the number of targets may be time-varying. First, sensors measure the target state according to a measurement model. A general model for a sensor measurement z of (true) target state x at time index k is given in equation 1. The general measurement model includes a vector of bias parameters β that may be constant or time-varying, and measurement noise w. The measurement noise is typically assumed zero-mean with covariance R and independent between different measurements (not auto-correlated over time) [2]. The function h maps the true measured state and bias parameters to the measurement space.

$$z(k) = h(x(k), \beta(k)) + v(k) \tag{1}$$

Each measurement is usually assumed to correspond to a single target, and each sensor is assumed to provide at most one measurement for each target at a single instance in time. These measurements are used to initialize and maintain estimates of the states of targets over time, a process known as filtering. In addition to the measurement model, filtering requires a model of how the target state evolves over time, referred to as a state-transition model [2]. In tracking applications this is often given in the form of a kinematic model. A general state-transition model is given in equation 2 where f is a function that maps the target state $x(k)$ at time k to the target state $x(k+1)$ at time $k+1$. The second argument to the function f indicates that this function may be time-varying. The transition is also subject to process noise w, which typically is assumed to be Gaussian, zero-mean with covariance Q, and independent over time [2]. Some formulations of the state-transition model also include modeling of a known control input, but for non-cooperative tracking this cannot be assumed known and is not included here.

$$x(k+1) = f(x(k), k+1) + w(k+1) \tag{2}$$

In the case of independent zero-mean Gaussian noises, zero measurement bias, and linear models for both the measurement and system, the optimal estimator (in terms of minimum mean-squared error) is the Kalman Filter (KF) [2]. The KF can be implemented efficiently using a recursive formulation, so it is very practical for online applications. Many practical applications do not conform exactly to the optimality requirements of the KF, but none the less it remains the most popular choice as the basis of many tracking systems. Extensions to the KF, such as the Extended Kalman Filter (EKF) [2] and Unscented Kalman Filter (UKF) [18], exist and have proven successful (through numerous applications) in expanding the scope of operation of the Kalman Filter framework. Due to the ubiquity of the KF framework, it is desirable for solutions to various tracking problems be compatible with its use. This compatibility is often achieved through linearization of a nonlinear measurement function (as in the EKF) or through a Gaussian approximation of a random variable (as in the UKF).

When using multiple sensors it becomes necessary to combine (or fuse) their information. There are three main types of fusion architecture that define how information is combined from different sources. These architectures will be reviewed here briefly in terms of their relevant aspects, but for a more thorough treatment the reader is referred to [19]. In centralized fusion architectures, each sensor sends all its measurements back to the central fusion center (CFC) where information is combined at the measurement level (measurement-to-track fusion) [19]. In practical systems a centralized fusion architecture often presents challenges in terms of communication bandwidth, with all sensors needing to transmit all measurements to the CFC to perform fusion [19]. To address this, a distributed fusion architecture can be used. In a distributed architecture, information from measurements is processed into tracks at the local sensor node before transmission to the CFC where the tracks are combined through track-to-track fusion [19]. Transmitting tracks instead of measurements can relax the requirement that communication takes place at every local sensor update and reduce overall bandwidth requirements, but introduces the challenge of fusing correlated information [19]. There is also a decentralized fusion architecture, where there is no one CFC that can communicate with all Local Fusion Centers (LFCs), so instead there are multiple fusion centers (FCs) that can communicate with only a subset of sensor nodes or other FCs [19]. A FC in a distributed architecture can receive information in the form of measurements from sensor nodes or tracks from other FCs [19]. Hierarchical architectures can also be used that perform fusion at various levels, effectively combining different architecture types [19]. With any fusion architecture it is essential to estimate and compensate for sensors biases so that the information to be fused will be consistent. The fusion architecture used will determine what information is feasibly available at the nodes where bias estimation is being performed, and therefor what bias estimation algorithms can be applied.

B. The Unscented Transform

First described in [18], the Unscented Transform (UT) can be used to approximate (as a Gaussian random variable) the

results of transforming a random variable using a nonlinear transformation. Since its introduction the UT has been shown to be effective for this purpose in numerous practical scenarios with a broad scope of application, including [20], [21], [22]. The UT will be used in the proposed algorithm for bias estimation, as described in section IV-A.

The basic approach with the UT is to draw a number of deterministic sigma points from the distribution of the random variable, apply the nonlinear transformation to each sigma point, and then calculate the parameters (mean and covariance) of the transformed distribution based on the transformed sigma points. Paraphrased from [18], the UT algorithm is described in algorithm 1. The notation $(\sqrt{A})_i$ indicates the ith row of the matrix square root of A. The parameter κ is used to control the spread of the sigma points, and n is the number of dimensions in $\bar{x}$.

Algorithm 1 UnscentedTransform($\hat{x},\ f$)

// Compute $2n+1$ sigma points
$\mathcal{X}_0 = \bar{x}$
$W_0 = \kappa/(n+\kappa)$
for each $i \in 1 \dots n$ **do**
  $\mathcal{X}_i = \bar{x} + (\sqrt{(n+\kappa)P_{xx}})_i$
  $W_i = 1/(2(n+\kappa))$
  $\mathcal{X}_{i+n} = \bar{x} - (\sqrt{(n+\kappa)P_{xx}})_i$
  $W_{i+n} = 1/(2(n+\kappa))$
end for
// Transform sigma points
for each $i \in 0 \dots 2n$ **do**
  $\mathcal{Y}_i = f(\mathcal{X}_i)$
end for
// Compute parameters of transformed distribution
$\bar{y} = \sum_{i=0}^{2n} W_i \mathcal{Y}_i$
$P_{yy} = \sum_{i=0}^{2n} W_i (\mathcal{Y}_i - \bar{y})(\mathcal{Y}_i - \bar{y})^T$
return $\hat{y} = \mathcal{N}(\bar{y}, P_{yy})$

This greatly simplifies the nonlinear transformation of the random variable, as all that is necessary is to evaluate the nonlinear function f at each of the sigma points.

C. Information Fusion Under Unknown Cross-Correlation

The Covariance Intersection (CI) algorithm was proposed in [23] as an approach for consistently fusing estimates whose correlations are unknown. The algorithm and variations have since been applied to many problems in multisensor-multitarget tracking, especially in distributed (track-to-track) fusion where tracks from different sensor nodes have unknown correlations [24], [25], [26]. The proposed bias estimation algorithm, described in section IV will make use of the CI algorithm. The fused CI estimate is given by equation 3 from [23], where estimates a, b with means $\bar{a}, \bar{b}$ and covariance matrices P_{aa}, P_{bb} are fused.

$$\begin{aligned} P_{cc} &= (\omega P_{aa}^{-1} + (1-\omega)P_{bb}^{-1})^{-1} \\ \bar{c} &= P_{cc}(\omega P_{aa}^{-1}\bar{a} + (1-\omega)P_{bb}^{-1}\bar{b}) \end{aligned} \tag{3}$$

The CI estimate is consistent for any cross-correlation matrix P_{ab} and any ω in the range $[0, 1]$, given that the estimates being fused are also consistent [23]. The notion of consistency here is that the estimated mean squared error (MSE) matrix P_{cc} does not underestimate the true MSE $\bar{P}_{cc}$. This is expressed in equation 4 from [23].

$$P_{cc} - \bar{P}_{cc} \geq 0 \tag{4}$$

While P_{ab} is assumed entirely unknown, ω is a free parameter that can be chosen to optimize some criteria. Cost functions that are commonly chosen are to minimize the determinant or the trace of P_{cc}, and many different optimization strategies can be used to minimize (or approximately minimize) the cost [23], [27], [28]. The CI estimate is the optimal (in the sense of the chosen cost function) consistent estimate if the cross-correlation is entirely unknown [27]. Furthermore the estimate is non-divergent if a fixed measure of cost is used, in the sense that fusion never makes the estimate worse (according to the cost function) [29]. The CI algorithm is summarized in algorithm 2.

Algorithm 2 CovarianceIntersection($\hat{a},\ \hat{b},\ J$)

// Find the parameter ω to minimize cost function J
$\omega = \arg\min_\omega J,$ subject to $0 \leq \omega \leq 1$
// Compute CI estimate
$\bar{c}, P_{cc}$ = ⟨Use equation 3 or 5⟩
return $\hat{c} = \mathcal{N}(\bar{c}, P_{cc})$

A more general CI equation that can be used to fuse estimates of unequal state dimension is briefly discussed in [29], though it is noted that efficient optimization to find the optimal weight ω can be more challenging. Furthermore, the proof of consistency is only explicitly given (in [23], [29]) for CI with equal state dimensions (as in equation 3). As pointed out in [30], it is also not straightforward to determine an appropriate cost function to determine ω. Nevertheless, fusion of estimates with unequal state dimension under unknown cross-correlation remains an area of active research, so the CI approach to this problem is worth considering. The fused CI estimate for unequal state dimensions is given by equation 5 from [29], where the larger-dimension state estimate $\bar{a}, P_{aa}$ is updated with information from $\bar{b}, P_{bb}$ and H maps the larger state to the smaller state.

$$\begin{aligned} \bar{c} &= \bar{a} + \\ &\quad (1-\omega)P_{aa}H^T((1-\omega)HP_{aa}H^T + \omega P_{bb})^{-1}(\bar{b} - H\bar{a}) \\ P_{cc} &= \frac{1}{\omega}(P_{aa} - \\ &\quad (1-\omega)P_{aa}H^T((1-\omega)HP_{aa}H^T + \omega P_{bb})^{-1}HP_{aa}) \end{aligned} \tag{5}$$

In [31] several alternative approaches are discussed for track-to-track fusion with unequal state dimensions using CI. Their work presents and compares four different methods of augmenting the smaller state so that it is the same size as the larger state, allowing the use of CI in the form given by equation 3. They also compare these approaches across different optimization methods for ω, since the augmented state will affect the optimization problem. A similar approach

also using an augmented state is proposed in [32], with an application to bias estimation. There the authors use a decentralized information filter (rather than CI) to update the state during fusion by augmenting the smaller state with 0 information about the missing components. In [33] a slightly different approach is used, performing CI to fuse the common states only. They assume that the non-common states are not affected by the incoming information about the common state. In both cases the information about the bias is not updated when new information about the common states is fused. None of the approaches presented in [31], [33], [32] allow for any information to be garnered about the non-common state components through cross-correlation with the common state, which is a significant drawback of these approaches.

An approach related to CI, but instead formulated in terms of a bounding covariance matrix, is presented in [34]. This is generalized in [35] to provide a tighter bound in the case where some maximum bound on the correlation coefficient between the estimates to be fused is known. In [36] the covariance bound (CB) approach from [34], [35] is built upon with a specific formulation for the case of fusing state vectors with unequal dimensions. There, a conservative bound on the joint covariance matrix is used to formulate a weighted least squares (WLS) problem that can be solved to yield a fused state estimate of all state components. A similar formulation is also given in [37]. The conservative bounding joint covariance matrix from [36] is given in equation 6. In the original paper this matrix is given for fusing an arbitrary number of estimates where each estimate's covariance block is multiplied by the inverse of its weight and the sum of the weights is one [36]. Here it has been presented in a simplified form for use with just two estimates so that there can be only one weight parameter ω.

$$\begin{aligned} \bar{x}_{CB} &= \begin{bmatrix} \bar{a}^T & \bar{b}^T \end{bmatrix}^T \\ P_{CB} &= \begin{bmatrix} \frac{1}{\omega} P_{aa} & 0 \\ 0 & \frac{1}{1-\omega} P_{bb} \end{bmatrix} \end{aligned} \tag{6}$$

Note that P_{aa} and P_{bb} need not be the same dimension. The WLS problem is then solved for the fused estimate $\bar{c}, P_{cc}$ as follows in equation 7 from [36] where H is used to map the components of each of the state estimates to the joint state consisting of all components.

$$\begin{aligned} K &= (H^T P_{CB}^{-1} H)^{-1} H^T P_{CB}^{-1} \\ \bar{c} &= K x_{CB} \\ P_{cc} &= K P_{CB} K^T \end{aligned} \tag{7}$$

The authors of [36] do not provide details on how to compute the weighting parameters for the covariance matrices, as noted in [30] where an alternative approach is proposed. Their approach is based on an inner ellipsoidal approximation and provides details on how to compute the required weight matrices by solving two semidefinite programming (SDP) problems [30]. Their results showed improved performance when compared with the CB method from [36] with equal weighting of the estimates forming the bounding joint covariance matrix, however it is likely that one could come up with a better weight selection strategy than this for the CB approach. It is also noted that their approach does not guarantee consistency [30].

Other approaches for fusion under unknown correlation exist that leverage a specific problem structure to achieve promising results. One such approach that has been used successfully in track-to-track fusion is sample-based reconstruction of cross-covariances. In [38], [39] this approach is applied with homogeneous state vectors, and in [40], [41] it is generalized to heterogeneous states. The sample-based reconstruction was found to perform similarly to the optimal fusion where cross-covariances are maintained rigorously [38], [39], [40], [41]. Other approaches that also leverage a specific problem structure include [42], [43], [44], [45]. All of these approaches have been shown to produce very promising results when they apply. The downside is that they require a specific correlation structure which limits their scope of application or requires reformulation to apply to new problems.

D. Terrain-Aided Tracking

Some commonly-used sensors (such as video sensors) measure only the angle to a target, and do not directly observe the range to the target. This leaves the target state in Cartesian coordinates not fully observable based on a measurement alone. One option to resolve this issue when tracking ground targets is by following the measured line of sight to some assumed model of the ground, allowing the measurements to be converted to Cartesian coordinates prior to tracking in a converted measurement Kalman Filter framework [46], [20]. This general approach is illustrated in figure 1.

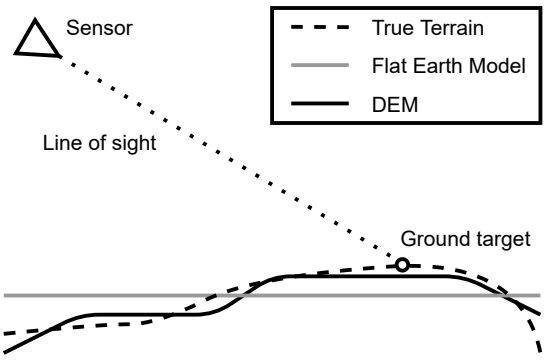

Fig. 1: An illustration of the line of sight method for determining the position of a target in Cartesian coordinates. Neither the flat earth model nor the digital elevation map (DEM) perfectly model the target's position on the true terrain.

When terrain is relatively consistent, a flat earth model or a curved earth model (such as WGS-84) can be appropriate and allow for convenient geometric conversions of the measurement to Cartesian coordinates [46]. These are commonly used in existing bias estimation literature and can allow for simplification of the bias estimation problem [8], [10]. If the terrain is varied in the tracking area but can be accepted as locally flat in the area of the target then a basic geometric measurement conversion can still be applied if the elevation in the area of the target is known [10].

If terrain is more varied, it may be more appropriate to use a Digital Elevation Model (DEM) to model its features more accurately. In this case the measurement conversion is not as straightforward because the shape of the terrain near the target also affects the converted measurement distribution. This is known as terrain-aided tracking, and has been used successfully to improve tracking results in varied terrain [47], [20], [48]. In [47] the UT is used to transform the measurement by transforming each sigma point using a line of sight intersection algorithm. A similar approach, instead using a sampling transform, is used in [48] where the measurement distribution in Cartesian coordinates is assumed to be a Gaussian mixture. The transformations in all of these approaches essentially come down to intersecting individual lines of sight from the sensor to the target with the terrain model (see figure 2) and then using the points of intersection to determine the distribution of the target position in Cartesian coordinates. The specifics of the intersection algorithm can vary depending on the DEM format and implementation decisions, and examples can be found in more detail in [47], [20], [48].

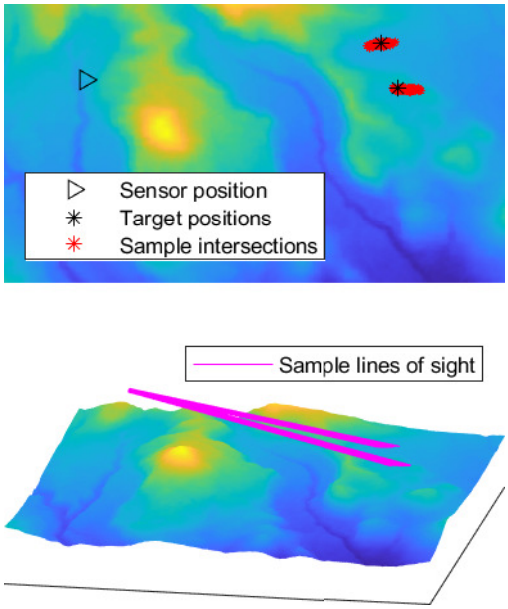

Fig. 2: In terrain-aided tracking sample lines of sight from the sensor in the direction of the target are intersected with a DEM to obtain samples from the distribution of the target positions. In this example the samples are randomly drawn from the measurement distribution, but deterministic samples (such as in the UT) can also be used.

The line of sight intersection algorithm outlined in [48] will be used in the simulated scenario described in section V and will be reviewed here briefly. For more details the reader is referred to [48]. The azimuth and elevation angles from the sensor to the target are first converted into a unit vector in the corresponding direction using the standard spherical to Cartesian conversion. This yields a line of the form given in equation 8 where X is the sensor position, D is the unit vector in the direction of the target, and t is a free parameter moving along the line.

$$L = X + tD \tag{8}$$

All points of intersection between L and the DEM surface X_w can be found by setting them equal, but the specific intersection point of interest is that which is in front of the sensor and nearest to it. Therefor t can be expressed as the solution to an optimization problem, as in equation 9 from [48].

$$\min_{t} \quad t \tag{9a}$$
$$\text{subject to} \quad X + tD = X_w, \tag{9b}$$
$$t > 0 \tag{9c}$$

The solution t is found by traversing the DEM grid until the nearest point of intersection is reached [48]. The grid tracing approach from [49] is used, allowing only the cells of the DEM in the path of the ray to be traversed, as shown in figure 3. As the cells are traversed in order of increasing distance from the sensor, each is tested for intersection first at a coarse level with fast check on the bounding rectangular prism. If the coarse check is passed, the specific shape of the cell (a bilinear interpolation of its corner elevations) is checked for intersection using the algorithm from [50]. When a point of intersection is found, it will be the one nearest to the sensor (see [50]) and the algorithm can be stopped.

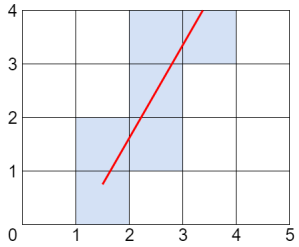

Fig. 3: DEM grid traversal along a line of sight.

Although the use of a DEM can improve tracking performance when simpler terrain models cannot adequately represent the terrain, they are not completely free of errors. One source of error is that the DEMs commonly used for tracking applications are limited in resolution. Computational requirements for online tracking in real time prohibit the use of DEMs with very high resolution, which can additionally are not always available for the tracking area of interest. The NASA SRTM data sets provide DEMs with resolutions of 30m [51] and 90m [52] for most of the planet, so these are reasonable resolutions to assume for practical purposes. Features of the terrain significantly smaller than the resolution of the DEM cannot be represented. The other main source of error in DEMs is elevation errors. Even large terrain features that can be seen given the resolution are not perfectly known in terms of their elevation. Figure 4 shows the level of detail of three different publicly available DEMs (at 1m [53], 30m [51], and 90m [52] resolution) for the area surrounding Crater Lake, Oregon, USA.

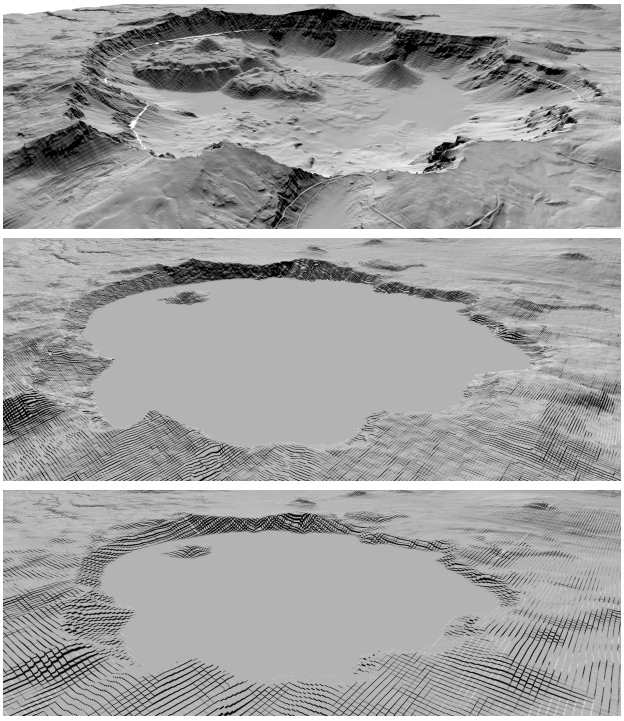

Fig. 4: DEMs of the area surrounding Crater Lake at 1m, 30m, and 90m resolutions, showing the level of detail in each. Note that in this example the 1m model also includes the underwater elevation while the 30m and 90m models just include a flat surface at sea level.

III. Problem Formulation

Assume $M(k)$ targets are observed by $N(k)$ sensors at time t_k. For asynchronous systems (as considered here) it may be the case that $N(k) = 1$, with different sensors observing the targets at different instances in time, globally indexed by k. The sensors may have different measurement models, but all use a measurement model of the form given in equation 1. The notation specific to the problem is made more clear in equation 10.

$$z^m_{s,t}(k) = h^m_s(x^m_s(k), x^m_t(k), \beta_s(k), v_s(k)) \tag{10}$$

The measurement $z^m_{s,t}(k)$ is generated by sensor s observing target t at time k. Here $x^m_s(k)$ and $x^m_t(k)$ are the true states (in the measurement space, noted by superscript m) of the sensor and the target, respectively. The sensor state is included explicitly here to indicate that there may be noisy measurements that relate to the sensor as well (for example, a GPS measurement of the sensor position). Since this is already in the measurement space, the measurement model h^m_s is primarily used to model how the true bias vector β_s enters the measurement equation. This is not restrictive and the bias vector may contain additive (offset) and/or multiplicative (scaling) biases. The measurement noise $v_s(k)$ is assumed Gaussian zero-mean with covariance matrix $R_s(k)$, and is assumed to be independent from other noises considered.

To accommodate heterogeneous sensor models in the measurement space, the measurements are converted into Cartesian coordinates to provide a common frame of reference. In equation 11 the measurement conversion function h^c_s is introduced, which may be highly nonlinear. The bias estimates $\hat{\beta}_s(k)$ are modelled as Gaussian random variables and are accounted for in the conversion process. These may be constant or time-varying according to some state-transition model of the form given by equation 2. The measurement conversion function may also depend on some additional global information z_g such as road map information or terrain data (see section V for an example). This can be considered to be constant (if known completely) or a random variable (if there is some uncertainty).

$$z^c_{s,t}(k) = h^c_s(z^m_s(k), \hat{\beta}_s(k), z_g) \tag{11}$$

The function h^c_s must map each point in the space of its input to a unique point in the state space. The converse need not be true (i.e. h^c_s may be many-to-one). Note that while z^m_s may contain variables referring to the sensor and the target, $z^c_{s,t}$ contains only the position of the target in the state space, which is commonly observed among all sensors. Since the converted measurements $z^c_{s,t}(k)$ depend on the bias estimates $\hat{\beta}_s(k)$ and possibly some additional information z_g, they cannot be considered to be uncorrelated over time. In the state space the targets are assumed to move according to a state-transition model of the form given in equation 2.

No particular fusion architecture is assumed. That is to say that a given fusion node may receive information about a target in the form of a measurement or a track, and received tracks may be correlated through unknown mechanisms with other tracks (such as the track at the local node). This information may be received in an asynchronous manner. Much of the discussion in sections IV and V will be in the context of a distributed or decentralized network, but if the network is centralized then all information is available to reconstruct the necessary data to carry out the same operations.

Finally, it is assumed that some prior information about the bias of each sensor is available (i.e. $\hat{\beta}_s(0)$ is known). In practice this can be achieved with an initial mean of 0 and covariance consistent with some known bounds on the bias values. This is similar to the one-point KF initialization method from [2]. The objective is to update and refine these bias estimates, beginning with the prior, according to information obtained from multiple sensors.

For a concrete example of this type of problem formulation, see section V.

IV. Flexible Sensor Bias Estimation

In this section a computationally efficient algorithm is proposed to estimate the measurement bias vectors in a practical scenario as described in section III above, without adding additional assumptions that narrow the scope of application. The novelty of the proposed algorithm is that is can be applied very generally without significant restrictions in terms of the measurement model, bias model, or fusion architecture. See section V for an example where the operational scenario does not allow the direct application of any computationally

efficient method in the existing literature, but can be handled in real time by the proposed algorithm.

One important note is that in this section the data association (measurement-to-track at the local node, and track-to-track when performing fusion) will be assumed to be known. This is because the association problem is not the focus of this work. In practical applications, data association must be performed online. If some targets are poorly resolved, it may be better to use only well resolved targets in the bias estimation process. This is not ideal, so future work may be warranted to include a more rigorous consideration of data association in this context specifically.

The general strategy in the proposed algorithm can be divided into two main tasks, contained in shaded boxes in figure 5. This figure shows how an update is performed for a single target and a single sensor. The first task, in the shaded box on the left, is to maintain consistent, bias-compensated tracks for each target at each sensor node, using only locally-sourced measurements. These will be referred to as consistent local tracks. The second task, in the shaded box on the right, is to update the bias estimate. This is accomplished by forming a joint estimate between the target state and the bias estimate and updating the combined state using consistent information about the target state from other sensor nodes.

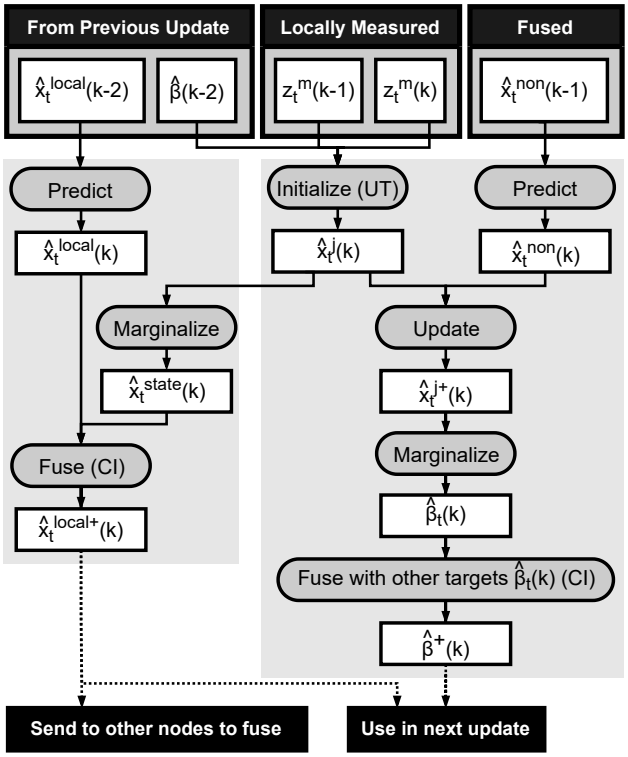

Fig. 5: A diagram of the proposed algorithm showing the data flow for a sensor node update using a single target identified by t. The sensor bias estimate is updated as part of the joint estimate $\hat{x}_t^j(k)$ through cross-correlation with the target state when fusing information from other sensors.

The consistent local tracks are updated using CI to retain their consistency despite correlations introduced by bias compensation, without modeling these correlations explicitly (see section IV-C for more details on bias compensation). The consistent local tracks are sent to other sensor nodes, who fuse them all together (excluding their own track) to form a consistent non-local track that summarizes the information about the target obtained from other sensors. When an estimate for tracking output is required at the local sensor, the consistent local and consistent non-local tracks are fused to provide this. CI is used to fuse in creating both the non-local track and the tracking output because tracks from different sensors may be correlated through process noise as well as the bias compensation process.

The bias estimate update first requires forming a joint estimate of the target state and the sensor bias. This is accomplished for the instantaneous time k using the UT and the two previous local measurements using the UT (the details of this are given in section IV-A below). This is reconstructed at every update to capture the current cross-correlations between the bias parameters and the target state directly at the instant of the update. This is required because these cross-correlations depend not only on the target state (which is tracked) but also on the sensor position (which is not tracked, but is measured). Next, the joint estimate is updated using the predicted consistent non-local track as a pseudo-measurement of sorts. Note that the joint estimate and the consistent non-local track are correlated through the bias compensation process (the bias estimates from each sensor use information from the same measurements). Because these correlations are not explicitly modeled and the non-local track does not include the bias components in the state vector, this requires information fusion under unknown cross-correlation with unequal state dimension. Several options for this are outlined in section II-C. Another option is to ignore the cross-correlations and use the standard KF update. This can result in inconsistent bias estimates, but there is reason to believe that the cross-correlations are small (in some sense) if the (true) bias uncertainty is small relative to the measurement uncertainty. This is because the only source of correlations is the bias compensation process. An additional alternative is proposed in section IV-B, attempting to strike a balance between some existing methods which can be too pessimistic, and the KF which can be too optimistic. Each of these options for the bias estimate update step are evaluated and compared through simulations in section V. Since each target update yields an updated bias estimate, the updated bias estimates from each target are then fused to yield the overall updated bias estimate. Again the bias estimates from each target are correlated, so CI should be used here for the fusion.

The remainder of this section is organized as follows. Section IV-A describes in detail the process of forming the joint estimate between the target state and bias estimate using the UT. Section IV-B proposes a new approach for the update step with unequal state vectors under unknown cross-correlation. Section IV-C provides more information about the bias compensation process, and the correlations introduced by this. The proposed algorithm is given with pseudo-code in section IV-D. An analysis of the computational complexity of the proposed algorithm is then given in section IV-E.

A. Unscented Transform for Joint Estimate Initialization

As described in section II-B, the UT can be used to form a Gaussian estimate of a random variable after it undergoes a nonlinear transformation. In this section, the UT will be used to initialize a joint estimate of the stacked target state and bias estimate. This approach can be used to initialize target states using various motion models, but here an example is shown for initialization based on a constant velocity (CV) model.

First, a stacked random variable consisting of the most recent two sensor measurements, the previous bias estimate, and any additional global information is formed as in equation 12. In this example with a two-point initialization, the bias estimate from time $k-2$ is used with measurements from times $k-1$ and k so that the bias estimate and the measurements are not correlated (i.e. the bias is updated once for every 2 measurements from the local sensor). The the joint distribution parameterized by $\bar{x}_{s,t}^{m,j}, P_{s,t}^{m,j}$ will be the input distribution from which the sigma points are drawn for the UT.

$$\begin{aligned} \bar{x}_{s,t}^{m,j} &= \begin{bmatrix} \bar{z}_{s,t}^{m}(k)^T & \bar{z}_{s,t}^{m}(k-1)^T & \bar{\beta}_s^T(k-2) & \bar{z}_g \end{bmatrix}^T \\ P_{s,t}^{m,j} &= \begin{bmatrix} R_s^m(k) & 0 & 0 & 0 \\ 0 & R_s^m(k-1) & 0 & 0 \\ 0 & 0 & P_\beta(k-2) & 0 \\ 0 & 0 & 0 & P_g \end{bmatrix} \end{aligned} \tag{12}$$

The transformation function f is then constructed as in equation 13 to transform each of the sigma points so that the track state is initialized in a joint distribution with the bias. In this equation $\mathcal{Z}_i^m(k)$, $\mathcal{Z}_i^m(k-1)$, $\mathcal{B}_i(k-2)$, and $\mathcal{Z}_i^g$ represent components of sigma point vector i (in the input space) corresponding to the most recent measurement, the previous measurement, the bias estimate, and the global information, respectively. The measurement conversion function h_s^c is used to map the sensor measurements to the target position in the state space while accounting for the bias uncertainty and global information. The time index is dropped from the bias estimate to simplify the notation. It is worth reiterating here that this is the bias estimate from a time before any of the measurements used in the UT so that it is not correlated with any of them, so for a CV model this can be used every other measurement, for a constant acceleration (CA) model every third measurement, and so on. The time between the measurements is denoted $T(k, k-1)$.

$$\begin{aligned} f(\mathcal{X}_i) &= \begin{bmatrix} \mathcal{P}_i(k) \\ \mathcal{V}_i(k) \\ \mathcal{B}_i \end{bmatrix} \\ \mathcal{P}_i(k) &= h_s^c(\mathcal{Z}_i^m(k), \mathcal{B}_i, \mathcal{Z}_i^g) \\ \mathcal{V}_i(k) &= \frac{\mathcal{P}_i(k) - \mathcal{P}_i(k-1)}{T(k, k-1)} \end{aligned} \tag{13}$$

The resulting transformed points $\mathcal{Y}_i = f(\mathcal{X}_i)$ are in the state space of the target motion model (position and velocity for a CV model) with bias augmented. These points are used to parameterize a Gaussian joint estimate of the target state and bias estimate, including the cross-correlation terms. See algorithm 1 for details about drawing sigma points and the rest of the UT algorithm. For higher-order kinematic models, more measurements can be used.

B. Measurement Update Equivalent to Covariance Intersection in Common State Components

The proposed algorithm involves an update step on line 20 or algorithm 3 that requires information fusion under unknown cross-correlation with unequal state dimensions, as reviewed in section II-C. See section IV-C for more details about the correlations involved. In this section an alternative is proposed using the KF update with a pseudomeasurement derived according to the CI algorithm. This produces the same estimate in common state components as using the standard CI algorithm (with equal state dimensions on just the common components), while also updating the uncommon state components.

The prior estimate $\hat{x}$ will be updated with information $\hat{x}'$ to obtain the posterior estimate $\hat{x}+$, where the state components of $\hat{x}'$ are a subset of the components of $\hat{x}$. The common and uncommon state components of the estimate with the larger state can be broken down into blocks as shown in equation 14, with subscripts c and u, respectively. The same block decomposition applies to $\hat{x}^+$ with the same notation.

$$\begin{aligned} \hat{x} &= \mathcal{N}(\bar{x}, P_x) \\ \bar{x} &= \begin{bmatrix} \bar{x}_c & \bar{x}_u \end{bmatrix}^T \\ P_x &= \begin{bmatrix} P_{cc} & P_{cu} \\ P_{uc} & P_{uu} \end{bmatrix} \end{aligned} \tag{14}$$

First, the CI estimate of the fused common state components is computed using the standard CI algorithm. This will be denoted as $\hat{x}_{ci}$ as shown in equation 15. This is then set equal to the posterior update of the common state components.

$$\begin{aligned} \hat{x}_{ci} &= CI(\hat{x}_c, \hat{x}', trace) \\ &= \mathcal{N}(\bar{x}_{ci}, P_{ci}) \\ &= \hat{x}_{cc}^+ \\ &= \mathcal{N}(\bar{x}_{cc}^+, P_{cc}^+) \end{aligned} \tag{15}$$

Then, starting from the above equivalence and the Kalman Filter measurement update equations, a pseudomeasurement z_{eq} with mean $\bar{z}_{eq}$ covariance R_{eq} is derived. This pseudomeasurement, when used in the standard KF update, results in the same updated (fused) target state estimate as the CI algorithm does (meaning that the updated estimate of the common state is consistent if the input estimates are consistent). The derivation begins in equation 16 below, where the H matrix is just a selector of the common state components taking the form $H = [I 0]$.

$$\begin{aligned} P^+ &= (I - PH^T(HPH^T + R_{eq})^{-1}H)P \\ &= (I - PH^T(P_{cc} + R_{eq})^{-1}H)P \\ &= (I - P \begin{bmatrix} (P_{cc} + R_{eq})^{-1} \\ 0_{mat} \end{bmatrix} H)P \\ &= (I - \begin{bmatrix} P_{cc}(P_{cc} + R_{eq})^{-1} & 0 \\ P_{uc}(P_{cc} + R_{eq})^{-1} & 0 \end{bmatrix})P \end{aligned} \tag{16}$$

Only the top-left block of the matrix from equation 16, pertaining to the common states, is required to solve for R_{eq}. Considering this block only after right-multiplying P yields equation 17.

$$P_{ci} = P_{cc} - P_{cc}(P_{cc} + R_{eq})^{-1}P_{cc} \tag{17}$$

Rearranging equation 17 to solve for R_{eq}, the final expression for the covariance of the pseudomeasurement is given in equation 18. Note that the covariance of the estimate of the common state components must be invertible, as must be $(P_{cc}^{-1} - P_{cc}^{-1}P_{ci}P_{cc}^{-1})$.

$$R_{eq} = (P_{cc}^{-1} - P_{cc}^{-1}P_{ci}P_{cc}^{-1})^{-1} - P_{cc} \tag{18}$$

A similar approach is used to derive $\bar{z}_{eq}$. Again all that is required to solve for $\bar{z}_{eq}$ is the top block of equation 19, corresponding to the common state components. Considering only this block of the vector yields equation 20, which is rearranged to solve for $\bar{z}_{eq}$. The final expression for $\bar{z}_{eq}$ is given in equation 21.

$$\begin{aligned}\bar{x}^+ &= \bar{x} + PH^T(HPH^T + R_{eq})^{-1}(\bar{z}_{eq} - H\bar{x}) \\ &= \bar{x} + \begin{bmatrix} P_{cc}(P_{cc} + R_{eq})^{-1} \\ P_{uc}(P_{cc} + R_{eq})^{-1} \end{bmatrix}(\bar{z}_{eq} - \bar{x}_c)\end{aligned} \tag{19}$$

$$\bar{x}_{ci} = \bar{x}_c + P_{cc}(P_{cc} + R_{eq})^{-1}(\bar{z}_{eq} - \bar{x}_c) \tag{20}$$

$$\bar{z}_{eq} = (I + R_{eq}P_{cc}^{-1})(\bar{x}_{ci} - \bar{x}_c) + \bar{x}_c \tag{21}$$

The measurement z_{eq} is then used in the standard KF update to update the prior estimate $\hat{x}$. The posterior estimate $\hat{x}^+$ resulting from the update is equivalent to CI fusion in the common state components, but the estimate of the uncommon state components is also updated through correlation with the common components. In this sense z_{eq} can be viewed as an approximately-decorrelated measurement under the assumption of unknown correlation between the two estimates of the common state components.

C. Bias Compensation

Once the bias has been estimated, it can be compensated for at the measurement level. As mentioned in section I, many existing works simply assume the bias estimate is good enough and adjust the measured values by the appropriate amount, and in [17] it is shown that considering the variance of the bias estimate can improve tracking performance. Another important reason to consider the variance of the bias estimate is for the purpose of data association, which must be considered prior to bias estimation. If the variance of the bias estimate is not small, the measurements (and resulting tracks) may not actually be consistent with the true position of the targets. This is illustrated in figure 6. The result of this can be missed or incorrect associations at the measurement or track level, in addition to reduced track accuracy. At the onset of the bias estimation process, only the prior information about the bias is known, which in the case of the proposed algorithm is assumed zero-mean with an appropriately large covariance (consistent with reasonable bias values). Therefor it is important for practical purposes to consider how bias compensation can be achieved such that both the measurements and tracks remain consistent. This discussion will assume that the bias estimate being used for compensation is consistent with the true bias.

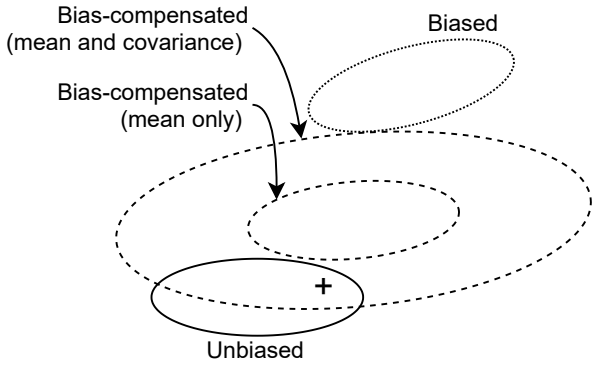

Fig. 6: An example comparing confidence ellipses of measurements that are unbiased, biased, and bias-compensated (with and without bias-inflation). Here the bias estimate covariance is not small relative to the measurement covariance. The measurement compensated using only the mean of the bias estimate is inconsistent with the true target position (shown with a cross), unlike the measurement that is inflated to account for the bias estimate covariance.

At the measurement level, this can be handled as a part of the UT. The bias estimate can be included in the input distribution such that the sigma points generated will include bias components. Then the affects of the bias parameters can be accounted for directly according to the measurement model, prior to applying the measurement conversion. The advantage of this approach is that it is achieved without requiring any additional assumptions about the measurement conversion or bias models, only that there is a known procedure for mapping points in the joint measurement-bias space to the target position in the state space. This also lends itself well to the formation of a joint estimate including the bias (as is done in section IV-A). This converted measurement of the target position in the state space that accounts for bias uncertainty will be referred to as a bias-inflated measurement.

At the track level, it must be considered that the bias-inflated measurements at different times are no longer independent from one another, since they depend on the bias estimate which has been recursively updated from a previous time. Recall that in [17] this is accounted for by modelling the cross-covariance terms explicitly (with some approximations). If this is not feasible for a given measurement and bias model, an alternative approach must be used. One (naive) possibility is to ignore the dependence of the bias-inflated measurements and simply use the KF to track targets. This should be less optimistic than if the variance of the bias estimate is ignored entirely in bias compensation, but is still an approximation due to the assumption of the KF that the measurements are independent from one another over time. If the bias estimate variance is small then this may be an acceptable approximation since the inflated measurements may reasonably be considered "mostly"

independent. If the bias estimate variance is large (such as at the onset of the bias estimation process) it may be important to consider this correlation so that the tracks remain consistent. In this case, the cross-correlation can be treated as unknown and an approach discussed in section II-C can be employed for the update (such as CI). This will converge more slowly than using the KF, but can guarantee consistency and is the approach used in the proposed algorithm to maintain the consistent local tracks (as discussed in section IV-D with reference to figure 5). As the bias estimate becomes more precisely known (its variance shrinks), so does the level of correlation between bias-inflated measurements. At some point the performance trade-off (between convergence and consistency) may favor switching from updating using the assumption of unknown correlation to using that of independence.

D. Proposed Bias Estimation Algorithm

Now that the basic approach for a bias estimate update has been outlined and the preliminaries are taken care of, it is possible to describe the proposed bias estimation algorithm in detail in the form of pseudo-code. The sensor node may either make local observations of targets or receive information about targets (in the form of consistent tracks) from another sensor node. The overall bias estimation process at a single sensor node maintains the bias estimate for this sensor, as well as both a consistent local track and a consistent non-local track (summarizing information received from other sensors) for each target. The consistent non-local tracks are assumed to be from some arbitrary time in the past, noted using the time index $k-\tau$. The local sensor node may transmit its local consistent track to other sensor nodes at any time (subject to practical constraints, for example the availability of a communication channel) to be included in their (from the perspective of the other node) non-local consistent track information.

Notation is introduced here to refer to sets of variables for all targets. This notation is described in equation 22. Note that since these equations are now referring to a single sensor, so the subscript s is dropped from the notation. The exception to this is $\hat{X}_s^{local}(k)$ in algorithm 4 where the sensor performing the update receives information from another sensor, indexed by s to make it clear that this is from a different sensor node.

$$\begin{aligned} Z^m(k) &= \{z_t^m(k) | t \in 1 \dots M(k)\} \\ \hat{X}^{local}(k) &= \{\hat{x}_t^{local}(k) | t \in 1 \dots M(k)\} \\ \hat{X}^{non}(k) &= \{\hat{x}_t^{non}(k) | t \in 1 \dots M(k)\} \end{aligned} \tag{22}$$

The bias estimate and consistent local track updates occur when local measurements are made, and this process is detailed in algorithm 3. The inputs for this algorithm (in addition to those maintained internally, discussed above) are the measurements of each target from this time and the previous measurement time, as well as (optionally) some global information z_g such as terrain data. Note that because 2 measurements are required for the update (with a CV model), this should be done only every other measurement update. When 2 measurements for a target are not yet ready, the arriving measurement can simply be queued up for the next update. If the consistent non-local track is not yet initialized, the bias update must be skipped until this is available. Similarly, targets can be skipped in the same manner if they are not associated (with local measurements or non-local tracks). On line 8 of this algorithm, the UT is used for initialization of the joint estimate between the target state and the bias vector, as described in section IV-A. The function f used in the UT is as defined in equation 13. On line 20 of this algorithm the update operation is fusion under unknown cross-correlation with unequal state dimensions. Several alternatives to achieve this are discussed earlier in section IV-D, and are compared through simulations in section V.

Algorithm 3 ProposedMeasurementUpdate $(Z^m(k), Z^m(k-1), \hat{X}^{local}(k-2), \hat{X}^{non}(k-\tau), \hat{\beta}(k-2), z_g)$

1: $\hat{\beta}(k) = \hat{\beta}(k-2)$
2: **for each** $t \in 1 \dots M(k)$ **do**
    // Skip targets without two measurements
3: **if** $z_t^m(k)$ or $z_t^m(k-1)$ not defined **then**
4: Store $z_t^m(k)$ for next update at time $k+1$
5: **continue**
6: **end if**
    // Initialize joint distribution (section IV-A)
7: $\hat{x}_t^{m,j}(k) = \langle$Use equation 12$\rangle$
8: $\hat{x}_t^j(k) = UT(\hat{x}_t^{m,j}(k), f)$
    // Update local consistent track
9: $\hat{x}_t^{state}(k) = \langle$Marginalize from $\hat{x}_t^j(k)\rangle$
10: **if** $\hat{x}_t^{local}(k-1)$ not defined **then**
11: $\hat{x}_t^{local}(k) = \hat{x}_t^{state}(k)$
12: **else**
13: $\hat{x}_t^{local}(k) = \langle$Predict from $\hat{x}_t^{local}(k-1)\rangle$
14: $\hat{x}_t^{local}(k) = CI(\hat{x}_t^{local}(k), \hat{x}_t^{state}(k), trace)$
15: **end if**
    // Skip targets with no non-local information
16: **if** $\hat{x}_t^{non}(k-\tau)$ not defined **then**
17: **continue**
18: **end if**
    // Update bias estimate
19: $\hat{x}_t^{non}(k) = \langle$Predict from $\hat{x}_t^{non}(k-\tau)\rangle$
20: $\hat{x}_t^{j+}(k) = \langle$Update $\hat{x}_t^j(k)$ with $\hat{x}_t^{non}(k)\rangle$
21: $\hat{\beta}_t(k) = \langle$Marginalize from $\hat{x}_t^{j+}(k)\rangle$
22: $\hat{\beta}(k) = CI(\hat{\beta}(k), \hat{\beta}_t(k), trace)$
23: **end for**
24: **return** $\hat{\beta}(k), \hat{X}^{local}(k)$

The consistent non-local tracks are updated or initialized when tracks are received from another sensor, as described in algorithm 4. Here the inputs are the consistent non-local track (maintained from the previous time step) and the consistent local track from arbitrary other sensor node identified by s. Note that for the latter, this is "local" from the perspective of sensor s, not the sensor being updated. The local measurement information from the sensor being updated is intentionally excluded from the consistent non-local track.

Algorithm 4 ProposedTrackUpdate($\hat{X}^{non}(k-\tau), \hat{X}_s^{local}(k)$)

1: **for each** $t \in 1 \dots M(k)$ **do**
   // Initialize non-local consistent track for new targets
2: **if** $\hat{x}_t^{non}(k-\tau)$ not defined **then**
3: $\hat{x}_t^{non}(k) = \hat{x}_{s,t}^{local}(k)$
   // Update non-local consistent track for existing targets
4: **else**
5: $\hat{x}_t^{non}(k) = \langle$Predict from $\hat{x}_t^{non}(k-\tau)\rangle$
6: $\hat{x}_t^{non}(k) = CI(\hat{x}_t^{non}(k), \hat{x}_{s,t}^{local}(k), trace)$
7: **end if**
8: **end for**
9: **return** $\hat{X}^{non}(k)$

These updates may happen at any time in a practical system, depending which information arrives when, calling the appropriate update function for the incoming information. The proposed algorithm requires data about a shared target from at least one non-local sensor in order to update the bias estimates, so can only be applied in scenarios with at least two sensors, but has no upper limit on the number of sensors. When online tracking output is required, the consistent tracks (local and non-local) maintained during bias estimation are fused for each target using CI to account for all available information and predicted to the current time. Alternatively, the tracking can be considered separately from bias estimation and this information can be bias-compensated and fused in a standard way as it arrives.

E. Computational Complexity

The proposed algorithm is intended for online use, so it is essential to have manageable computational requirements. This section will first consider the basic operations that dominate the computational load, which for this algorithm will be matrix inversion and finding the square root of a matrix. Next, the two major existing algorithms used in the proposed algorithm (CI and the UT) will be discussed in terms of their use of these dominating basic operations. Finally these will be placed in the context of the proposed algorithm to assess its overall computational complexity.

Inversion of an n-by-n matrix is often considered to have computational complexity $\mathcal{O}(n^3)$. Although asymptotically-faster algorithms exist to invert a matrix as fast as $\mathcal{O}(n^{2.373})$ [54], they have large constant factors that can prohibitive for practical applications unless n is very large [55]. For this discussion the computational complexity of matrix inversion will be considered to be cubic, though the real-time practical considerations would still apply if a near-quadratic algorithm were discovered. The complexity of finding the square root of an n-by-n matrix is also $\mathcal{O}(n^3)$ [56].

Matrix inversion is the dominating factor of the computational complexity of the CI algorithm. In general for CI a constrained optimization problem must be solved to minimize the cost function which is (typically) a function requiring inversion (see algorithm 2). The proposed algorithm uses the trace of the fused matrix as a cost function, and solves this optimization problem using the bisection method, requiring a constant number of matrix inversions (assuming the required precision to be constant). Therefor the computational complexity of the CI algorithm to fuse two n-by-n matrices is also $\mathcal{O}(n^3)$. It is worth noting that the computational complexity of standard KF update is also $\mathcal{O}(n^3)$, but with smaller constants since solving an optimization problem is not required.

The UT involves finding the square root of the input covariance matrix, but also involves the transformation of each sigma point according to the function f (see algorithm 1). The UT in general and does not specify what the function f is, so is must be considered as a variable factor. Therefor the computational complexity of the UT for an input matrix of size n-by-n is $\mathcal{O}(n^3 + nf)$.

The proposed non-local consistent track update (algorithm 4) is effectively a standard application of the CI algorithm. The track update is performed whenever information is received from other sensors. This must of course be performed for each target, with the number of targets denoted in this section without a time index as M. The information being fused is the target state, its dimension denoted in this section as n_x. Accordingly, the computational complexity of this operations for all targets for each update is given in equation 23. For most practical applications using kinematic models to track non-cooperative targets, this state vector does not get particularly large. For example, using a constant velocity model in 3 dimensions the dimension of the target state vector will be 6. In any case, this is similar to a standard tracking application (without bias estimation) and computational demands at least similar to this must be expected when fusing information from other sensors.

$$\mathcal{O}(ProposedTrackUpdate) = \mathcal{O}(Mn_x^3) \tag{23}$$

The proposed measurement update (algorithm 3) is where the bias estimation actually happens and involves both CI and the UT, as well as an update step which also must be considered. CI is used to update both on the local consistent track (state vector of dimension n_x), as well as the bias estimate. The dimension of the bias vector will be referred to as n_β. The UT is performed here on the joint distribution created by stacking two measurements and the bias vector (see section IV-A). The dimension of the measurement vector will be denoted n_m, so the dimension of the joint distribution that is input to the UT is $2n_m + n_\beta$. The update step, line 20 of this algorithm, can be achieved in several ways (see section V for a comparison). This is a fusion with unequal state dimension where a joint distribution between the target state and bias vector (dimension $n_x + n_\beta$) is updated with information about the target state (dimension n_x). The computational complexities of each of these update methods (compared through simulation in section V) is dominated again by matrix inversion in the dimension of this joint state. Each of these operations is required for each target used in the update. Putting this all together, the computational complexity of the proposed measurement update is given in equation 24.

$$\begin{aligned}&\mathcal{O}(ProposedMeasurementUpdate) = \\ &\quad \mathcal{O}(M((n_m+n_\beta)^3 + (n_m+n_\beta)f + (n_x+n_\beta)^3))\end{aligned} \tag{24}$$

What is important to notice is that neither of the complexities in equations 23 or 24 depend directly on the number of sensors (denoted in this section without a time index as N). This only comes into play in how frequently algorithm 4 is called. In some sense, this could be considered linear in the number of sensors if the number of sensors are constant over time and all report information regularly, although the proposed algorithm does not assume this to be the case. It is also important that both of these algorithms depend only linearly on the number of targets. The terms involving f resulting from the use of the UT are difficult to compare due to the very general nature of the approach, but if f is fast to evaluate this will not be an issue.

Compare this to the full form of the ASKF operating on sensors with uniform measurement and bias models, which is a standard KF with computational complexity dominated by matrix inversion. The full ASKF uses a state vector of dimension $Mn_t + Nn_\beta$, so it has a much larger asymptotic complexity of $\mathcal{O}((Mn_t + Nn_\beta)^3)$, which cubic both in the number of targets and in the number of sensors. Even more computationally-friendly approaches (such as the pseudo-measurement approach) usually involve a stacked state vector that grows with the number of sensors (or the approach is limited to two sensors), resulting in complexity cubic in the number of sensors but linear in the number of targets. This makes the proposed algorithm much more computationally efficient for scenarios with a large number of sensors.

Since tracking is also performed online, its computational complexity is also important to consider. The proposed algorithm can be used in a standard tracking framework with the bias-compensated converted measurements filtered using a KF and non-local tracks fused using a standard application of the CI algorithm. Alternatively, the local consistent track and non-local consistent tracks, which are already being maintained for bias estimation purposes, can be fused using the standard CI algorithm. In either case, the computation of the updates for tracking output are dominated again by matrix inversion, with complexity cubic in the size of the state vector and linear in the number of targets.

V. Simulations

To evaluate the proposed algorithm and demonstrate its effectiveness and flexibility, simulations were conducted for an application to terrain-aided tracking using video sensors. The proposed algorithm for bias estimation and compensation is compared against tracking with unbiased measurements, and with bias-ignorant measurements (subject to bias that is ignored when tracking). These comparisons serve as high and low baselines, respectively. Several approaches for fusion under unknown cross-correlation with unequal state vectors (applied in the context of the proposed algorithm) are also compared, as well as the different approaches to bias compensation outlined in section IV-C. Two approaches to bias-compensated tracking are compared as well. In the first, a standard KF is used to track with the bias-compensated converted measurements. This first approach ignores some sources of correlation over time between the measurements (more details are given in sections IV-C and V-C). These are based only on local measurements at each sensor node and the bias estimates. The second approach is to fuse the consistent local track and the consistent non-local track, maintained already by each sensor node for the purposes of bias estimation, using CI. This accounts for the correlations between the tracks while utilizing all the information available to the sensor node. For performance evaluation purposes 500 Monte Carlo runs were conducted. Estimation accuracy in both bias estimates and bias-compensated tracks are evaluated using the Root Mean Square Error (RMSE) metric. Track consistency is evaluated using chi-square tests on Average Normalized Estimation Error Squared (ANEES) metric.

A. Scenario

In each Monte Carlo run the true target and sensor trajectories are regenerated randomly according to their process noises and (static) initial conditions, and the true sensor biases (constant over a single run) are drawn from a distribution consistent with the bias prior. Targets move according to a 2-dimensional nearly constant velocity (NCV) model with their elevation fixed to the ground, while sensors move through the air according to a nearly constant acceleration (NCA) model. Both of these models are described in detail in [2]. Based on the trajectories, bias, and measurement noise distributions, noisy sensor measurements are generated randomly for each run and these are used to estimate the sensor biases and target states.

A distributed fusion architecture with 4 sensors observing 4 targets is considered. The sensors take measurements at a rate of 10 frames per second for a period of 300 seconds. Each sensor measures their own position in 3 dimensions as well as the azimuth and elevation angles to each target, subject to Gaussian zero-mean measurement noise. The measurement noise standard deviations were set to 0.5 degrees for the angular components and 3 meters for the positional components. In addition to the zero-mean noise, the angular measurements are also subject to additive constant biases. The bias prior for the angular components is zero-mean with a standard deviation of 2 degrees. The targets are tracked using a 3-dimensional NCV model, as described in [2]. At each time step the sensors communicate their tracks to one another after updating their local tracks and bias estimates. This is to mimic the realistic condition that sensors cannot communicate this information instantaneously, and the effect is that when sensor nodes are performing their local update they have access to information from other nodes only from the previous time step. To illustrate the ability of the proposed algorithm to function in a fully distributed scenario with unreliable communication, simulation results are also presented for a scenario where each sensor only has a 50% chance to receive information from each other sensor at every time step.

B. Digital Elevation Models

Two DEMs were used in the simulated scenario. The first, a high resolution DEM with a resolution of 1m (from [53]), was used to simulate realistic true trajectories for ground targets.

The second, a DEM with a resolution of 30m (from [51]), was used during tracking. DEMs at a 30m resolution are generally available for most locations around the world [51] and they are computationally feasible for use in real time. The level of detail of each of these DEMs can be seen in figure 4. The differences between these two DEMs mimic realistic terrain elevation errors when tracking.

C. Measurement Models for Terrain-Aided Tracking

The measurement model h_s^m of the form in equation 10, defining how the true additive bias affects the measurement, is given below in equation 25.

$$\begin{aligned} z_{s,t}^m(k) &= h_s^m(x_s^m(k), x_t^m(k), \beta_s, v_s(k)) \\ &= \begin{bmatrix} x_{s,x}(k) \\ x_{s,y}(k) \\ x_{s,z}(k) \\ x_{t,az}(k) \\ x_{t,el}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \beta_{az} \\ \beta_{el} \end{bmatrix} + v_s(k) \\ &= \bar{z}_{s,t}^m(k) + v_s(k) \\ &= \mathcal{N}(\bar{z}_{s,t}^m(k), R_s(k)) \end{aligned} \tag{25}$$

The measurement conversion model is applied to transform the measurement $z_{s,t}^m(k)$ and bias estimate to a converted bias-compensated measurement in the target state space. Here the global information z_g is a one-dimensional zero-mean Gaussian noise used to represent the terrain uncertainty. The UT is used here with the joint distribution consisting of the measurement, bias estimate, and global information as input. The nonlinear transformation h_s^c will be applied to the sigma points as in equation 26.

$$\begin{aligned} z_{s,t}^c(k) &= h_s^c(z_s^m(k), \hat{\beta}_s(k), z_g) \\ \mathcal{Z}_i^c(k) &= h_s^c(\mathcal{Z}_i^m(k), \mathcal{B}_i(k), \mathcal{Z}_i^g) \end{aligned} \tag{26}$$

First, bias-corrected sigma points $\mathcal{Z}_i^{m,bc}$ in the measurement space are computed by subtracting the bias components from the corresponding measurement components of each sigma point, as in equation 27.

$$\mathcal{Z}_i^{m,bc}(k) = \begin{bmatrix} \mathcal{Z}_{i,s,x}^m \\ \mathcal{Z}_{i,s,y}^m \\ \mathcal{Z}_{i,s,z}^m \\ \mathcal{Z}_{i,t,az}^m \\ \mathcal{Z}_{i,t,el}^m \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathcal{B}_{i,az} \\ \mathcal{B}_{i,el} \end{bmatrix} \tag{27}$$

Next, each representing a possible line of sight from the sensor to the target, these bias-corrected sigma points are transformed to the state space by calculating its point of intersection with the tracking DEM. This is expressed in equation 28.

$$\mathcal{Z}_i^c(k) = IntersectionPoint(\mathcal{Z}_i^{m,bc}(k), DEM, \mathcal{Z}_i^g) \tag{28}$$

The DEM intersection algorithm described in [48] is used to compute $\mathcal{Z}_i^c$ (see section II-D). The final argument to the function in equation 28 is the sigma point component $\mathcal{Z}_i^g$, corresponding to the DEM elevation error. When each point of intersection between the line of sight $\mathcal{Z}_i^{m,bc}$ and the DEM is calculated, the entire DEM is offset by adding $\mathcal{Z}_i^g$ to its elevation values. This simplified error model for the DEM is used to retain computational efficiency without ignoring the errors entirely. Note that here the converted measurements are not uncorrelated over time due to the uncertainty introduced by the DEM.

D. Algorithm and Scenario Variation Nomenclature

The algorithm proposed in section IV-D leaves several options open in terms of the joint estimate update used to refine the bias estimate. The identifier used for each algorithm variation will correspond to the method used in the update step, where the standard Kalman Filter, the method proposed in section IV-B, Covariance Intersection (with unequal state dimension) from [29], and Covariance Bounds from [36] will be considered. The abbreviations KF, CIKF, CI, and CB will be used, respectively, to identify these variations. These algorithms will also be compared with the standard KF applied to unbiased measurements (as a high performance baseline) and biased measurements without any bias compensation (as a low performance baseline). These baseline approaches will be identified by UB and BI, respectively. In addition to different algorithm variations, two scenarios are considered, one with full communication and one with limited communication, as outlined in section V-A. The identifiers for these scenarios will be FC and LC, respectively. For example, results presented that use the standard Kalman Filter for the joint estimate update under full communication will be labelled KF-FC.

E. Results

Bias estimation is the primary focus of this work and it is of course important that the proposed approach can produce accurate bias estimates. Figure 7 shows a plot of the RMSE of the bias estimates (Euclidean distance error) over time for each of the algorithm and scenario variations. Similar trends are visible in each sensor with the lowest RMSE coming from the CIKF update, followed closely by the KF update, and then the CI update. In each case, the LC scenarios have slightly higher error than the corresponding FC scenarios. This difference between the scenarios is slightly more pronounced in the CI update algorithm variation.

In these simulations the data association has been assumed to be known, but in a real world application the converted bias-compensated measurements and the local consistent track from each sensor will need to be associated with one another during the bias estimation process. If this data association is to be performed reliably, both of these should be consistent estimates. This is verified using chi-square tests on the ANEES metric. The one-sided chi-square test is used here due to its similarity with the standard gating process used in data association. Figures 8 and 9 show the results of these tests for the converted bias-compensated measurements and the local consistent tracks, respectively, for a single sensor and a single target. Nearly all of the time the ANEES is within the shaded 95% acceptance region, indicating that both are consistent estimates.

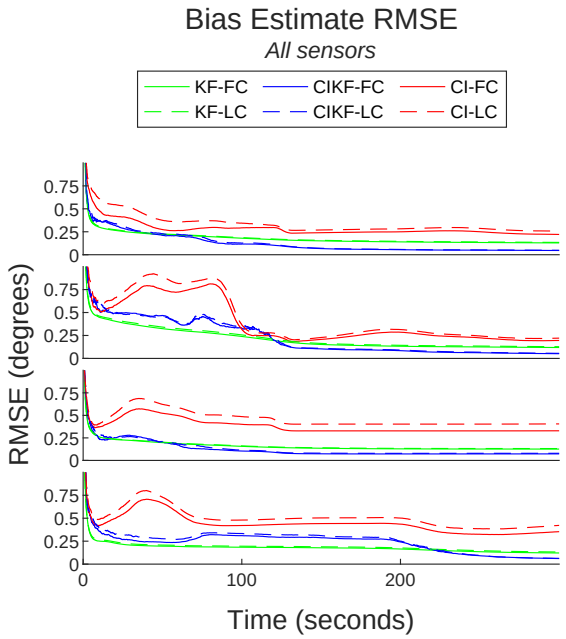

Fig. 7: Root Mean Square Error (RMSE) for the bias estimates $\hat{\beta}_s$ for each sensor s.

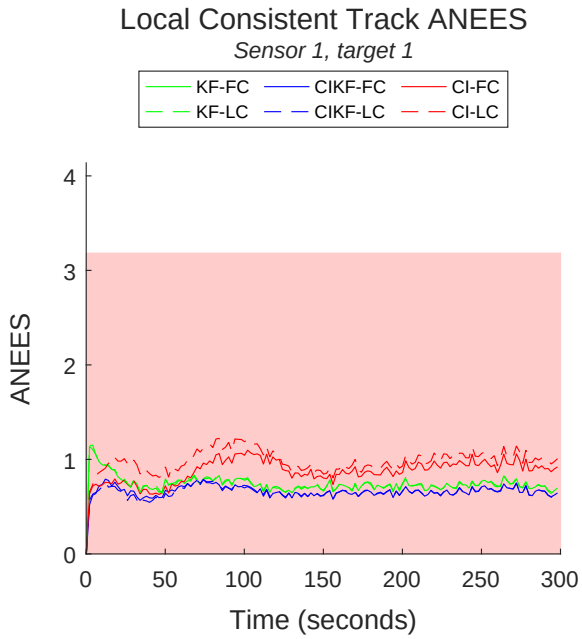

Fig. 9: Average Normalized Estimation Error Squared (ANEES) for the local consistent tracks $\hat{x}_{1,1}^{local}$, with the chi-square test acceptance region shaded.

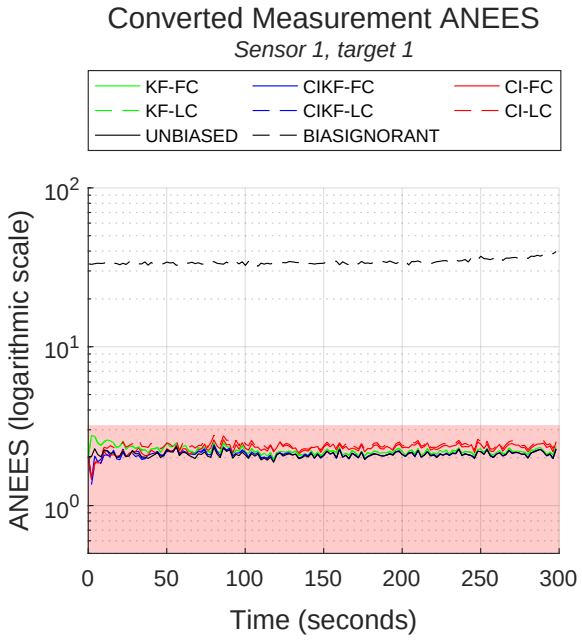

Fig. 8: Average Normalized Estimation Error Squared (ANEES) for the converted bias-compensated measurements $z_{1,1}^c$, with the chi-square test acceptance region shaded.

In addition to bias estimation performance, it is also important to consider the performance of bias-compensated tracking in terms of both accuracy and consistency. Perhaps the simplest approach is to use the standard KF framework operating on the converted bias-compensated measurements originating from local sensor node. Figure 10 shows a plot of the RMSE of these tracks for a single sensor and target, including the unbiased and bias-ignorant trackers as a baseline. A logarithmic scale here is used due to the substantially higher RMSE of the bias-ignorant tracks. The lowest RMSE among the bias-compensated tracks comes from the algorithm variation using the CIKF update, followed closely by the KF variation, and then the CI variation. Similar to what was seen in the RMSE of the bias estimates, the LC scenarios have slightly higher error than the corresponding FC scenarios, with a slightly more pronounced difference between the two in the CI variation. Bias-compensated tracks using any of the proposed algorithm variations are significantly more accurate than the bias-ignorant tracks, and those using the KF and CIKF variations converge to a level of track accuracy comparable to the unbiased tracks.

Notice that track that was bias-compensated according to the bias estimates from CIKF update algorithm actually outperforms the unbiased tracker by the end of the simulation period. The unbiased tracker is included as a high performance baseline, so this at first glance seems unexpected and must be explained. As previously noted, this application violates the assumptions of the KF that the measurement noise is zero-mean and independent over time (see sections IV-C and V-C). This can lead the filter to be inconsistent, optimistic in the accuracy of its estimates. This optimism is seen clearly in figure 11, showing the chi-square test results for these tracks. The ANEES metric quickly exceeds the 95% acceptance region for all trackers, even the unbiased tracker. The bias-compensated measurements have inflated covariance relative to the unbiased measurements, partially mitigating the optimism, but this is not a proper solution. Since the KF and CIKF algorithm variations

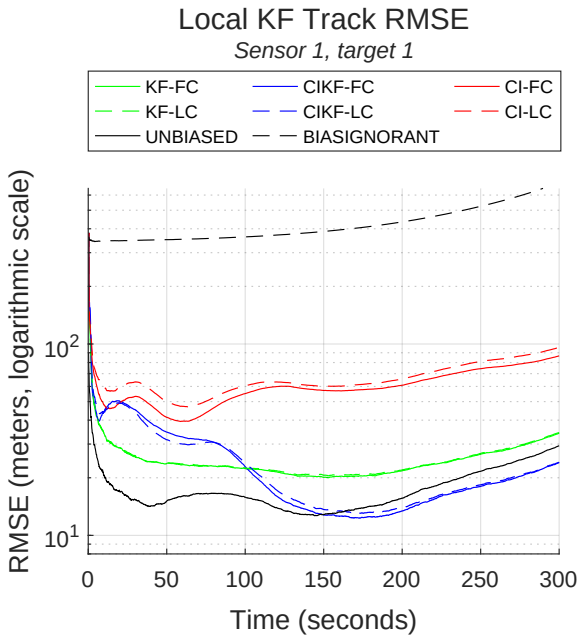

Fig. 10: Root Mean Square Error (RMSE) plotted on a logarithmic scale for the tracks from standard KF operating on bias-compensated measurements $z^c_{1,1}$.

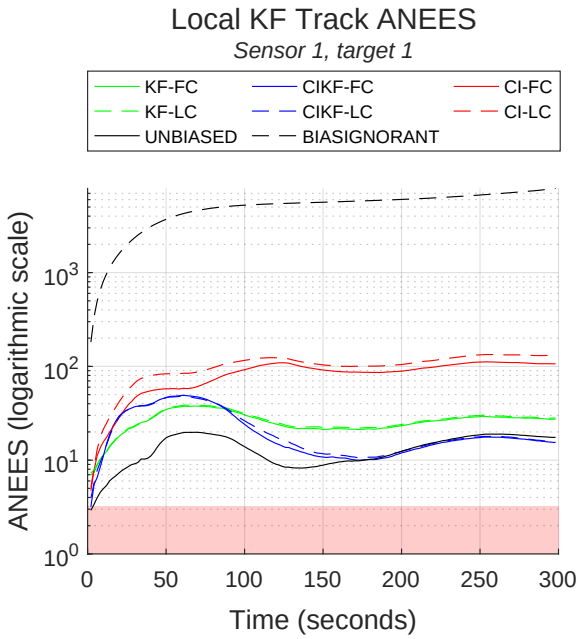

Fig. 11: Average Normalized Estimation Error Squared (ANEES) for the tracks from standard KF operating on bias-compensated measurements $z^c_{1,1}$, with the chi-square test acceptance region shaded.

have ANEES values that converge to levels comparable to those of the unbiased tracker, the majority of the dependence over time seems to be due to the terrain errors. Modelling the terrain errors more accurately as a time-varying (location-dependent) bias specific to individual targets may be a more appropriate solution, and this may be a fruitful avenue for future work.

The standard KF cannot be expected to produce consistent tracks with measurements that are not independent over time unless this dependence is explicitly accounted for. In general this may not always be possible, and indeed this problem seems likely to occur in practically all real world non-cooperative tracking applications that rely on an imperfect terrain model. As seen in the local consistent tracks, using CI to update the tracks instead of the KF allows for the tracks to be maintained in a consistent way, as long as the measurements or tracks to be fused are each consistent themselves. This consistency is not obtained without a price however, the estimation accuracy may suffer. This can be seen by comparing the RMSE of the local consistent tracks, shown in figure 12, with that of the standard KF tracks, shown in figure 10. Both of these tracks use the same measurements and bias estimates, and the tracking methods show a trade-off between consistency and accuracy.

The local consistent tracks mentioned in the above comparison are used during bias estimation and are not intended to be used directly as tracker output, so their consistency is prioritized over their accuracy. If the consistent local tracks $\hat{x}^{local}_{s,t}$ and the consistent non-local tracks $\hat{x}^{nonlocal}_{s,t}$ (already maintained during bias estimation) are fused using CI then a more accurate track can be produced that remains consistent. Examining the RMSE of these fused tracks in figure 13 reveals that they are similarly accurate to the tracks from the standard KF tracker, although they do not decrease as smoothly. The chi-square test on the ANEES of these tracks confirms that they are also consistent, as seen in figure 14.

The data so far has been presented for only sensor 1 and target 1 to show a high level of detail, but similar results are seen across all sensors and targets. To demonstrate this, the RMSE of the standard KF tracks, local consistent tracks, and fused tracks for all sensors and targets are included in figures 15, 16, and 17, respectively. Unlike the more detailed plots, a linear scale is used here to provide a more intuitive understanding of the differences between the various tracks. ANEES plots for all sensors and all targets are omitted for brevity.

VI. Discussion

The simulation results presented in section V-E demonstrate that the proposed algorithm can effectively estimate sensor biases, particularly the algorithm variations using KF and CIKF for the bias estimate update. The bias estimates from these variations can be used to reduce the bias-compensated track RMSE to levels comparable to those of a tracker operating on unbiased measurements while using a standard KF tracking framework. Although the tracks obtained from the standard KF are not consistent due to the converted bias-compensated measurements not being zero-mean or independent over time, this effect is also observed in the unbiased tracker and is primarily a result of imperfectly-known terrain. If the proposed algorithm is used in other applications that do not involve

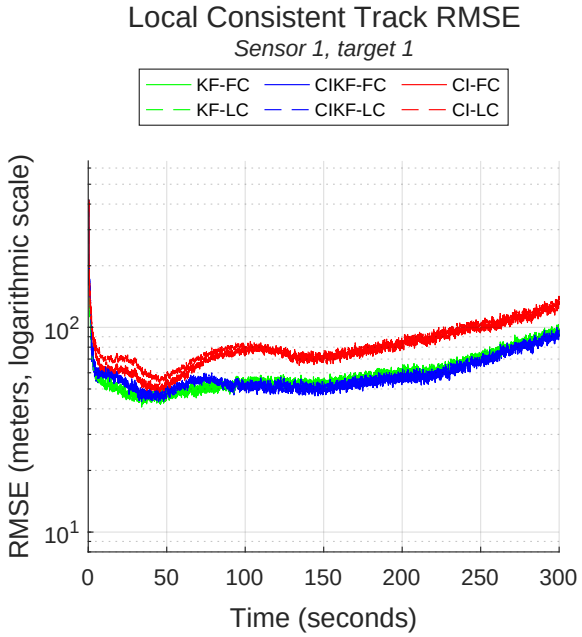

Fig. 12: Root Mean Square Error (RMSE) plotted on a logarithmic scale for the local consistent tracks $\hat{x}_{1,1}^{local}$.

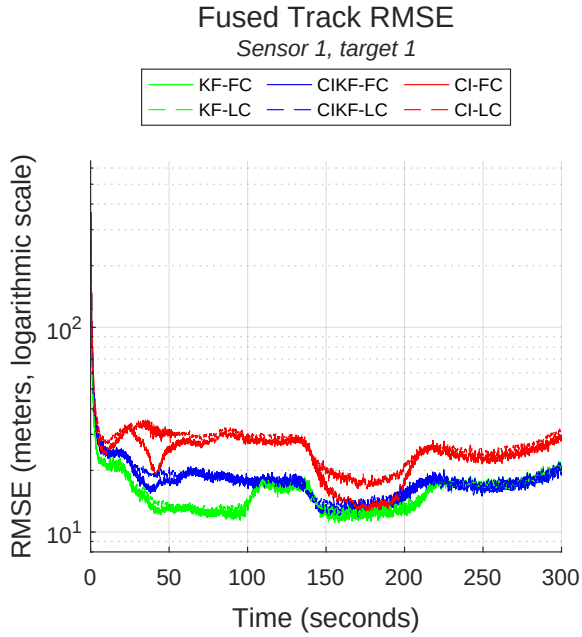

Fig. 13: Root Mean Square Error (RMSE) plotted on a logarithmic scale for the tracks obtained by fusing the local consistent tracks $\hat{x}_{1,1}^{local}$ and non-local consistent tracks $\hat{x}_{1,1}^{nonlocal}$ using CI.

terrain models (for example, in stereo video or radar applications) this may be less of an issue. For the application at hand, this lack of consistency is resolved while retaining a similar level of track accuracy by fusing the consistent local and non-local tracks that are already maintained for the purposes of bias estimation. The simulations are done in the context of a distributed fusion architecture with unreliable asynchronous communication, using angle-only measurements taken from a sensor platform with location uncertainty, and a nonlinear terrain-aided measurement conversion model to track targets through varied terrain in real time. This is a challenging operational scenario that cannot be handled by existing computationally-efficient bias estimation approaches without significant modification, and was chosen to showcase the flexibility afforded by the proposed approach. Although not directly demonstrated through simulation, the proposed algorithm can also handle heterogeneous sensor models, networks with a large number of sensors, time-varying biases, and other fusion architectures. This flexibility allows bias estimation to be applied with minimal modification in combination with various tracking techniques and under operational conditions where it was not previously possible. This bridges an important gap in practical tracking applications.

It is important to mention that the proposed algorithm is not without limitations. The most pressing limitation is that it is not theoretically optimal. Various correlations are assumed to be unknown during the bias estimation process, or are neglected in some algorithm variations. Some notion of optimality could be chased by explicitly modelling these correlations, but this would require non-negligible effort that would be necessarily application-specific. Such an approach would come at the expense of flexibility and probably additional computational demands. It is also worth noting that even if this expense is paid, optimality is likely to only be achieved under stringent assumptions that are unlikely to hold up strictly in the real world. It is for this reason that the proposed approach eschews an optimal solution in favor of one that is flexible.

Another related limitation of the proposed algorithm is that none of the options for performing the joint estimate update to update the bias estimate are necessarily ideal, even under the assumption of unknown correlations. As seen in the simulation results, they can yield varied results. These approaches and their limitations have been discussed already in section II-C and IV-B. This is already a problem that is an area of active research, but future developments here may lead to direct improvements in the performance of the proposed algorithm if they can be effectively applied to it.

In terms of bias-compensated tracking, another limitation is that the two approaches presented in simulations here are two extremes. The standard KF framework assumes measurement noise to be entirely independent over time, while updating using CI exclusively assumes the correlation over time to be entirely unknown. In reality, the measurement noise originating from the sensors directly is independent over time, but the uncertainty introduced by the terrain conversion and bias compensation processes is not. Hybrid approaches that take advantage of this (such as Split Covariance Intersection (SCI), originally proposed in [57]) have been applied to similar situations. If the independent and dependent components of the measurements can be effectively separated, this may provide a path for improvement. This could also possibly be applied

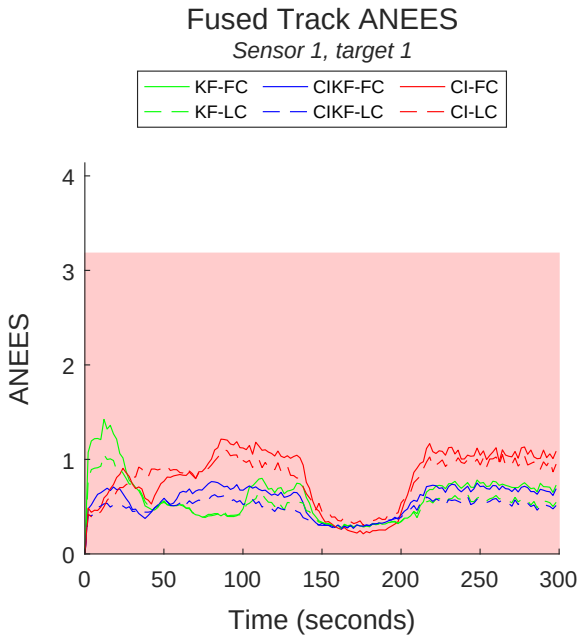

Fig. 14: Average Normalized Estimation Error Squared (ANEES) for the tracks obtained by fusing the local consistent tracks $\hat{x}_{1,1}^{local}$ and non-local consistent tracks $\hat{x}_{1,1}^{nonlocal}$ using CI, with the chi-square test acceptance region shaded.

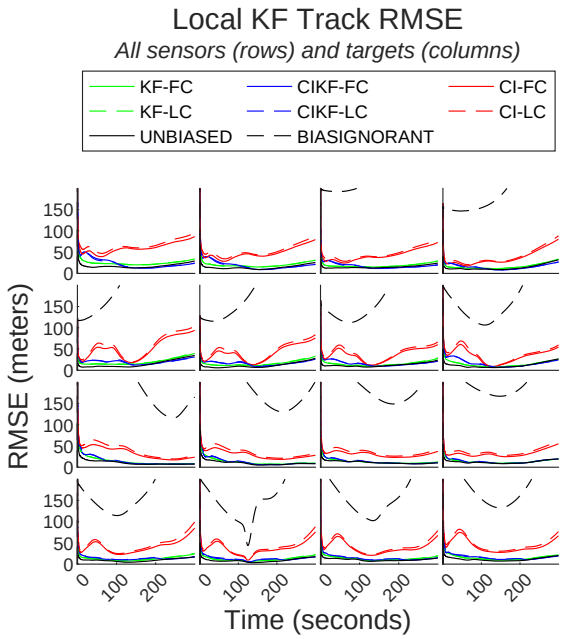

Fig. 15: Root Mean Square Error (RMSE) for the tracks from standard KF operating on bias-compensated measurements $z_{s,t}^{c}$.

to the maintenance of the consistent local and non-local tracks used in bias estimation, which may improve performance not only of tracking but also of bias estimation.

VII. Conclusion

In multisensor-multitarget tracking applications sensor biases that are not compensated for can result in significantly degraded performance. This can take the form of decreased track accuracy, decreased track probability of detection, formation of ghost tracks, or missed associations. In light of this, bias estimation and compensation are essential for practical applications where it is often difficult to ensure that there are no sensor biases present. Despite a significant body of literature on the subject, existing computationally-efficient approaches remain tied to specific assumptions about the operational conditions such as particular fusion architectures, communication network properties, sensor types, measurement models, terrain configurations, constant or time-varying biases, or number of sensors.

The algorithm proposed here is one that is generally applicable to a wide range of scenarios, and here lies its novelty. It is formulated by restructuring the estimation problem so that virtually any shared state information about a target can be used to update the bias estimates. By compiling state information about a single target from other sensor nodes using the CI algorithm a consistent summary of this information is maintained that can be used to update the local sensor's bias estimate through correlation with the common target state components. The UT is used to form a joint estimate of the target state and bias estimate, and the summarized state

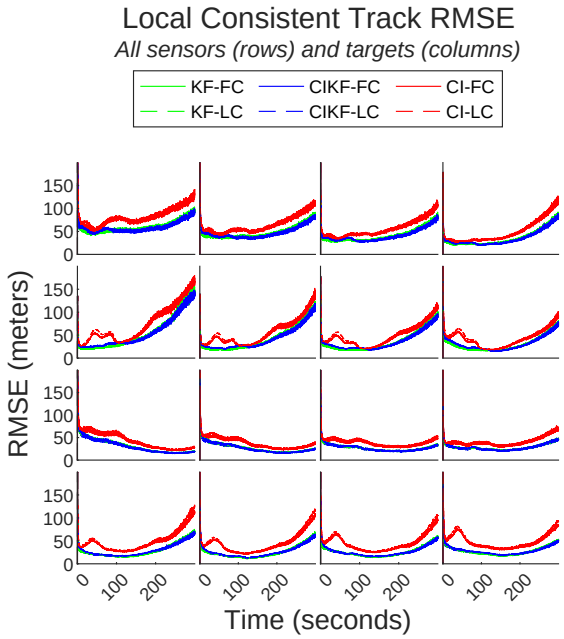

Fig. 16: Root Mean Square Error (RMSE) for the local consistent tracks $\hat{x}_{s,t}^{local}$.

information from other sensors is then used to update the joint state. Although the problem is restructured to minimize the degree of correlation between the joint estimate and the track summary, it is not completely eliminated. Therefor the update step requires fusion under unknown correlation with unequal state dimensions, which remains an area of active research

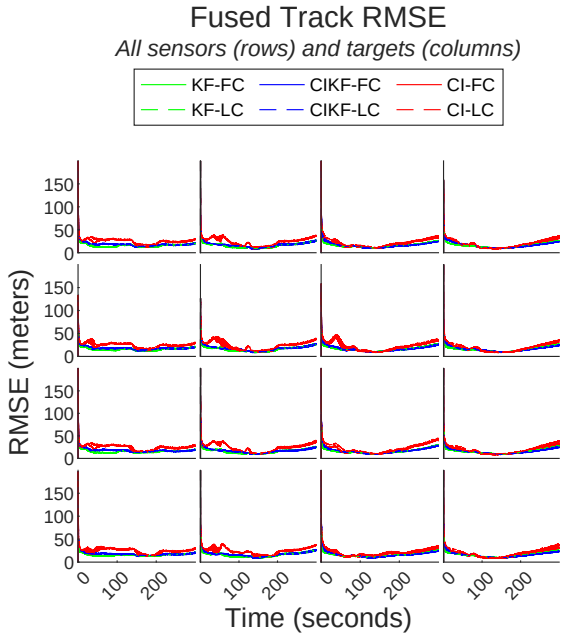

Fig. 17: Root Mean Square Error (RMSE) for the tracks obtained by fusing the local consistent tracks $\hat{x}_{s,t}^{local}$ and non-local consistent tracks $\hat{x}_{s,t}^{nonlocal}$ using CI.

interest. Several options for accomplishing this are explored through simulations, and this may be an area where further improvements are possible. Despite the fact that each of these options are imperfect, the proposed algorithm is shown in a challenging simulated scenario to estimate the sensor biases effectively. The corresponding bias-compensated measurements can be used in a standard KF tracking framework yielding tracks that are comparable to those of a tracker operating on unbiased measurements in terms of accuracy and consistency. In the demonstrated scenario, even the unbiased tracker is inconsistent due to the dependence of the converted measurements over time. Consistent and comparably accurate tracks are obtained by fusing the consistent local and non-local tracks already being maintained for the purpose of bias estimation. This flexible approach allows for sensor biases to be effectively estimated and compensated for in real time under practical operational conditions and with various tracking approaches where it was not previously possible to do so.

REFERENCES

[1] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960.

[2] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[3] B. A. van Doorn and H. A. Blom, "Systematic error estimation in multisensor fusion systems," in *Signal and Data Processing of Small Targets 1993*, vol. 1954. International Society for Optics and Photonics, 1993, pp. 450–461.

[4] B. Friedland, "Treatment of bias in recursive filtering," *IEEE Transactions on Automatic Control*, vol. 14, no. 4, pp. 359–367, 1969.

[5] J. Yi, X. Wan, and D. Li, "Exactly decoupled kalman filtering for multitarget state estimation with sensor bias," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 3, pp. 2256–2271, 2019.

[6] K. Kastella, B. Yeary, T. Zadra, R. Brouillard, and E. Frangione, "Bias modeling and estimation for gmti applications," in *Proceedings of the third international conference on information fusion*, vol. 1. IEEE, 2000, pp. TUC1–7.

[7] A. Alouani, T. Rice, and W. Blair, "A two-stage filter for state estimation in the presence of dynamical stochastic bias," in *1992 American Control Conference*. IEEE, 1992, pp. 1784–1788.

[8] E. Taghavi, R. Tharmarasa, T. Kirubarajan, Y. Bar-Shalom, and M. Mcdonald, "A practical bias estimation algorithm for multisensor-multitarget tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 1, pp. 2–19, 2016.

[9] E. Taghavi, D. Song, R. Tharmarasa, T. Kirubarajan, M. McDonald, B. Balaji, and D. Brown, "Geo-registration and geo-location using two airborne video sensors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 4, pp. 2910–2921, 2020.

[10] D. Song, R. Tharmarasa, W. Wang, B. Rao, D. Brown, and T. Kirubarajan, "Efficient bias estimation in airborne video geo-registration for ground target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, 2021.

[11] H. Zhu, H. Leung, and K.-V. Yuen, "A joint data association, registration, and fusion approach for distributed tracking," *Information Sciences*, vol. 324, pp. 186–196, 2015.

[12] X. Yong, Y. Fang, Y. Wu, and P. Yang, "An asynchronous sensor bias estimation algorithm utilizing targets' positions only," *Information Fusion*, vol. 27, pp. 54–63, 2016.

[13] P. Li, R. Goodall, and V. Kadirkamanathan, "Parameter estimation of railway vehicle dynamic model using rao-blackwellised particle filter," in *2003 European Control Conference (ECC)*. IEEE, 2003, pp. 2384–2389.

[14] N. Gordon, B. Ristic, and S. Arulampalam, "Beyond the kalman filter: Particle filters for tracking applications," *Artech House, London*, vol. 830, no. 5, pp. 1–4, 2004.

[15] J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 197–223.

[16] K. Gellert and E. Schlögl, "Parameter learning and change detection using a particle filter with accelerated adaptation," *FIRN Research Paper, Forthcoming*, 2018.

[17] M. Ying, H. Jiang-yuan, and Y. Zhi-hui, "3d asynchronous multisensor target tracking performance with bias compensation," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 5. IEEE, 2010, pp. V5–401.

[18] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068. International Society for Optics and Photonics, 1997, pp. 182–193.

[19] F. Castanedo, "A review of data fusion techniques," *The scientific world journal*, vol. 2013, 2013.

[20] C.-H. Kim, K. Choi, C.-K. Ryoo, K.-D. Park, J.-B. Kim, K.-S. Kim, and J.-L. Jo, "Terrain data aided passive ground target tracking," in *2009 ICCAS-SICE*. IEEE, 2009, pp. 3646–3650.

[21] X. Luo and I. Moroz, "Ensemble kalman filter with the unscented transform," *Physica D: Nonlinear Phenomena*, vol. 238, no. 5, pp. 549–562, 2009.

[22] H. Zou, J. Tao, S. K. Elsayed, E. E. Elattar, A. Almalaq, and M. A. Mohamed, "Stochastic multi-carrier energy management in the smart islands using reinforcement learning and unscented transform," *International Journal of Electrical Power & Energy Systems*, vol. 130, p. 106988, 2021.

[23] S. J. Julier and J. K. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations," in *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, vol. 4. IEEE, 1997, pp. 2369–2373.

[24] H. Li, F. Nashashibi, B. Lefaudeux, and E. Pollard, "Track-to-track fusion using split covariance intersection filter-information matrix filter (scif-imf) for vehicle surrounding environment perception," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 1430–1435.

[25] G. Li, G. Battistelli, W. Yi, and L. Kong, "Distributed multi-sensor multi-view fusion based on generalized covariance intersection," *Signal Processing*, vol. 166, p. 107246, 2020.

[26] B. Noack, J. Sijs, and U. D. Hanebeck, "Inverse covariance intersection: New insights and properties," in *2017 20th International Conference on Information Fusion (Fusion)*. IEEE, 2017, pp. 1–8.

[27] J. K. Uhlmann, "Covariance consistency methods for fault-tolerant distributed data fusion," *Information Fusion*, vol. 4, no. 3, pp. 201–215, 2003.

[28] W. Niehsen, "Information fusion based on fast covariance intersection filtering," in *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002.(IEEE Cat. No. 02EX5997)*, vol. 2. IEEE, 2002, pp. 901–904.

[29] J. K. Uhlmann, "Dynamic map building and localization: New theoretical foundations," Ph.D. dissertation, University of Oxford Oxford, 1995.

[30] J. R. A. Klemets and M. Hovd, "Hierarchical decentralized state estimation with unknown correlation for multiple and partially overlapping state vectors," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2018, pp. 508–514.

[31] C. Allig and G. Wanielik, "Unequal dimension track-to-track fusion approaches using covariance intersection," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[32] V. K. Saini, A. A. Paranjape, and A. Maity, "Decentralized information filter with non-common states, and application to sensor bias estimation," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 0711.

[33] V. Saini, A. A. Paranjape, and A. Maity, "Decentralized information filter with noncommon states," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 9, pp. 2042–2054, 2019.

[34] U. D. Hanebeck and K. Briechle, "New results for stochastic prediction and filtering with unknown correlations," in *Conference Documentation International Conference on Multisensor Fusion and Integration for Intelligent Systems. MFI 2001 (Cat. No. 01TH8590)*. IEEE, 2001, pp. 147–152.

[35] U. D. Hanebeck, K. Briechle, and J. Horn, "A tight bound for the joint covariance of two random vectors with unknown but constrained cross-correlation," in *Conference Documentation International Conference on Multisensor Fusion and Integration for Intelligent Systems. MFI 2001 (Cat. No. 01TH8590)*. IEEE, 2001, pp. 85–90.

[36] B. Noack, J. Sijs, and U. D. Hanebeck, "Fusion strategies for unequal state vectors in distributed kalman filtering," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3262–3267, 2014.

[37] Z. Li, K. You, and S. Song, "Cooperative field prediction and smoothing via covariance intersection," *IEEE Transactions on Signal Processing*, vol. 69, pp. 797–808, 2021.

[38] J. Steinbring, B. Noack, M. Reinhardt, and U. D. Hanebeck, "Optimal sample-based fusion for distributed state estimation," in *2016 19th International Conference on Information Fusion (FUSION)*. IEEE, 2016, pp. 1600–1607.

[39] S. Radtke, B. Noack, U. D. Hanebeck, and O. Straka, "Reconstruction of cross-correlations with constant number of deterministic samples," in *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, 2018, pp. 1638–1645.

[40] S. Radtke, B. Noack, and U. D. Hanebeck, "Distributed estimation with partially overlapping states based on deterministic sample-based fusion," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 1822–1829.

[41] ——, "Reconstruction of cross-correlations between heterogeneous trackers using deterministic samples," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 3236–3241, 2020.

[42] C. Allig and G. Wanielik, "Heterogeneous track-to-track fusion using equivalent measurement and unscented transform," in *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, 2018, pp. 1948–1954.

[43] K. Yang, Y. Bar-Shalom, and K.-C. Chang, "Information matrix fusion for nonlinear, asynchronous and heterogeneous systems," in *2019 22th International Conference on Information Fusion (FUSION)*. IEEE, 2019, pp. 1–6.

[44] O. Dagan and N. R. Ahmed, "Heterogeneous decentralized fusion using conditionally factorized channel filters," in *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2020, pp. 46–53.

[45] ——, "Exact and approximate heterogeneous bayesian decentralized data fusion," *arXiv preprint arXiv:2101.11116*, 2021.

[46] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, and C. N. Taylor, "Vision-based target geo-location using a fixed-wing miniature air vehicle," *Journal of Intelligent and Robotic Systems*, vol. 47, no. 4, pp. 361–382, 2006.

[47] N. Collins and C. Baird, "Terrain aided passive estimation," in *Proceedings of the IEEE National Aerospace and Electronics Conference*. IEEE, 1989, pp. 909–916.

[48] D. Schonborn, T. Kirubarajan, R. Tharmarasa, M. McDonald, and M. Bradford, "Terrain-aided tracking using a gaussian mixture measurement model," *Submitted to IEEE Transactions on Aerospace and Electronic Systems (September 2021)*, 2021.

[49] F. K. Musgrave, "Grid tracing: Fast ray tracing for height fields," Yale University Department of Computer Science Research, Tech. Rep. YALEU/DCS/RR-639, 1988.

[50] S. D. Ramsey, K. Potter, and C. Hansen, "Ray bilinear patch intersections," *Journal of Graphics Tools*, vol. 9, no. 3, pp. 41–47, 2004.

[51] NASA JPL, "Nasa shuttle radar topography mission global 1 arc second [data set]," *NASA EOSDIS Land Processes DAAC*, 2013. [Online]. Available: https://doi.org/10.5067/MEaSUREs/SRTM/SRTMGL1.003

[52] NASA, "Shuttle radar topography mission (srtm) version 2. version 2," *Archived by National Aeronautics and Space Administration, U.S. Government, NASA*, 2000. [Online]. Available: https://http://www2.jpl.nasa.gov/srtm/

[53] J. E. Robinson, "High-resolution digital elevation dataset for crater lake national park and vicinity, oregon, based on lidar survey of august–september 2010 and bathymetric survey of july 2000," *US Geological Survey Data Series*, vol. 716, 2012.

[54] V. V. Williams, "Multiplying matrices faster than coppersmith-winograd," in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, 2012, pp. 887–898.

[55] A. V. Smirnov, "The bilinear complexity and practical algorithms for matrix multiplication," *Computational Mathematics and Mathematical Physics*, vol. 53, no. 12, pp. 1781–1795, 2013.

[56] N. J. Higham, "Computing real square roots of a real matrix," *Linear Algebra and its applications*, vol. 88, pp. 405–430, 1987.

[57] S. J. Julier and J. K. Uhlmann, "Using covariance intersection for slam," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 3–20, 2007.