Practical Solutions to Tracking Problems

Practical Solutions to Tracking Problems

By David Schonborn,

A Thesis Submitted to the School of Graduate Studies in the Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy

McMaster University
Doctor of Philosophy (2022)
Hamilton, Ontario (Department of Electrical and Computer Engineering)

TITLE: Practical Solutions to Tracking Problems
AUTHOR: David Schonborn (McMaster University)
SUPERVISOR: Dr. T. Kirubarajan
NUMBER OF PAGES: xi, 108

Abstract

Tracking systems are already encountered in everyday life in numerous applications, but many algorithms from the existing literature rely on assumptions that do not always hold in realistic scenarios, or can only be applied in niche circumstances. Therefor this thesis is motivated to develop new approaches that relax assumptions and restrictions, improve tracking performance, and are applicable in a broad range of scenarios. In the area of terrain-aided tracking this an algorithm is proposed to track targets using a Gaussian mixture measurement distribution to better represent multimodal distributions that can arise due to terrain conditions. This allowed effective use in a wider range of terrain conditions than existing approaches, which assume a unimodal Gaussian measurement distribution. Next, the problem of estimating and compensating for sensor biases is considered in the context of terrain-aided tracking. Existing approaches to bias estimation cannot be easily reconciled with the nonlinear converted measurement model applied in terrain-aided tracking. To address this, a novel efficient bias estimation algorithm is proposed that can be applied to a wide range of measurement models and operational scenarios, allowing for effective bias estimation and measurement compensation to be performed in situations that cannot be handled by existing algorithms. Finally, to address scenarios where converted measurement tracking is not possible or desired, the problem of sensor motion compensation when tracking in pixel coordinates is considered. Existing approaches compensate for sensor motion by transforming state estimates between frames, but are only able to achieve partial transformation of the state estimate and its covariance matrix. This thesis proposes a novel algorithm used to transform the full state estimate and its covariance matrix, improving tracking performance when tracking with a low frame rate and when tracking targets moving with a nearly coordinated turn motion model. Each of the proposed algorithms are evaluated in several simulated scenarios and compared against existing approaches and baselines to demonstrate their efficacy.

Acknowledgements

Here I will take the chance to give thanks to some of the many people who have helped me so much along the way and made this an experience to cherish. Nobody can accomplish much alone and even if I could, I'd prefer to do it with all of you anyway.

Thank you Dr. Kirubarajan, I could not have done this without you. Your kind support and advice have helped me immeasurably to develop academically and as a person. I will not forget that you always afforded me the grace to take time for myself when I needed to. It has been a pleasure and an honour getting to know you both professionally and personally. I hope that I can help others as much as you have helped me.

I am lucky to have access to not just one wonderful mentor, but two. Thank you Dr. Tharmarasa for always being there to provide guidance, discuss problems, teach me, and just talk. It is always a pleasure to see your friendly face. It seems like every time I do, I learn something new.

I would also like to thank my supervisory committee members Dr. Jeremic and Dr. Chen. Your helpful advice and feedback have not only guided my work but also helped me to present it more clearly and effectively, something one can always improve at I think.

Of course I would like to thank my family, Julia, Al, Joanna, John, Betty-Ruth, and my girlfriend, Jen. Having your love and support through life means the world. You already know that, but thank you for everything.

Last but not least I would like to thank my second family, my friends. Thank you Adrian, Chris, Kane, Jamie, Richard, Sean, Warren, Eric, Moody, Sarah, Daley, Adrienne, James, Tammy, Tim, Cory, Pengfei, Saeid, Mehrnoosh, Khaqan, Keqi, Victoria, and Laura. I'm fortunate to have so many great friends who support me in achieving my goals and make my life better.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, David SCHONBORN, declare that the work presented in this thesis, “Practical Solutions to Tracking Problems”, is my own. All sources consulted are acknowledged and co-authors are identified and credited clearly.

Chapter 1

Introduction

1.1 A Review of Some Challenges in Modern Tracking

Tracking systems are an important component of many promising technologies such as autonomous vehicles, smart cities, and surveillance systems, among others. Although tracking systems are already becoming an integral part of everyday life, deployed in numerous practical applications, many challenges remain. In many cases, algorithms from the existing literature are found to be effective, but they rely on assumptions that do not necessarily hold in the real world, or they cannot be applied in a wide range of scenarios. When these approaches can be applied and their assumptions are a good approximation of the real world then they are effective, but this is not always the case. This thesis, which is organized as a sandwich thesis with three different publications that address three inter-related problems in the same tracking scenario of air-to-ground tracking, is therefor motivated to work towards addressing these limitations, investigating the possibility of relaxing assumptions or restrictions of existing approaches that limit their use or effectiveness is a variety of operational conditions. The applications considered are terrain-aided tracking in varied terrain, sensor bias estimation and measurement compensation, and sensor motion compensation for tracking in pixel coordinates. These applications, the limitations of existing approaches, and novel approaches proposed in this thesis are introduced briefly in this section.

One assumption that is frequently used (and sometimes valid) in air-to-ground tracking scenarios is that the terrain within the area of interest is flat, sometimes called the flat earth assumption. This assumption is commonly used to convert bearings-only measurements to measurements in Cartesian coordinates in the world frame of reference, both assumed to have Gaussian distributions. When the flat earth assumption does not hold, the accuracy of the conversion degrades. This has been partially addressed by the use of Digital Elevation Models (DEMs) to consider terrain which is not flat during measurement conversion, but the converted distribution is still assumed to be Gaussian by current state of the art approaches. In practice, some terrain is peaky and the grazing angle to targets may be low. This leads to a multimodal distribution of the target's position measurement due to uncertainty in the sensor's position and orientation, as well as errors in the DEM. Chapter 2 considers such a scenario, and proposes to instead model the converted measurement in Cartesian coordinates using a Gaussian mixture distribution with a dynamically-estimated number of mixture components. When the flat earth assumption holds this approach automatically falls back to the standard Gaussian assumption (a Gaussian mixture with a single component), so it is not tied to specific assumptions about the terrain and flight path.

Some algorithms see their performance degrade when their assumptions are less valid, while others are strictly applicable under particular operational conditions and *cannot* be applied otherwise. This is particularly prevalent in sensor bias estimation (or sensor calibration). Standard tools such as the Kalman Filter can effectively remove zero-mean noise from measurements through tracking, but cannot handle errors introduced by sensor biases. Sensor bias estimation and compensation algorithms are therefor employed to estimate and reduce the effects of these errors on tracking performance. Many state of the art sensor bias estimation algorithms rely on a specific measurement model, bias model, number of sensors, fusion architecture, full-rate synchronous communication, or have a high computational cost. Although there exist effective algorithms, they often cannot be applied at all outside of their niche. This is an important limitation, since practical tracking systems often involve multiple heterogeneous sensors, may use a variety of fusion architectures, have limited or unreliable communication, and limited computational resources. It is sometimes possible to use combination of niche approaches to address this, but this makes constructing a practical system an exercise in patchwork. It also means that any changes to the system can require substantial effort to accommodate. Chapter 3 proposes a flexible approach to sensor bias estimation and compensation that can be used with minimal modification in a wide range of operational scenarios, with a reasonable computational cost. The proposed approach allows for sensor bias estimation to be performed in some scenarios where it was not previously possible, as well as providing an option that substantially eases integration in practical complex systems.

Tracking systems often opt to track targets in the world frame of reference using converted measurements (as in chapter 2) or the Extended Kalman Filter, but sometimes this is not possible due to system limitations. For example, using a single a passive imaging sensor where the sensor location and orientation is not known. One alternative that is often applied is to track targets directly in pixel coordinates. This introduces the need to compensate for sensor motion between frames. The Homography Transform is normally used for this purpose in video tracking. In its standard form this transformation has no notion of uncertainty and can only be used to transform position points from their location in one frame to their corresponding location in the following frame. This is a problem if the targets are being tracked using a Kalman Filter, which maintains target state estimates in the form of Gaussian random variables and often tracks other state components in addition to position (such as velocity, acceleration, and turn rate). An extension exists to apply the Homography Transform to position and velocity components of the state estimate mean, and to the elements of the state estimate covariance matrix corresponding to the position state components. This allows for partial transformation of the state estimate of a target being tracked using a Nearly Constant Velocity kinematic model. In chapter 4 a further extension is proposed to allow for the full transformation of target state estimates for three common kinematic models, designed to be readily extensible to other models. This expands on the capability of existing approaches to transform state estimates for additional motion models, as well as allowing a more complete transformation of state estimates for Nearly Constant Velocity models.

The remainder of this thesis is organized as follows. A brief summary of the thesis contributions is given in section 1.2, along with a listing of the papers on which the following chapters are based. Chapters 2 through 4 each cover the one of the problems introduced above, in full detail. Each of these chapters includes a thorough introduction to the corresponding problem, the limitations of existing approaches, proposed solutions, experiments to validate the proposed approaches, and related discussions. A conclusion is presented in chapter 5.

1.2 Thesis Contributions

The contributions of this thesis are as follows:

- A novel image processing algorithm for estimation of the number of components of a Gaussian Mixture distribution, appropriate for use in terrain-aided tracking (chapter 2).
- Fully describes a tracking framework for terrain-aided tracking with a Gaussian Mixture measurement model (chapter 2).
- A novel, efficient algorithm for sensor bias estimation and bias-compensated tracking that can be used with a variety of sensor models, bias models, fusion architectures, and communication models (chapter 3).
- A novel approach for information fusion under unknown correlation with unequal state vectors that is less conservative than Covariance Intersection and more conservative than the Kalman Filter (chapter 3).
- A novel algorithm for sensor motion compensation when tracking in pixel coordinates that can transform full state estimates between frames as long as the state vector can be encoded as a series of position points and subsequently decoded, analogous to the Homography Transform (chapter 4).
- Derivations to encode the state vectors of three common kinematic motion models as a series of position points and subsequently decode them (chapter 4).

Chapters 2, 3, and 4 of this thesis are each based on the work done initially for a research paper. The full citation for each is given below for reference and to give due credit to the co-authors.

[2] D. Schonborn, T. Kirubarajan, R. Tharmarasa, M. McDonald, and M. Bradford, "Terrain-aided tracking using a gaussian mixture measurement model," *To be submitted to IEEE Transactions on Aerospace and Electronic Systems (July 2022)*, 2022.

[3] D. Schonborn, T. Kirubarajan, and R. Tharmarasa, "Sensor bias estimation and measurement compensation for multisensor-multitarget tracking," *To be submitted to IEEE Transactions on Aerospace and Electronic Systems (July 2022)*, 2022.

[4] D. Schonborn, T. Kirubarajan, and R. Tharmarasa, "Unscented homography transform for low frame rate tracking in pixel coordinates," *To be submitted to IEEE Transactions on Aerospace and Electronic Systems (July 2022)*, 2022.

Chapter 2

Terrain-Aided Tracking Using a Gaussian Mixture Measurement Model

2.1 Introduction

The goal of target tracking is to estimate the state of a target (or multiple targets in multi-target tracking) using sensors. Sensors capture data at the signal level, which is processed by a detector to obtain position measurements of the targets (also known as detections). These measurements are then processed by the tracker which initializes and maintains estimates of the target states. Figure 2.1 shows how a terrain reference module (used for terrain-aided tracking) can fit into such a tracking framework.

Video cameras are commonly used in air-to-ground tracking applications due to their low cost, low size, and high accuracy. In combination with a detection algorithm, video data can measure the position of a target accurately within the image plane, but often times the interest is in determining the position of a target in geographic coordinates. Geographic localization of targets presents a challenge due to the lack of range information available from video data. One way to estimate the range for targets on the ground is to follow the line of sight from the sensor in the direction of the measured object until the ground location is reached based on some model of the earth's surface [5, 6, 7, 8]. Examples of earth surface models include spherical [9, 10], ellipsoidal [10], and flat earth models [8] as well as more complex Digital Elevation Models (DEMs) [11, 12, 13, 8].

In [14, 15] terrain data is used to assess terrain traversability and constrain target motion to areas where terrain can be traversed. In [16, 17] terrain data, including road network information or other classification information, is used to constrain target motion to areas that are likely to be driven on. These approaches are all forms of terrain-aided tracking, however this chapter focuses on air-to-ground target tracking using the line of sight method with a surface model represented as a digital elevation matrix.

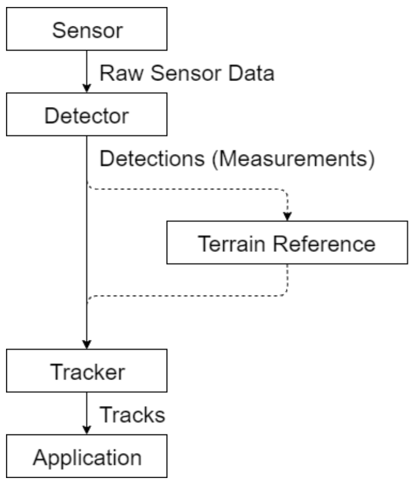

FIGURE 2.1: A tracking framework with optional terrain reference module for terrain-aided tracking

In [5], a DEM in combination with measurements of the sensor's position and orientation are used to add range information to the bearing measurement. This, in conjunction with a Kalman filter, is used to track the target. Similar strategies are employed in [18, 19, 6, 8, 20, 21, 7, 22, 23].

Figure 2.2 illustrates the basic idea of the line of sight tracking. The line of sight direction is measured by orientation sensors and the sensor position is measured by GPS [5]. Starting from the sensor's position, the line-of-sight is followed until it intersects with the DEM [5]. The point of intersection in the DEM is used to calculate the range to the target geometrically, and some Gaussian noise (correlated with the angular measurements to the target) was assumed [5]. Derivations for the geolocation estimate and error covariance were given for both a flat earth model and a DEM in [8]. These derivations were based on the assumption that sources of error in the geolocation estimate are errors in the sensor's position, Euler angles to the target, and terrain height. These errors are assumed to be zero-mean and Gaussian [8].

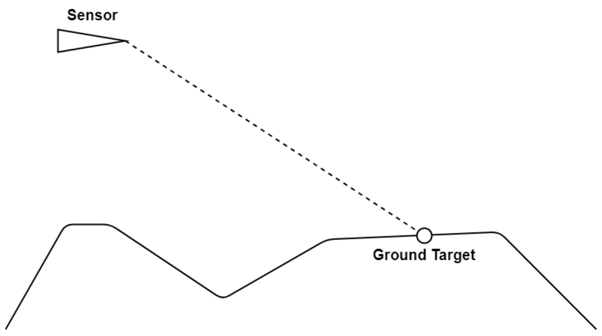

FIGURE 2.2: A basic example of line of sight tracking using a digital terrain map

A limitation to the use of DEM tracking is that the assumed distribution of the geolocation measurement (or the augmented range measurement if tracking in spherical coordinates) is Gaussian [5, 18, 21]. In practice, some terrains are peaky and the grazing angle to targets may be low, leading to a multimodal distribution of the target's position measurement due to uncertainty in the sensor's position and orientation, as well as errors in the DEM [18, 24]. Figure 2.3 illustrates the practical scenario described above. Due to the near-obstruction of line-of-sight between the sensor and the target by peaky terrain, an error in the DEM can lead to a false estimation of the target's position to be nearer than its actual position. These measurement errors may lead to additional problems in the tracking stage such as reduced track accuracy, broken tracks, track swaps, increased confirmation latency, and formation of multiple tracks for the same target (in multi-target tracking).

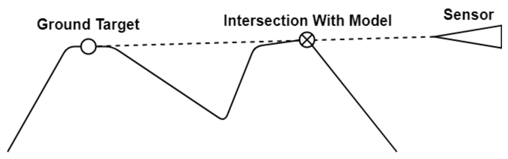

FIGURE 2.3: Illustration of multimodal distribution due to low grazing angle and peaky terrain

In [6] the unscented transform and Unscented Kalman Filter (UKF) are used to partially address this limitation. The distribution of the target position was estimated as Gaussian, which is convenient for integration with existing tracking frameworks. However, it did not capture the multimodal nature of the distribution. The unscented transform yielded an error covariance that accounted for the interaction between the measurement errors and the varying terrain height around the target position [6]. This is unlike the derivations from [8] which only accounted for the terrain height and its error covariance at the measured target position (where the measurement mean intersects the terrain surface).

CONDENSATION algorithm, a particle-based algorithm, has been suggested [18, 21] to address the multimodal distribution issue but has not been implemented as concerns of increased computational complexity abound when used for multiple target tracking involving large number of particles [25]. Additionally, the CONDENSATION algorithm requires a fully integrated tracking process which cannot fit easily into most existing tracking frameworks.

Traditional tracking pipelines can improve terrain-aided tracking results by selecting an appropriate flight path and avoiding conditions that cause difficulty [26]. However, due to flight path restrictions or the requirement to track multiple targets, the approach suffers a limitation as the optimum flight path may not be available or well-defined.

This chapter presents an algorithm that accurately tracks a target using passive, bearing-only sensors combined with terrain information and measurements of the sensor's position and orientation. It addresses the limitations outlined above by presenting a method that is robust to conditions such as low grazing angle, difficult terrain, or having only a low resolution DEM available. The proposed algorithm recognizes when the target's position is ambiguous and it does not depend on a particular target flight path. It also allows for multi-target tracking and performs computations efficiently, suitable for online use in realistic scenarios.

The contributions of this chapter are as follows. Firstly, an algorithm is proposed to efficiently parameterize a Gaussian mixture measurement distribution for the purposes of terrain-aided tracking measurements. It can be used to estimate the number of mixture components, addressing the issue of unknown number of mixture components and allowing standard methods to be used. The algorithm can also be used to cluster samples from the measurement distribution to obtain an estimate of the distribution parameters using simpler methods (based on the sample mean, sample covariance, and weight of each cluster). Secondly, a framework that integrates the Gaussian mixture measurement model into an existing terrain-aided tracking frameworks using components that are already in use while requiring minimal modification. The framework can switch between the Gaussian mixture measurement model and the standard Gaussian measurement model when appropriate. Lastly, simulation results are presented to validate the performance of the Gaussian mixture measurement model under difficult tracking conditions.

The remainder of the chapter is organized thus. Section 2.2 provides background on the subject of digital elevation models, terrain-aided tracking, and multiple hypothesis tracking. It also outlines the theoretical framework for this work. Section 2.3 presents the mathematical context for the problem under consideration. Section 2.4 presents the details of the proposed fast algorithm for parameterization of a Gaussian mixture measurement distribution appropriate for use in terrain-aided tracking. In section 2.5 simulated results are presented to compare tracking performance using the proposed measurement model against those using a Gaussian measurement model. Section 2.6 concludes this work, evaluating the significance of results from the previous section, identifying limitations of the proposed approach, and directions for future work.

2.2 Background

To fully understand the proposed algorithms, it is essential to present background discussions for digital elevation models, surface model intersections, Gaussian mixture models, multiple hypothesis tracking, and morphological image processing.

2.2.1 Digital Elevation Model (DEM)

A Digital Elevation Model (DEM) is a database used to obtain an estimate of terrain elevation for a given pair of geographic coordinates (latitude and longitude). DEMs have important applications in Geographic Information Systems (GIS) [27], target tracking, and many others. This chapter focuses on data represented by a Digital Elevation Matrix which are compact and available for public download [11, 13].

A Digital Elevation Matrix consists of a set of regularly-spaced "posts", each with an estimate of the elevation at that post. Each post is associated with a geographic location in a specified projection to enable conversions between geographic and local coordinate system using standard methods. Between the posts, no information about the terrain height is provided, as illustrated by figure 2.4.

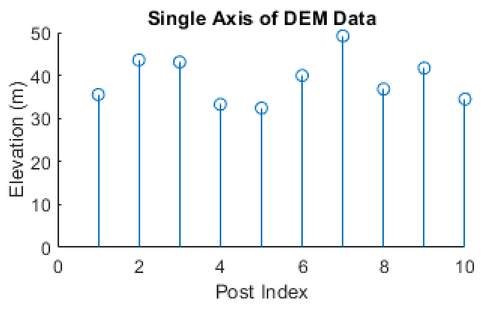

FIGURE 2.4: A representation of a single axis of DEM data with elevation measurements at posts, and no information in between.

The elevation at points between the posts can be approximated using bilinear interpolation of the neighboring four posts. Results from [28] suggests that the bilinear interpolation provides reasonable accuracy and computational efficiency for this purpose. Hence, the terrain surface modelled by the DEM is a grid of bilinear patches. Equation 2.1 represents such a patch, where z_W is the elevation at a point on the patch, z_{ij} are the elevations provided by the neighboring posts with increments of i, j corresponding to increments in the grid in the x, y directions, respectively.

$$z_W = w_{y0}(w_{x0}z_{00} + w_{x1}z_{10}) + w_{y1}(w_{x0}z_{01} + w_{x1}z_{11}) \tag{2.1}$$

The weights w_{x0}, w_{x1}, w_{y0}, w_{y1} correspond to the distance between the point on the surface and the neighboring posts in the x and y directions respectively for the lower and upper posts (this is normal bilinear interpolation).

2.2.2 Surface Model Intersections

Terrain-aided tracking by line of sight is based on the intersection of the line of sight (a ray from the sensor position in the direction of the target) and the DEM surface. Determining the first point of intersection along this ray will be required for such applications. Since the DEM is a grid of bilinear patches, this task is divided into a grid traversing over patches in the line of sight and checking each patch for an intersection.

Ray-Grid Intersection

In [18] the DEM is intersected by traversing the grid cell-by-cell in the x, y coordinates starting from the cell containing the sensor until a point of intersection is found where the elevation in the cell is greater than the z component of the line of sight as it passes through that cell.

In [29], a modified Bresenham Digital Differential Analyzer (DDA) algorithm was presented to traverse a grid along a line. Their algorithm identified all cells in the grid that intersect the line, from the starting point as illustrated in figure 2.5. The elevation component can be rejected for intersections without an exact intersection test (if the elevation of the ray does not pass through the elevation of the cell), otherwise an exact test is performed [29].

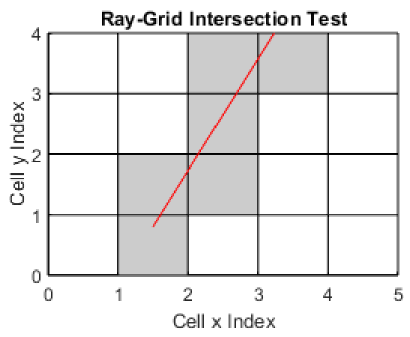

FIGURE 2.5: An illustration of the cells identified by the modified Bresenham DDA algorithm.

Ray-Bilinear Patch Intersection

In [18] it was suggested to use interpolation to obtain a more precise point of intersection within the intersecting cell, but no details about the interpolation approach or intersection algorithm were included. An efficient algorithm for intersecting rays with a bilinear interpolated patch was presented in [30]. A patch may also have multiple intersections, in which case the first (nearest to the sensor) represents the correct point in the line of sight [30].

2.2.3 Gaussian Mixture Models (GMMs)

Mixture models can be used to represent a multimodal distribution as a weighted sum of K component distributions. When the component distributions are all Gaussians, then the distribution is called a Gaussian mixture (or a mixture of Gaussians). GMMs are a desirable way to represent multiple hypotheses in a target tracking scenario because they fit conveniently within the Bayesian recursive estimation framework for Multiple Hypothesis Tracking (MHT) (see section 2.2.4). Equation 2.2 is the general definition of a Gaussian mixture distribution, where μ_i, C_i are respectively the mean and covariance of each Gaussian component distribution.

$$p(x) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\mu_i, C_i) \tag{2.2}$$

Various general-purpose approaches exist for estimating a GMM from a set of samples. Two popular approaches are the K-means (KM) and Expectation Maximization (EM) algorithms [31, 32, 33]. Both are closely related as they are center-based clustering algorithms that involve initializing clusters and updating their centroids based on their relationship with the samples [33]. KM algorithms cluster the samples based on their Euclidean distance from each cluster centroid and use the mean of each cluster's samples to update the centroid for the next iteration, continuing until a stable configuration is reached [32]. EM algorithms work in a similar iterative manner, but also considers the probability of each sample's membership to each cluster and may use a more complex distance metric [32]. Both algorithms benefit from good cluster initialization.

KM and EM in their standard forms assume that the number of components is known. In [33], an algorithm was proposed to learn the K value in KM and suggest a method for evaluating the model fit with each choice of K. Their algorithm involves K_{max} iterations for the evaluation of KM. Assuming a constant number of dimensions, the computational complexity of the KM algorithm is $\mathcal{O}(Kni)$ where K is the number of components, n the number of samples, and i the number of iterations [34]. If K_{max} iterations are performed to determine K then then total computational complexity is $\mathcal{O}(K_{max}^2 ni)$. Note that K_{max} is the number of iterations of the entire KM/EM algorithm, each assuming a different fixed value for K, while i is the number of iterations performed within each algorithm run. The K_{max}^2 term arises from the fact that K varies from 1 to K_{max}. This complexity may be prohibitive for real time use in tracking applications where a high frame rate is required and when K_{max} is large. Additionally it is possible that the resulting mixture with the best fit may involve an excessive number of components that do not correspond well to different hypotheses in a tracking context (see section 2.4.3). Section 2.4.3 of this chapter proposes an alternative method to cluster samples and determines a reasonable number of components to use in the context of terrain-aided tracking.

2.2.4 Multiple Hypothesis Tracking (MHT)

In [35], it was demonstrated that Multiple Hypothesis Tracking (MHT) could be an effective method for resolving ambiguities in visual tracking problems. With MHT, the final decision on data association was delayed by maintaining a tree of all hypotheses originating from a particular observation, though in practice the tree must be pruned aggressively for MHT to remain computationally feasible. In each frame, the tree was updated as each hypothesis is associated with new observations as well as a dummy observation (indicating a missed detection). The most likely global hypothesis at each frame was then formed with the constraint that each observation is either a false alarm or associated with a single track (i.e. there are no unresolved targets).

The approaches in [35, 36, 37] all utilized MHT for multi-target tracking. Additionally [36, 37] used MHT to resolve ambiguities with the measurement distribution (in terms of Doppler measurements and grating lobes). The same approach will be applied in this chapter to resolve ambiguity in terms of the Gaussian mixture measurement modes as well as in measurement-to-track association (for multi-target tracking) without much modification.

Track Trees

In MHT, the track hypotheses originating from each observation are represented by track trees, constructed per frame for every observation (detection) to consider the possibility that new targets are being observed [35]. In subsequent frames, these trees are updated by adding hypothesis branches with increasing depth corresponding to the different possible association outcomes originating from the root observation of each tree. In a particular tree, each directed path from the root to a leaf of the tree is one hypothesis corresponding to the initial observation (the root). These hypotheses are each given a track score, which is used in pruning the tracks as well as computing the global track hypothesis [35, 36, 37]. The track tree can be pruned based on an N-scan approach and kept to a maximum number of branches to keep computation feasible [35]. The N-scan approach involves looking back N frames in the tree and pruning branches that are not consistent with the current global hypothesis. The maximum size can be enforced by keeping only the top hypotheses based on their track scores. Additional logic-based (i.e. "M out of N") or score-based track management can be employed to confirm or terminate tracks.

At each frame, existing hypothesis trees are updated to account for new observations of previously-observed objects and the possibility of missed detections. New hypothesis trees are formed to consider possible newly-detected objects [35]. For objects that have been observed before, existing track hypotheses (in all trees) are updated by appending a new branch for each new observation that lies within their gating region (where the Mahalanobis distance between the observed and hypothesis predicted location is less than a specified threshold). This is seen in equation 2.3 where d_M is the Mahalanobis distance between the predicted hypothesis (with mean μ_x and covariance Σ_x obtained using a Kalman filter) and the observed location μ_y. The threshold can be adjusted to increase or decrease the size of the gating region. This gating procedure reduces the total number of hypotheses to consider.

$$d_M = (\mu_x - \mu_y)^T (\Sigma_x)^{-1} (\mu_x - \mu_y) \tag{2.3}$$

In [36, 37] this gating test based on Mahalanobis distance was applied for all combinations of existing hypotheses with each ambiguous case (i.e. the Doppler measurement or grating lobe) from all new measurements. A new hypothesis is formed for each combination passing the test. In the standard MHT [35], each measurement generates only one hypothesis to test per existing hypothesis, while in the formulations from [36, 37] each measurement generates one or more hypotheses to test for each existing hypothesis.

In [36], an equation for the probability of each hypothesis is derived using Bayes' theorem and shown below in equation 2.4. The first factor in the product correspond to the measurements' likelihood (including the Doppler measurement) given the hypothesized association. The second factor is the probability of the current hypothesis based on the previous one, which includes both the measurement-to-track assignment and the Doppler order assignment. The final factor is the probability of the parent hypothesis, and c is a normalization constant chosen so that the sum of the probabilities for all hypotheses is 1.

$$P\{A^{k,l}|Z^k\} = \frac{1}{c} p[Z(k) \,\Big|\, a(k), A^{k-1,s}, Z^{k-1}] \cdot P\{a(k) \,\Big|\, A^{k-1,s}, Z^{z-1}\} \cdot P\{A^{k-1,s}\} \quad (2.4)$$

A simplified final expression is repeated in equation 2.5. The probability depends on the number of new measurements $m(k)$, number of associated measurements N_D, number of false alarms N_{NF}, density of new measurements or clutter μ_{n+f}, the number of existing targets N_{TGT}, probability of detection P_D, tracking volume V, and the likelihood of each measurement given the hypothesis $f_{ti}(\cdot)$ (Kalman Filter innovation PDF).

$$P\{A^{k,l}|Z^k\} = \frac{N_D!}{c \cdot m(k)!} \mu_{n+f} \cdot P_D^{N_D} \cdot (1 - P_D)^{N_{TGT} - N_D} \cdot V^{-N_{NF}} \cdot \prod_{i \in a_T(k)}^{m(k)} f_{ti}(Z_i^U, k) \quad (2.5)$$

In the derivation, the Doppler distribution is assumed to be uniform so its contribution is absorbed into the normalization constant c. However in [37], some information about the strengths of the possible grating lobes was considered. Since this distribution is no longer uniform, an additional factor is added to the equation. This factor corresponds to the weight of the lobe for the lobe hypothesis, but can also be the weight ϕ_i from a Gaussian mixture component to hypothesize about different modes.

Global Hypothesis

Forming a global hypothesis is achieved by determining the most likely set of compatible tracks based on all of the the current hypotheses. As described in [35], the formation of a global hypothesis is equivalent to solving the Maximum Weight Independent Set (MWIS) problem on a graph with nodes for each hypothesis and edges between nodes which are in conflict. Solving this problem is equivalent to solving the Maximum Weight Clique problem on the complement graph. The complement graph used in the Maximum Weight Clique problem places edges between nodes which are compatible, instead of placing edges between nodes which are in conflict (as in MWIS). The Maximum Weight Clique problem can be solved using a number of different exact or approximate algorithms, and the simulations in this chapter use the algorithm from [38]. When considering track compatibility, a pair of tracks will be considered incompatible if they both involve the association of the same measurement (multiple targets cannot originate from a single measurement) [35, 36, 37].

2.2.5 Morphological Image Processing

In [39], morphological image processing techniques were used with sparse Lidar data to segment the data points. Their goal was to find structures (walls) in the surrounding area but binning and segmenting the Lidar data directly using Connected Component Labeling (CCL) led to incorrect clustering due to the sparse nature of the data points. Sparsity was addressed by applying a dilation to the bin map before clustering using CCL. This technique will be applied in this work.

Dilation

It expands existing pixels in a binary image with values of 1 so that their neighboring pixels also have values of 1. The region of pixels to be considered as neighbors can be defined by specifying a structuring element matrix [39]. By enlarging these regions in the image, nearby regions may join one another, addressing the data sparsity issue. See figure 2.9 to see the effects of dilation on CCL as applied in this chapter.

Connected Component Labeling (CCL)

It is used to label clusters of pixels with values of 1 in binary images that are connected. Pixels are considered connected if a path exists between adjacent pixels with values of 1 according to some neighbor relation (usually 4-way or 8-way). Each pixel is assigned a label such that pixels sharing the same label belong to the same connected component. See figure 2.9 for an example of labeled connected components (with and without dilation). A review of conventional CCL algorithms as well as a faster algorithm can be found in [40]. This map of labels is used in [39] to cluster the data points.

2.3 System Model and Considerations

Ground point targets moving through rough terrain are tracked using a passive sensor from an airborne platform equipped with sensors to measure its own orientation and position. The platform is also equipped with a Digital Elevation Model (DEM) of the tracking area. The information from these sources will be used to estimate the position distribution of the detected object based on the method proposed in section 2.4. Consideration of the assumptions, reference coordinates, scenario, sensor measurements and target motion model are discussed below.

2.3.1 Assumptions

The sensor is assumed to be able to measure its own position and the angles to the targets (elevation and azimuth) with zero-mean Gaussian noise. The DEM is assumed to provide the correct elevation for a given set of geographic coordinates, though this is not the case in reality (the effect of this assumption is investigated by considering simulation results using multiple DEMs with varying resolution and accuracy in section 2.5).

Multiple targets are considered in the tracking stage, but this chapter focuses on modeling the terrain-aided detection distribution. At this stage, the targets are considered individually based on the assumption of a one-to-one correspondence between targets and true measurements (no unresolved targets, no extended targets).

2.3.2 Reference Coordinates

The targets are tracked in the local frame of reference, that of the DEM, with units in meters. Conversions back and forth between the local frame of reference and geographic coordinates are straightforward using standard tools (see section 2.2.1). These conversions are necessary for practical applications, for example to bring tracking information back to geographic coordinates for the end use application. They are used for the simulations discussed in section 2.5.

2.3.3 Scenario

A target located at position X_T is observed by a sensor located at position X_S with viewing direction vector D_T (a vector in the direction of the line of sight between the sensor and the target). Three dimensions are considered. The scenario is illustrated in figure 2.6.

$$X_T = [x_T, y_T, z_T]^T \tag{2.6}$$

$$X_S = [x_S, y_S, z_S]^T \tag{2.7}$$

$$D_T = [x_D, y_D, z_D]^T \tag{2.8}$$

The target is a ground point target so its position lies on the terrain surface. The elevation of the terrain is represented by the function $z_W(x, y)$, modeling a 3D surface where the elevation z_W at each point in the x, y plane is defined by the DEM (see section 2.2.1 for details).

2.3.4 Sensor Measurements

As assumed (see section 2.3.1), the sensor measurement probability distribution function (PDF) is given by equation 2.9 where Z_S is the sensor measurement vector and C_S is the sensor measurement noise covariance matrix. Note that the angle to the target is measured in terms of azimuth θ_T and elevation φ_T, so the measurement noise is Gaussian in those variables.

$$Z_S = [x_S, y_S, z_S, \theta_T, \varphi_T]^T + \mathcal{N}(0, C_S) \tag{2.9}$$

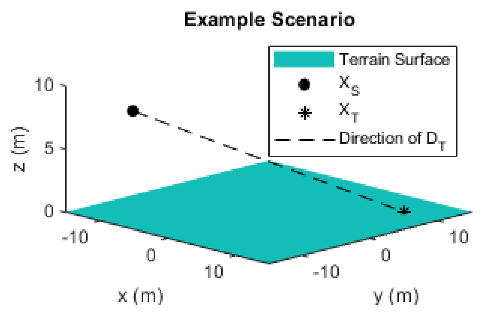

FIGURE 2.6: An illustration of the sensor position X_S, target position X_T, and direction vector from the sensor to the target D_T. D_T starts from X_S and extends to X_T.

Intuitively the sensor is measuring the line of sight to the target. The measured variables are combined to form the line of sight L in equation 2.10 where t is a free parameter that moves along the line and D_T is obtained through the standard spherical to cartesian conversion.

$$L = X_S + tD_T \tag{2.10}$$

Figure 2.7 shows samples from the distribution of L which represent possible lines of sight based on the information from the sensor measurements.

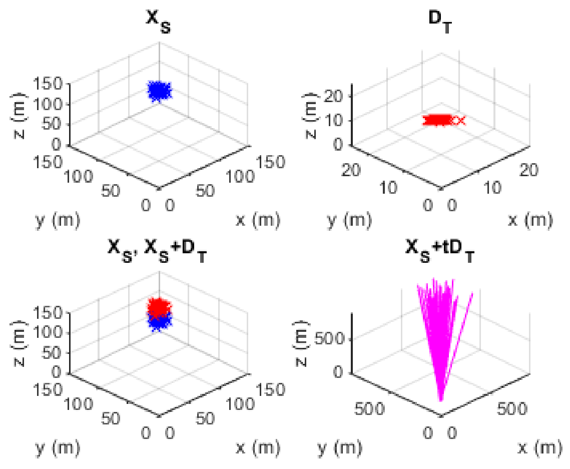

FIGURE 2.7: Samples from the distributions of random variables related to the line of sight from the sensor to the target.

2.3.5 Target Motion Model

The methods proposed for parameterizing the measurement distribution in this chapter are not dependent on a specific target motion model, so any motion model can be used. A constant velocity model with process noise, chosen for its ease of implementation, is used in the simulations presented in section 2.5. The proposed method involves a multimodal measurement model split into several Gaussian hypotheses for Multiple Hypothesis Tracking as described in section 2.2.4. This has the effect of increasing the number of hypotheses in the tracking stage, however each hypothesis can still be tracked in the normal way. This approach allows for target state predictions to be evaluated in the normal way with respect to the target motion model. Maneuvering targets could also be handled in the same way with minor modifications to existing methods based on IMM-MHT (such as in [41]) through the introduction of additional hypotheses generated by each component of the measurement.

2.4 Sample-Based Terrain-Aided Measurements

2.4.1 Sample Representation

A set of n samples s_i are drawn from the distribution of L (equations 2.11 to 2.14). Each sample represents a possible line of sight from the sensor to the target. Let L^* be the set containing these samples as in equation 2.15. These line of sight samples are visualized in figure 2.7.

$$i \in 1,\ 2,\ ..,\ n \tag{2.11}$$

$$l_i = x_i + t d_i \tag{2.12}$$

$$x_i \leftarrow X_S \tag{2.13}$$

$$d_i \leftarrow D_T \tag{2.14}$$

$$L^* = \{l_1,\ l_2,\ ..,\ l_n\} \tag{2.15}$$

The target is assumed to be a ground point target, so the target position is the nearest location to the sensor where the line of sight intersects with the terrain surface. Sample points of intersection s_i are given by equation 2.16.

$$s_i = x_i + t_i d_i \tag{2.16}$$

t_i is chosen as the minimum positive t value (in front of the sensor) where the sample line l_i intersects the terrain surface X_w. This is expressed as an optimization problem in equation 2.17.

$$\begin{aligned} \min_{t} \quad & t && \text{(2.17a)} \\ \text{subject to} \quad & x_i + td_i = X_w, && \text{(2.17b)} \\ & t > 0 && \text{(2.17c)} \end{aligned}$$

Recall that the world surface X_w is modelled as a grid of bilinear patches. The algorithm presented in [30] and reviewed in section 2.2.2 can be applied to find the point of intersection with the minimum t for each bilinear patch. Since the DEM has a finite number of patches, it is possible in theory to find the minimum t in equation 2.17 by searching each patch in the DEM to determine all possible values of t that meet the constraints and then selecting the smallest one. Such an approach presents computational concerns if the DEM is large, so it is desirable to avoid searching every patch.

Patches of interest are those that intersect the ray in the x, y coordinates, as seen in figure 2.5. Let patch i, j be the bilinear patch between posts i, j and $i+1, j+1$. The x, y coordinates of the ray starting point are converted to pixel coordinates for the purposes of grid traversal, as in equation 2.18 where s_{pixel} is the pixel size (DEM post spacing or resolution) in meters.

$$[x_{pixel}, y_{pixel}] = [x_{local}/s_{pixel}, y_{local}/s_{pixel}] \tag{2.18}$$

The sample ray in pixel coordinates is then intersected with the DEM at the cell level using the algorithm presented in [29] and described in section 2.2.2 to determine candidate cells. Each of the candidate cells are tested for exact intersection as described in section 2.2.2 until a true intersection is found. Since only the nearest point of intersection for each line is required the process stops when one is found. This method is used for each line of sight sample l_i in L^*.

Each sample s_i represents one possible target location, in global coordinates. Let S^* be the set containing these samples as in equation 2.19. These samples will be used to estimate the distribution of the target position.

$$S^* = \{s_1,\ s_2,\ ..,\ s_n\} \tag{2.19}$$

2.4.2 Gaussian Model

Once samples S^* of the target position are generated as in section 2.4.1, a Gaussian approximation of the distribution can be formed. This is done by taking the sample mean and sample covariance of S^*. The sample mean is computed by equation 2.20. The sample covariance matrix is computed by equation 2.21 where F is a matrix with rows corresponding to the x, y, z components of the samples and the columns corresponding to the samples (as in equation 2.23). This is similar to the method from [6] where the unscented transform is used, and will be used as base for comparison.

$$\mu_p = \frac{1}{n}\sum_{i=1}^{n} s_i \tag{2.20}$$

$$C_p = \frac{1}{n-1}(F - \mathbf{1}_n\mu_p^T)(F - \mathbf{1}_n\mu_p^T)^T \tag{2.21}$$

$$\mathbf{1}_n = [\underbrace{1,\ 1,\ ..,\ 1}_{n\text{-times}}] \tag{2.22}$$

$$F = [s_1,\ s_2,\ ..,\ s_n] \tag{2.23}$$

2.4.3 Gaussian Mixture Model

Similar to the Gaussian model, the samples taken are used to approximate the measurement distribution, but the distribution here is a multivariate Gaussian Mixture model. The distribution for this model is given in equation 2.24 where K is the number of components of the mixture model and ϕ_i, μ_i, C_i are respectively the weight, mean, and covariance matrix corresponding to each mixture component indexed by i (see section 2.2.3).

$$p(x_T) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\mu_i, C_i) \tag{2.24}$$

To form the Gaussian mixture distribution, it is necessary to determine the number of components K, and then estimate the weight, mean, and covariance matrix for each component. A method is proposed below to cluster the samples into sub-populations for formation of a Gaussian mixture while simultaneously estimating for the number of components K. Once the samples are clustered, the samples from each cluster will be used to estimate the parameters for one component of the Gaussian Mixture.

Approximate Clustering and Determination of the Number of Components

The clusters can be formed by partitioning the x, y space, and each sample assigned to a unique cluster based on its position in x, y coordinates. During the sampling process described in section 2.4.1, a binary mask M, the same size as the DEM is produced. When each sample is taken, the mask pixel corresponding to the DEM cell where the intersection occurred is set to 1. This relationship is expressed in equation 2.25. In this equation, s_{pixel} is the sample converted to pixel coordinates, which is then floored to yield its integer cell coordinates in the DEM.

$$M_{x,y} = \begin{cases} 1 & \exists s \in S^* \text{ where } \lfloor s_{pixel} \rfloor = [x,y] \\ 0 & otherwise \end{cases} \tag{2.25}$$

Figure 2.8 shows an example of such a binary mask. While the DEM can be quite large, most of the values in the mask will be zero for realistic measurements. To save memory and computation time, the mask $M_{x,y}$ can be represented as a sparse matrix where only cells containing non-zero values are actually stored.

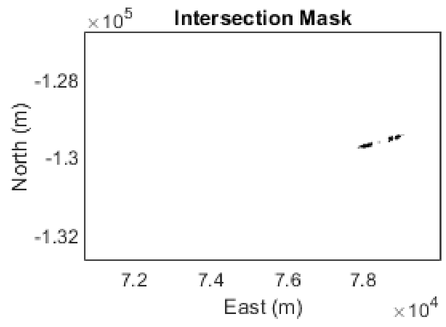

FIGURE 2.8: Binary mask of the DEM cells that contain points of intersection

A two-pass connected-component labelling algorithm is applied to this mask to cluster the DEM cells. By using a sparse matrix representation of the mask, it is possible to avoid iterating over the whole size of the DEM. Only mask cells with non-zero values and their neighbors need to be considered, reducing the total computation time.

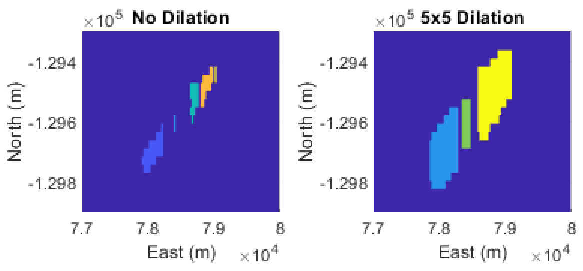

FIGURE 2.9: DEM cells containing points of intersection, clustered by connected component labelling with no dilation (left) and with dilation using a 5x5 structuring element matrix of ones (right)

Depending on the sample density and pixel size of the DEM, this may result in a large number of clusters being formed. Having many clusters is not desirable from a computational perspective since each cluster will form a hypothesis that must be considered for tracking. Hence, clusters that are close together are merged to reduce the overall number of components. This can be achieved through binary image dilation of the mask M before the clusters are formed, as seen in figure 2.9. Through direct labeling, six components are found. But with dilation applied first, several components are merged resulting in only three components. This addresses data point sparsity in connected component labeling the same way as in [39].

Gaussian Mixture Parameterization Using Cluster Sample Mean and Sample Covariance

After approximate clustering, each of the K components of the Gaussian mixture can be parameterized using samples from each of the K clusters in the same method that was used for a single Gaussian in section 2.4.2. That is, the mean and covariance for each component are estimated using the sample mean and sample covariance for the samples in the corresponding cluster as in equations 2.26 and 2.27.

$$\mu_i = \frac{1}{m_i}\sum_{j=1}^{m_i} s_j \tag{2.26}$$

$$C_i = \frac{1}{m_i - 1}(F - \mathbf{1}_{m_i}\mu_p^T)(F - \mathbf{1}_{m_i}\mu_p^T)^T \tag{2.27}$$

Here $F_i, \mathbf{1}_{m_i}$ are defined as in section 2.4.2 (but using the samples from cluster i), m_i is the numbers of samples in each component (cluster) i, and s_j is the corresponding samples in that component with $j \in \{1, 2, .., m_i\}$. To complete the formation of the Gaussian mixture, it is also necessary to determine the weight for each component. Each component is weighted according to the relative frequency of samples occurring in that component, as in equation 2.28. This guarantees that the sample weights ϕ_i will sum to 1 as is required for a mixture model.

$$\phi_i = \frac{m_i}{n} \tag{2.28}$$

A limitation of this approach is that some of the clusters may have a small number of samples. This increases the likelihood that the computed sample covariance matrix is not positive-definite, resulting in a non-invertible innovation covariance matrix during tracking. Inversion of the innovation covariance matrix is required to compute the Kalman gain during the update stage, so clusters with a small number of samples may be discarded. The overall chance of a singular covariance sample matrix occurring can be reduced by increasing the number of samples used, but this may be undesirable due to the increased computational load it would incur.

Gaussian Mixture Parameterization Using Expectation Maximization

As identified in section 2.2.3, Expectation Maximization is a well-established algorithm for parameterization of a Gaussian Mixture distribution. The main drawback of this approach is that the number of components K needs to be known, or multiple parameterizations must be compared based on some fitment criteria to choose the best. By first applying the algorithm proposed in section 2.4.3 to estimate the number of components and then applying a standard EM algorithm, a parameterization appropriate for terrain-aided tracking can be obtained. The term appropriate is used in the sense that the resulting Gaussian mixture distribution should not have an excessive number of components, and these components should be well-spaced so as to form relatively distinct hypotheses for the target location. The advantages of dynamically selecting a reasonable value for the number of components, compared with selecting a fixed value for K that may be too large, are illustrated through experimental results in section 2.5.

2.5 Simulations

Simulations were conducted to evaluate the performance of different tracking methods in a variety of tracking conditions. These simulations had two main objectives. Each configuration was tested over 5000 Monte Carlo runs in a scenario with a single sensor tracking two targets moving through rough terrain. Performance is evaluated in terms of track accuracy, number of false tracks, track completeness, and computation time. False alarms (uniform randomly distributed in tracking frame) and missed detections are included.

2.5.1 Tracking Methods Compared

The baseline method uses a Gaussian distribution to model the target position measurements, as described in section 2.4.2. Figure 2.10 shows a flowchart of the algorithm used to parameterize the Gaussian measurement.

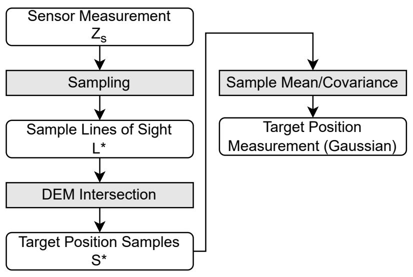

FIGURE 2.10: Flowchart of the algorithm to parameterize a Gaussian terrain-aided measurement

The rest of the methods model the measurement using a Gaussian mixture distribution, but use different methods to parameterize the distribution. Two of these methods use the approximate clustering method (described in section 2.4.3) to estimate the number of mixture components K dynamically (denoted DK, for Dynamic K). One of the DK approaches is the sample mean and sample covariance approach (SMC, GMM-DK-SMC) that parameterizes the components by the cluster sample means and sample covariances, while the other (GMM-DK-EM) uses Expectation Maximization (EM) for this purpose. Figures 2.11 and 2.12 show flowcharts for the GMM-DK-SMC and GMM-DK-EM algorithms, respectively.

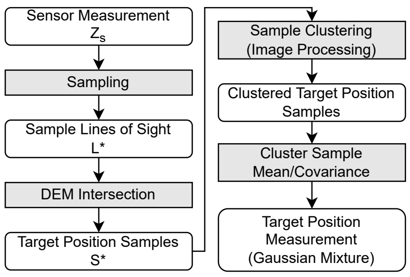

FIGURE 2.11: Flowchart of the GMM-DK-SMC algorithm to parameterize a Gaussian mixture terrain-aided measurement

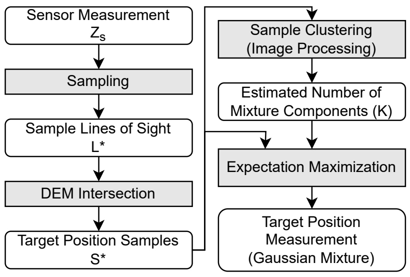

FIGURE 2.12: Flowchart of the GMM-DK-EM algorithm to parameterize a Gaussian mixture terrain-aided measurement

The final method being evaluated (GMM-K6-EM) uses EM with six mixture components, an over-fit of the measurement distribution under most circumstances. Figure 2.13 shows a flowchart for the GMM-K6-EM algorithm. The open source EM implementation used in these simulations is described in detail in [42]. Note that the GMM trackers will sometimes be referred to as simply DK-SMC, DK-EM, and K6-EM for brevity.

Figure 2.14 illustrates how each of the measurement distributions under consideration compares with the distribution of the samples from the measured target position distribution during difficult tracking conditions. Figure 2.14a shows the measurement distribution as sample lines of sight from the sensor to each of the two targets. Figure 2.14b shows the point of intersection of each sampled line of sight with the DEM, corresponding to samples from the distribution of the target positions. These samples are used to parameterize the measurements used in each of the compared approaches.

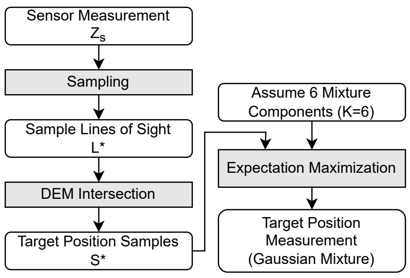

FIGURE 2.13: Flowchart of the GMM-K6-EM algorithm to parameterize a Gaussian mixture terrain-aided measurement

Figures 2.14c through 2.14f show samples from the distributions of the measurements used in each approach, which should ideally be as similar as possible to the position samples in figure 2.14b. In figure 2.14c a Gaussian distribution is assumed and parameterized according to the sample mean and sample covariance of the position samples. These samples are a good match with the northernmost target's position samples, but for the southernmost target, the samples have a substantially different distribution. In figures 2.14d and 2.14e a Gaussian mixture distribution is assumed with the number of components determined dynamically (in this case one component for the northernmost target, two components for the southernmost target). The difference between these two figures is in how the parameters of each Gaussian mixture component are determined, as described above, and the results are similar. In both of these cases the samples from the measurement distribution for the southernmost target are a much closer match with the position samples for this target. In figure 2.14f a Gaussian mixture distribution with six components is assumed. This results in redundant mixture components which overlap one another to the extent that they cannot be discerned visually. Although the samples appear similar in these figures to those obtained from the distributions of GMMs with fewer components, there is redundancy between components and the number of components does not accurately reflect the number of modes in the distribution. Mixture components are considered individually during the tracking process as different hypotheses about the measured target position, so redundancy and extra components are not desirable.

2.5.2 DEMs of the Area Surrounding Crater Lake

Crater Lake (Oregon, USA) was selected as the simulation region because it met several criteria. First, it contained peaky terrain needed to introduce multimodal distributions. Additionally a high resolution DEM is publicly available for use as the terrain ground truth [13], as well as several lower-resolution DEMs [11, 12] more typical of what might be available for general tracking use in various locations around the world. To investigate the effects of DEM accuracy and resolution on tracking performance, several compound DEMs were also created to compare based on the above mentioned DEMs. All the DEMs used are described below.

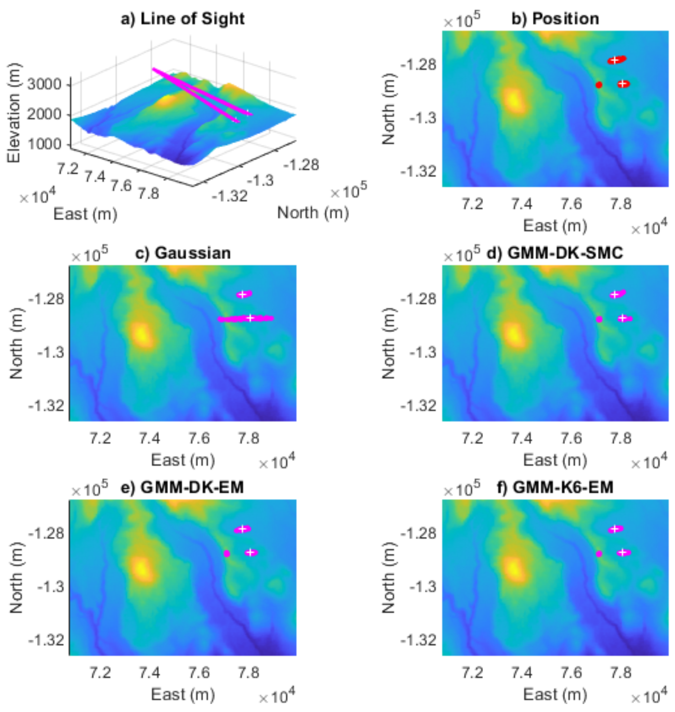

FIGURE 2.14: Samples from the measurement distribution transformed to their corresponding line of sight (a) and target position samples (b), and from the measurement distributions assumed when using the four different tracking methods (c through f). In each plot the true target positions are marked in white.

DEM with 1m Resolution (Ground Truth)

Created using Lidar, this DEM has a resolution (post spacing) of 1 meter [13]. Figures 2.15 and 2.16 show QGIS [27] renderings of the high resolution DEM both in overview to demonstrate the level of detail. This DEM is used as the ground truth due to its significantly higher resolution. It is used to generate the true sensor and target trajectories, and to determine if the sensor has a line of sight to the target. It is assumed that the DEM with 1m resolution is close to the true terrain for the purposes of performance evaluation. This DEM was not used for online tracking due to the associated computational load (it has a large number of cells compared with other DEMs used) and because DEMs at 1m resolution are not currently available for many locations.

DEMs with 30m and 90m Resolution

NASA SRTM data for most of the world is publicly available with approximately 30m and 90m resolutions [11] [12]. Due the wide availability of DEMs at these resolutions, they will be used for the purposes of tracking. Figures 2.17 and 2.18 show QGIS [27] renderings providing an overview of the DEMs with 30m and 90m resolutions respectively.

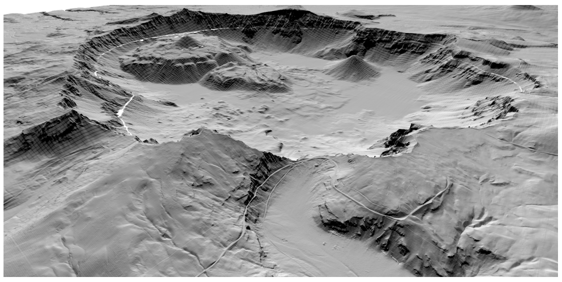

FIGURE 2.15: Challenging terrain surrounding Crater Lake visualized using the DEM with 1m resolution

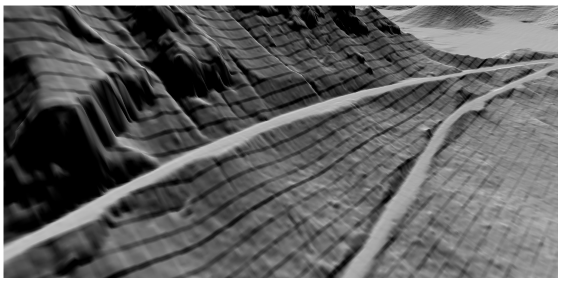

FIGURE 2.16: A view of the level of detail on the 1m resolution DEM of the Crater Lake area

Corrected DEMs with 30m and 90m Resolution

Corrected DEMs were created based on each of the 30m and 90m resolution DEMs. These corrected DEMs retained the same resolution as the models they were based on, but the elevation at each post was corrected to match the true (1m) DEM. These DEMs are intended to provide an upper bound on tracking performance using a DEM of that resolution to give some idea of how much improvement can be obtained by fixing or estimating the post heights.

2.5.3 Sensor and Target Trajectories

The sensor and the target follow an aerial and ground trajectory respectively with a constant velocity model having independent zero-mean Gaussian process noise. The trajectory of the target is determined using the true terrain model. Trajectories of both types are generated independently for each Monte Carlo run.

2.5.4 Results

Performance was evaluated for all the trackers in terms of track accuracy (residual Euclidean error and RMSE), track probability of detection (TPD), and number of false tracks (NFT). An overall summary of the results (averaged over all time points in the simulation) is given in Table 2.1. The average value in Res., RMSE, NFT and TPD

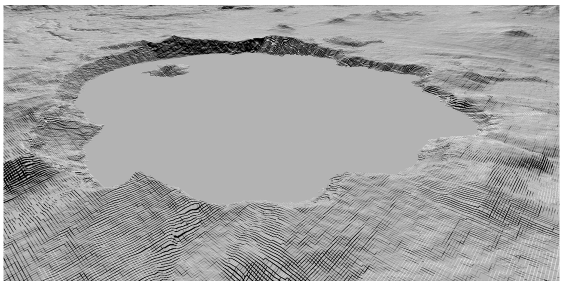

FIGURE 2.17: The Crater Lake area visualized using the DEM with 30m resolution

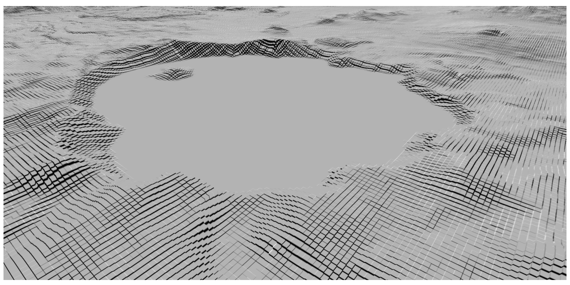

FIGURE 2.18: The Crater Lake area visualized using the DEM with 90m resolution

in Table 2.1 is an average over all time steps, targets, and Monte Carlo runs. It contains only data points where each target is tracked at the corresponding time step, during the specific Monte Carlo run. Note, the best tracker for each metric is marked with (b) while the worst is marked with (w). The GMM-K6-EM tracker formed fewer tracks (both true and false) than the other trackers. It had the lowest NFT, the lowest TPD and the highest track error metrics in all scenarios. The performance of trackers using a Gaussian mixture measurement model with dynamically selected number of components (GMM-DK-SMC and GMM-DK-EM) were similar to one another across all metrics, which is expected since they result in a very similar measurement distribution under most conditions. These trackers outperformed the baseline Gaussian measurement tracker's track accuracy. However, for NFT and TPD, both GMM-DK-SMC and GMM-DK-EM were similar to the baseline Gaussian measurement tracker. When using the corrected DEMs, these trackers performed the same or better than the baseline in the NFT and TPD metrics.

The GMM-K6-EM tracker notably has the lowest NFT in all scenarios but, considering the substantial reduction in TPD from this tracker, this reduction in false tracks seems to be mostly due to a reduced ability to form tracks in general. With the uncorrected DEMs, the NFT slightly increased for the GMM trackers with dynamically selected number of components (GMM-DK-SMC, GMM-DK-EM) when compared with the baseline which is undesirable. The increase in NFT can be explained by considering

30m				
	Gaussian	DK-SMC	DK-EM	K6-EM
Res. (m)	110.779	79.081(b)	79.312	112.403(w)
RMSE (m)	149.584	114.256(b)	114.349	151.696(w)
NFT	1.884	1.912(w)	1.908	0.840(b)
TPD	0.642	0.645(b)	0.645(b)	0.506(w)
30m (corrected)				
	Gaussian	DK-SMC	DK-EM	K6-EM
Res. (m)	75.046	58.186	57.645(b)	98.854(w)
RMSE (m)	108.069	83.162	82.243(b)	138.781(w)
NFT	1.823(w)	1.820	1.808	0.845(b)
TPD	0.700	0.703(b)	0.703(b)	0.568(w)
90m				
	Gaussian	DK-SMC	DK-EM	K6-EM
Res. (m)	95.037	75.937	75.146(b)	107.485(w)
RMSE (m)	130.094	107.255	105.862(b)	140.291(w)
NFT	1.852	1.871(w)	1.868	0.831(b)
TPD	0.689	0.690(b)	0.690(b)	0.547(w)
90m (corrected)				
	Gaussian	DK-SMC	DK-EM	K6-EM
Res. (m)	71.507	56.742	55.213(b)	96.140(w)
RMSE (m)	102.471	83.051	80.821(b)	135.656(w)
NFT	1.820(w)	1.819	1.809	0.837(b)
TPD	0.709	0.710(b)	0.709	0.576(w)

TABLE 2.1: Summary of performance evaluation results for all DEMs and measurement models.

how each of the trackers handle true detections that occur with a multimodal distribution. In these cases, the GMM trackers initialize and propagate hypotheses based on the false mode(s) as well as the true mode, while the Gaussian tracker does so based on a Gaussian approximation of the distribution. The modal ambiguity in the trackers using GMM measurements is resolved using MHT. However, for some instances, a track based on the false mode is more likely to occur, hence it will appear in the global hypothesis instead of the track based on the true mode. In some cases, this results in the track error being large enough that it is not associated with the truth, resulting in a false track. Comparatively, the Gaussian measurement model may associate with the existing track and pull it off course, and still be associated with the truth, not producing a false track. In this way the performance penalty of misleading multimodal measurements is shifted to the NFT metric for GMM trackers, while it is shifted to the track error metrics for the Gaussian tracker. With the corrected DEMs the GMM-DK trackers show a slight improvement in terms of NFT, suggesting that the problem is primarily a result of inaccuracy in terrain elevation, which is assumed to be accurate by the proposed methods.

As expected, correcting the DEMs resulted in improved performance for all trackers across all metrics. This suggests a direction for future work in developing a method that alleviates the assumption of accurate terrain data to produce better performance results.

Since a corrected DEM is not generally available, it would be desirable to obtain such results using the uncorrected DEM. One possible approach could be implemented during the sampling phase. Instead of sampling from just the measurement distribution, the distribution of the terrain data could also be sampled. Care would need to be taken to design such an approach in a way that remains computationally feasible while obtaining reasonable coverage of the sample space.

Interestingly the performance for all trackers across all metrics was actually better for the DEMs with 90m resolution than the ones with 30m resolution, but this may not be the case under all tracking conditions. In the simulated scenarios, most samples from the measured distribution that are not near the target (i.e. belonging to the false mode for a multimodal distribution) end up on a peak located between the sensor and the targets' true position (see figure 2.14 for an example of this). The 90m DEMs can reduce the prevalence of peaks (both high and low) when compared with the 30m DEMs due to the smoothing of the area between posts that occurs, hence an improvement in performance. If the targets were located on the peak, the 90m DEMs could potentially reduce performance through the same smoothing effect compared with the 30m DEMs. This underscores the situational nature of the task of choosing an optimal tracking approach.

Figure 2.19 shows the residual error over time, while figure 2.20 shows the number of false tracks over time. The residual error for each time step in the simulation is computed for all targets and Monte Carlo runs. The residual error was restricted to samples that involved the target being tracked.

At the start of the simulation period, the targets are both traveling through consistent terrain with a sufficiently high grazing angle, resulting in a measurement distribution that is unimodal. The GMM trackers with dynamically selected number of components (GMM-DK-SMC, GMM-DK-EM) are equivalent to the Gaussian tracker since they have just one component most of the time. This equivalence can be seen in their performance results during this time period, in terms of residual error (figure 2.19) and NFT (figure 2.20). The GMM tracker with six components (GMM-K6-EM) results in a significant overfit of the measurement distribution during this period, and the resulting performance degradation can be seen in terms of track residual error.

At $t = 100$, the tracking conditions become more difficult as the targets enter peaky terrain and are observed from a lower grazing angle, resulting in a multimodal measurement distribution most of the time. During this period, the performance of the Gaussian tracker diverges from that of the GMM trackers with dynamic selection of the number of mixture components, which is now usually selected to be greater than one. It can be seen that these trackers are able to maintain a more accurate track for a longer period of time as the tracking conditions worsen when compared to the baseline Gaussian tracker, while NFT remains largely similar. At this time, the GMM-K6-EM tracker is less of an overfit than it was when the measurement distribution was unimodal. As such the accuracy degradation of this tracker when compared with the GMM trackers that dynamically select the number of components is not as significant, but is still notable.

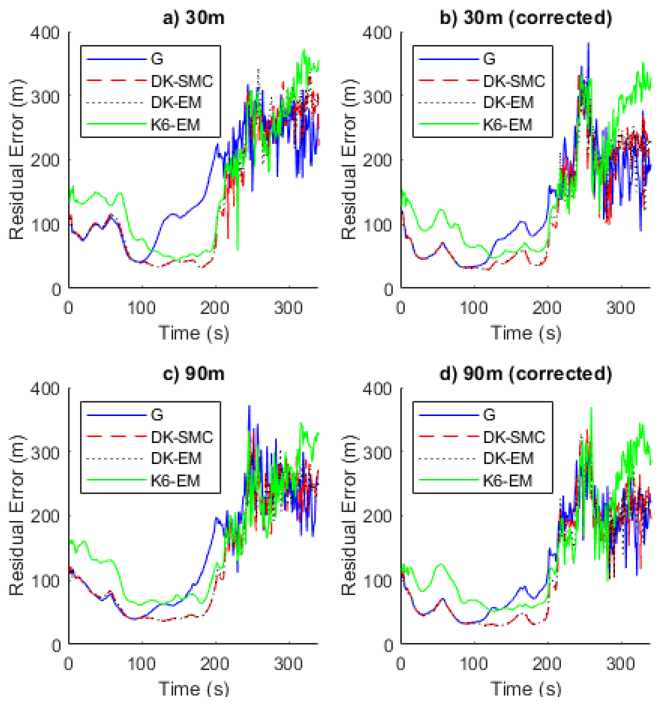

FIGURE 2.19: Residual error (meters) over time for the 4 tracking DEMs with varying resolution and accuracy

Eventually (at $t = 200$) the tracking conditions become very difficult as the targets remain in difficult terrain and also become occluded by terrain for a significant portion of the time. During this period, all trackers struggle to maintain a consistent and accurate track. The differences in NFT between trackers and between the different DEMs is most interesting. Consider the period between $t = 205$ and $t = 220$, where the GMM-DK-EM tracker maintains a lower number of false alarms than the other trackers. If the corrected DEMs are used, this tracker can reduce NFT.

Another interesting observation can be made between $t = 260$ and $t = 300$ when looking at the 30m DEMs. The uncorrected 30m DEM seems to be quite misleading, resulting in a notably larger number of false tracks for all trackers when compared with the corrected 30m DEM. The increase in false tracks is more prevalent for the GMM trackers with dynamic selection of the number of components, which may be overconfident in the tracks formed due to their greater leveraging on misleading terrain data near the targets' positions. Future work aimed at determining a terrain-aided measurement distribution that does not assume the terrain data to be correct may help to reduce false tracks caused by terrain elevation errors.

2.5.5 Computation Time

One of the objectives for the proposed algorithm, outlined in section 2.1, is that it should be usable in real-time. To determine if this has been achieved, each algorithm

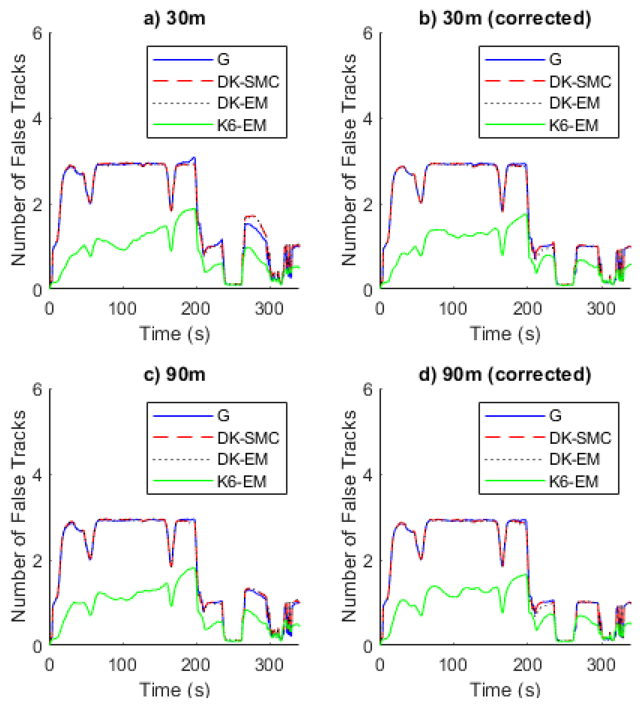

FIGURE 2.20: Number of false tracks over time for the four trackers and DEMs with varying resolution and accuracy

was timed during 200 Monte Carlo runs using both 30m and 90m resolution DEMs. The timed computations included the parameterization of the terrain-aided measurement distributions for all true detections and false alarms, as well as tracking using MHT to resolve both association ambiguity and ambiguity among the mixture components (where applicable). Therefor the whole online tracking process except for the detection stage (which is assumed to have already been performed and is the same for all trackers) has been timed. All timing was carried out using an Intel Core i7-6700HQ CPU running on a single thread. The average computation time (wall time) for each tracker for a simulation of 340 frames as well as the corresponding computation rate in frames per second (FPS) are given in table 2.2.

30m				
	Gaussian	DK-SMC	DK-EM	K6-EM
Time (s)	24.546	28.018	27.734	25.682
FPS	13.943	12.210	12.338	13.354
90m				
	Gaussian	DK-SMC	DK-EM	K6-EM
Time (s)	15.613	17.628	16.986	16.611
FPS	23.779	20.830	21.640	22.187

TABLE 2.2: Summary of computation times (wall time) and rates for each DEM resolution and tracker.

It is expected that the GMM trackers will have longer computation time when compared with the Gaussian tracker for two reasons. Firstly, computing the parameters for the Gaussian measurement distribution is relatively straightforward, requiring only the calculation of the sample mean and sample covariance, unlike the Gaussian mixture measurements. Secondly, the Gaussian will always result in only one additional hypothesis per measurement to consider associating with each existing track, while each Gaussian mixture measurement may result in more than one association to consider. As expected, the Gaussian tracker had the lowest computation time in simulations.

For the computation of the distribution parameters, both the GMM-DK-SMC and GMM-DK-EM trackers employ the proposed image processing methods to cluster samples and estimate the number of mixture components, while the GMM-K6-EM tracker skips this step entirely. The GMM-DK-EM and GMM-K6-EM trackers use EM to compute the final mixture parameters, so it is expected that GMM-K6-EM will have an overall lower computation time for the distribution parameters. The GMM-DK-SMC tracker uses the cluster sample mean and sample covariance when computing the distribution parameters, which is similar to the Gaussian distribution for each cluster and depends on the number of samples used to estimate the distribution. Therefor the computation time for this step is expected to be similar to the same step of the Gaussian distribution (not including the image processing previously mentioned).

The main factor for MHT tracking computation time is the number of hypotheses being maintained. New measurements using the GMM trackers will generate one new hypothesis for each mixture component to consider for each existing hypothesis. The GMM-DK-SMC and GMM-DK-EM trackers will have one or more mixture components, however in simulation the number of mixture components rarely exceeded two, and was frequently one.

The GMM-K6-EM tracker always uses six mixture components. Initially, it is expected that the GMM-K6-EM tracker would take the longest time during the tracking phase due to the greater number of hypotheses corresponding to each mixture component, but it is also important to consider that MHT trackers use pruning to reduce the number of hypotheses by removing those that have a low track score. Recall that in section 2.5.4 it was noted that the GMM-K6-EM tracker formed fewer tracks overall than all other trackers, despite the larger number of mixture components. The lower computation time for the GMM-K6-EM tracker suggests that many of the hypotheses generated by this tracker are quickly deemed unlikely and are pruned from the hypothesis tree.

Examining the measured total computation time for each of the GMM trackers, the results are largely as expected. The GMM-K6-EM tracker skips the image processing step and forms fewer tracks, and has a lower computation time than the other GMM trackers. The GMM-DK-EM has a lower computational time when compared with the GMM-DK-SMC despite the fact that EM is an iterative method that depends on the number of samples as well as the number of iterations. However the cluster sample mean and sample covariance implementation was created for the purposes of these simulations, while the EM implementation is an open source implementation that is highly optimized (described in detail in [42]). In light of this, such a result is not entirely unexpected.

Lastly, increasing the resolution of the DEM (30m post spacing compared with 90m) is expected to result in increased computation time because more cells of the DEM need to be traversed before reaching the point of intersection during sampling (see section 2.2.2). This expectation is consistent with the computation times observed with different DEMs. The differences in performance between trackers can be observed on both DEMs. If DEM resolution is very high, computational cost may become prohibitive.

2.6 Conclusion

This chapter considers the problem of terrain-aided tracking of ground targets from an aerial platform using a sensor that provides only angular measurements of the position of a target (such as a camera). Previous works assume the measurement distribution to be Gaussian, which fits conveniently into most existing tracking frameworks. However, the actual measurement distribution may be multimodal under some operational conditions. To address this mismatch, this work modeled the measurement distribution as a Gaussian mixture distribution. A framework was proposed for terrain-aided tracking of multiple targets using Multiple Hypothesis Tracking (MHT) to resolve ambiguities both in measurement-to-track association as well in which mode of the multimodal measurement distribution best represents the location of the target.

Three different methods for computing the parameters of a Gaussian mixture measurement models were considered. Two of the methods used the proposed clustering method to estimate the number of mixture components, followed by computation of the final distribution parameters using either cluster sample mean and sample covariance (in GMM-DK-SMC) or EM (in GMM-DK-EM). The final method (GMM-K6-EM) assumed a distribution with six components and used EM to parameterize the distribution. These methods, including a baseline Gaussian measurement distribution, were tested using four different tracking DEMs with varying resolution and accuracy. Performance was evaluated in terms of track accuracy (residual error and RMSE), track probability of detection (TPD), and number of false tracks (NFT). GMM-DK-SMC and GMM-DK-EM performed similarly across all metrics in all scenarios, GMM-DK-EM achieved the best residual error and RMSE among all trackers evaluated. The GMM-DK-SMC and GMM-DK-EM outperformed the Gaussian baseline in all evaluated categories and DEMs used except for the NFT when using uncorrected DEMs.

The proposed methods offer improvement in track accuracy that is vital for use in realistic tracking applications. Additionally, the improved performance achieved across all metrics when using corrected DEMs suggests a direction for future work. The sampling method could be adjusted to account for uncertainty in terrain data, may yield better performance when applied to trackers.

Chapter 3

Sensor Bias Estimation and Compensation for Practical Multisensor-Multitarget Tracking

3.1 Introduction

In multisensor-multitarget tracking, the objective is to estimate the state (position, velocity, etc.) of targets using information obtained from multiple sensors. These sensors are subject to two types of measurement errors which are zero-mean noise and bias errors. The zero-mean noise error can be removed effectively through filtering (for example, using a Kalman Filter) [43]. The bias error which is either constant or slowly-varying over time, not zero-mean, can not always be removed through standard filtering processes [44]. Bias error can negatively affect estimation performance in any tracking problem and it is particularly problematic in multisensor-multitarget tracking where association of measurements or tracks from different sensors is required. This performance degradation 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 found in the existing literature, some are optimal under certain conditions, but most address only a few of the realistic challenges faced by many practical systems. Furthermore, several practical methods are only applicable to certain specific scenarios, requiring particular sensor and bias models, number of sensors, fusion architecture, substantial computational resources, synchronous communication, and/or high communication rate. These make applications to large heterogeneous sensor networks difficult and error-prone.

Bias estimation and compensation methods can be broadly divided into three categories. The first category uses an augmented state vector to simultaneously estimate the state of all targets and sensor biases. The second category formulates a pseudomeasurement of the sensor biases that is independent of the target state. The third category employs particle methods to estimate various states including sensor biases. Each of these categories is reviewed below highlighting 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 [45, 46, 47]. 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 [45, 47]. However, if all these state variables are stacked, the result can be a very large state vector when 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), a large state vector can become computationally prohibitive for use in a real time system [45]. 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, thereby reducing the computational load. In [45, 48] approximations are used to decouple the augmented state, but are subject to performance degradation. In [47] 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, becoming computationally expensive for large sensor networks. A two-stage approach was proposed in [46] to decouple the target states from the sensor biases when the biases are constant, which was extended by [49] to accommodate dynamic biases with white or coloured Gaussian noises. However, [49] also points 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 [45, 47, 49]. Such algorithms are therefor only applicable to time-synchronized centralized fusion networks. Extension to nonlinear systems is possible through linearization using the Extended Kalman Filter (EKF), but these require situational derivations and are still sub-optimal [44]. For highly nonlinear systems it may be desirable to use a robust tool such as the Unscented Transform, which ASKF methods require substantial extension to accommodate.

Another popular approach is to formulate a pseudomeasurement of the sensor biases independent of the target state by taking the difference between measurements or tracks from different sensors such that the target state cancels out [50, 51, 52, 53], leaving a noisy measurement of the sensor biases that can be used to estimate the biases directly. The pseudomeasurement strategy has been successfully employed in various fusion architectures [52, 51, 53, 50] and with asynchronous data rates [50, 54]. In most cases the state vector for bias estimation is the stacked vector of the biases from the sensors used to formulate the pseudomeasurement. Hence it is much more computationally efficient than the ASKF which also stacks all the target states. This state vector grows with the number of sensors, so for large sensor networks it becomes computationally expensive. Beyond computational load, many of these methods are limited, in that they work with a pair of sensors [52, 51, 54], hence using them with a large sensor network is not possible without the additional consideration of pairing sensors. Some methods, such as in [50], 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 which often lead to different solution methodologies. For example in [50], 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 [51], a similar starting point is taken but applied to a pair of video sensors. The resulting solution requires minimizing a nonlinear cost function using the genetic algorithm. However in [52], an alternative method is proposed that applies approximations allowing RLS to be used. In all, highly nonlinear systems remain a challenge for pseudomeasurement approach in a similar way as with the AKSF methods. There are solutions that can be applied using a converted measurement model [50, 51, 52], but requires situational modelling of the bias in the converted measurement model.

The final category of methods that have been applied to bias estimation use particle methods such as the particle filter and 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 they also have drawbacks. The computational load can be substantial, especially if the state space dimension is large, since so many particles must be used to achieve adequate coverage. The required number of particles grows exponentially in the number of dimensions. Several 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 [55]. However if the number of dimensions that are required in the state remains large (for example if there are many bias parameters), the problem persists. Particle filters are also known to suffer from the problems of degeneracy and sample impoverishment [56]. 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 particles that are not relevant to the distribution estimate [56]. Degeneracy can be partly solved by resampling, where samples are redrawn to have equal weights but still represent approximately the same distribution [56]. However resampling introduces a tuning parameter to set a degeneracy threshold when resampling should occur, and can also result in worsened sample impoverishment [56]. 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. The impoverishment problem is especially prominent when process noise is low, or zero as in the case of constant biases [56]. Approaches have been proposed to counteract sample impoverishment by adding some artificial process noise to static parameters [57, 58] but this solution introduces at least one additional tuning parameter.

Another important aspect of the sensor estimation problem is bias compensation. Many existing works assume the bias estimate to be sufficiently accurate to correct the bias, and treat the problem of bias compensation by adjusting the measured values, subtracting the mean of the estimated bias [52]. In [59], 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, suggesting that it is not always reasonable to ignore the uncertainty of the bias estimate during bias compensation. The method from [59] requires explicit modeling of the cross-covariance terms in the update step and simplifies them with some approximations such as 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.

Given the limitations of existing methods discussed above, there is a need for a unified, flexible bias estimation and measurement compensation 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 chapter, a novel algorithm for sensor bias estimation is presented to address this need. This is the main contribution of this chapter. The secondary contribution of this chapter is a proposed approach for fusion under unknown correlation of unequal state vector estimates, used in the proposed algorithm. The proposed approach is less conservative than Covariance Intersection and more conservative than the Kalman Filter. Section 3.2 provides background on existing algorithms that are used within the proposed algorithm. A mathematical formulation of the general problem is given in section 3.3 that introduces the relevant notations and concepts. Section 3.4 describes the proposed algorithm for bias estimation and measurement compensation in detail, along with an analysis of computational complexity. An example application with required implementation details, corresponding simulation results, and discussions are presented in section 3.5. A conclusion summarizes the findings of this work in section 3.6.

3.2 Background

3.2.1 Multisensor-Multitarget Tracking

In multisensor-multitarget tracking problems, a number of sensors are used to track multiple targets and 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 3.1. 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 of measurements (not auto-correlated over time) [44]. The function h maps the true measured state and bias parameters to the measurement space.

$$z(k) = h(x(k), \beta(k)) + v(k) \tag{3.1}$$

Each measurement is 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 that shows how the target state evolves over time, referred to as a state-transition model [44]. In tracking applications, the state-transition model is often given in the form of a kinematic model. A general state-transition model is given in equation 3.2 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 the 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 of time [44]. Some formulations of the state-transition model also include modeling of a known control input which cannot be assumed to be known for non-cooperative tracking.

$$x(k+1) = f(x(k), k+1) + w(k+1) \tag{3.2}$$

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) [44]. 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 nonetheless it remains the most popular choice as the basis of many tracking systems. Extensions to the KF, such as the Extended Kalman Filter (EKF) [44] and Unscented Kalman Filter (UKF) [60], 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. Compatibility is often achieved through linearization of a nonlinear measurement function (for EKF) or through a Gaussian approximation of a random variable (for UKF).

When using multiple sensors, it becomes necessary to combine (or fuse) their information. There are three main types of fusion architecture (centralized, distributed, and decentralized) that define how information is combined from different sources. 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) [61]. 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 [61]. To address the challenges of centralized fusion, 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 [61]. 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 [61].

In decentralized fusion architecture, there are multiple fusion centers (FCs) that can communicate with only a subset of sensor nodes or other FCs [61]. A FC in a distributed architecture can receive information in the form of measurements from sensor nodes or tracks from other FCs [61]. Hierarchical architectures can also be used that perform fusion at various levels, effectively combining different architecture types [61]. 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 therefore what bias estimation algorithms can be applied.

3.2.2 The Unscented Transform

First described in [60], 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 [6, 62, 63]. The UT will be used in the proposed algorithm for bias estimation.

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 [60], 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
 1: $\mathcal{X}_0 = \bar{x}$
 2: $W_0 = \kappa/(n + \kappa)$
 3: for each $i \in 1 \ldots n$ do
 4:     $\mathcal{X}_i = \bar{x} + (\sqrt{(n + \kappa)P_{xx}})_i$
 5:     $W_i = 1/(2(n + \kappa))$
 6:     $\mathcal{X}_{i+n} = \bar{x} - (\sqrt{(n + \kappa)P_{xx}})_i$
 7:     $W_{i+n} = 1/(2(n + \kappa))$
 8: end for
    // Transform sigma points
 9: for each $i \in 0 \ldots 2n$ do
10:     $\mathcal{Y}_i = f(\mathcal{X}_i)$
11: end for
    // Compute parameters of transformed distribution
12: $\bar{y} = \sum_{i=0}^{2n} W_i \mathcal{Y}_i$
13: $P_{yy} = \sum_{i=0}^{2n} W_i (\mathcal{Y}_i - \bar{y})(\mathcal{Y}_i - \bar{y})^T$
14: return $\hat{y} = \mathcal{N}(\bar{y}, P_{yy})$
```

The UT simplifies the nonlinear transformation of the random variable, as all that is needed is to evaluate the nonlinear function f at each of the sigma points.

3.2.3 Information Fusion Under Unknown Cross-Correlation

The Covariance Intersection (CI) algorithm was proposed in [64] 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 [65, 66, 67]. The fused CI estimate is given by equation 3.3 [64], where estimates a, b with means $\bar{a}, \bar{b}$ and covariance matrices P_{aa}, P_{bb} are fused.

$$\begin{aligned} P_{cc} &= (\omega P_{aa}^{-1} + (1 - \omega) P_{bb}^{-1})^{-1} \\ \bar{c} &= P_{cc}(\omega P_{aa}^{-1} \bar{a} + (1 - \omega) P_{bb}^{-1} \bar{b}) \end{aligned} \tag{3.3}$$

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 [64]. The notion of consistency here is that the estimated mean squared error (MSE) matrix P_{cc} does not underestimate the true MSE $\bar{P}_{cc}$, as expressed in equation 3.4 from [64].

$$P_{cc} - \bar{P}_{cc} \geq 0 \tag{3.4}$$

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 the cost [64, 68, 69]. The CI estimate is the optimal (in the sense of the chosen cost function) consistent estimate if the cross-correlation is entirely unknown [68]. 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) [70]. The CI algorithm is summarized in algorithm 2.

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

```
   // Find the parameter $\omega$ to minimize cost function $J$
1: $\omega = \arg\min_\omega J$, subject to $0 \leq \omega \leq 1$
   // Compute CI estimate
2: $\bar{c}, P_{cc} = \langle$Use equation 3.3 or 3.5$\rangle$
3: 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 was discussed in [70], though it was noted that efficient optimization to find the optimal weight ω can be more challenging. Furthermore, the proof of consistency is explicitly given, [64, 70], for CI with equal state dimensions (as in equation 3.3). As pointed out in [71], 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 is worth considering. The fused CI estimate for unequal state dimensions is given by equation 3.5 [70], 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.

$$\begin{aligned}\bar{c} &= \bar{a}+\\ &\quad (1-\omega)P_{aa}H^T((1-\omega)HP_{aa}H^T+\omega P_{bb})^{-1}(\bar{b}-H\bar{a})\\ P_{cc} &= \frac{1}{\omega}(P_{aa}-\\ &\quad (1-\omega)P_{aa}H^T((1-\omega)HP_{aa}H^T+\omega P_{bb})^{-1}HP_{aa})\end{aligned} \tag{3.5}$$

In [72], several alternative approaches were discussed for track-to-track fusion with unequal state dimensions using CI. Their work presented and compared 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 3.3. They also compared 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 [73], with an application to bias estimation. There the authors used a decentralized information filter (rather than CI) to update the state during fusion by augmenting the smaller state with zero information about the missing components. In

[74], a slightly different approach was used, performing CI to fuse the common states only. They assumed 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 [72, 74, 73] 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, formulated in terms of a bounding covariance matrix, was presented in [75]. This was generalized in [76] 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 [77], the covariance bound (CB) approach [75, 76] was 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 was used to formulate a weighted least squares (WLS) problem that could be solved to yield a fused state estimate of all state components. A similar formulation was also given in [78]. The conservative bounding joint covariance matrix from [77] is given in equation 3.6. The matrix was given for fusing an arbitrary number of estimates where each estimate's covariance block was multiplied by the inverse of its weight and the sum of the weights is one [77]. Here it is presented in a simplified form for use with just two estimates so that there can be only one weight parameter ω.

$$\begin{aligned} \bar{x}_{CB} &= \begin{bmatrix} \bar{a}^T & \bar{b}^T \end{bmatrix}^T \\ P_{CB} &= \begin{bmatrix} \frac{1}{\omega} P_{aa} & 0 \\ 0 & \frac{1}{1-\omega} P_{bb} \end{bmatrix} \end{aligned} \tag{3.6}$$

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 3.7 [77] where H is used to map the components of each of the state estimates to the joint state consisting of all components.

$$\begin{aligned} K &= (H^T P_{CB}^{-1} H)^{-1} H^T P_{CB}^{-1} \\ \bar{c} &= K x_{CB} \\ P_{cc} &= K P_{CB} K^T \end{aligned} \tag{3.7}$$

The authors in [77] did not provide details on how to compute the weighting parameters for the covariance matrices, as noted in [71] where an alternative approach was proposed. Their approach was based on an inner ellipsoidal approximation and provided details on how to compute the required weight matrices by solving two semidefinite programming (SDP) problems [71]. Their results showed improved performance when compared with the CB method from [77] with equal weighting of the estimates forming the bounding joint covariance matrix, however a better weight selection strategy for the CB approach can be achieved. Also, the approach in [71] does not guarantee consistency.

Other approaches for fusion under unknown correlation that leverage on a specific problem structure to achieve promising results are available. One such approach that has been used successfully in track-to-track fusion is sample-based reconstruction of cross-covariances. In [79, 80], such an approach was applied with homogeneous state vectors, and in [81, 82] it was generalized to heterogeneous states. The sample-based reconstruction and the optimal fusion where cross-covariances are maintained rigorously had similar performance [79, 80, 81, 82]. The drawback of these approaches is that they require a specific correlation structure which limits their scope of application or requires reformulation to apply to new problems.

3.2.4 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, leaving 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 [26, 6]. The general line of sight approach is illustrated in figure 3.1.

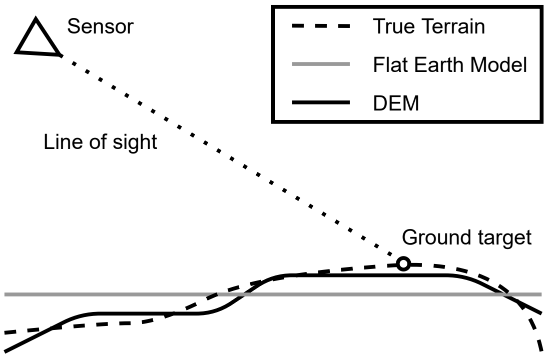

FIGURE 3.1: An illustration of the line of sight method for determining the position of a target in Cartesian coordinates.

When terrain is relatively consistent, a flat earth model or a curved earth model (such as WGS-84) can be appropriate which allows for convenient geometric conversions of the measurement to Cartesian coordinates [26]. These are commonly used in bias estimation and allow for simplification of the bias estimation problem [50, 52]. If the terrain is varied around the tracking area and can be accepted as locally flat, then a basic geometric measurement conversion can be applied if the elevation in the area of the target is known [52].

If terrain is further varied, it becomes imperative to use a digital elevation model (DEM) to model the terrain's features accurately. In this case, the measurement conversion is complex 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 [5, 6, 2]. In [5], the UT is used to transform the measurement by transforming each sigma point using a line of sight intersection algorithm. A similar approach, using a sampling transform [2] where the measurement distribution in Cartesian coordinates is assumed to be a Gaussian mixture. The transformations in all of these approaches involve intersecting individual lines of sight from the sensor to the target with the terrain model (see figure 3.2) and 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. Detailed examples are in [5, 6, 2].

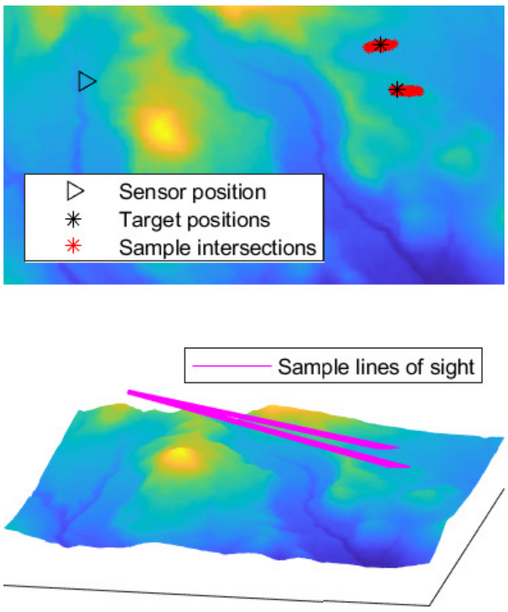

Figure 3.2: Illustration of terrain-aided tracking using line of sight in DEM.

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 leads to a line of the form given in equation 3.8, 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.

$$L = X + tD \tag{3.8}$$

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 3.9 [2].

$$\begin{aligned} \min_{t} \quad & t && (3.9a)\\ \text{subject to} \quad & X + tD = X_w, && (3.9b)\\ & t > 0 && (3.9c) \end{aligned}$$

The solution t is found by traversing the DEM grid until the nearest point of intersection is reached [2]. The grid tracing approach from [29] is used, allowing only the cells of the DEM in the path of the ray to be traversed, as shown in figure 3.3. As the cells are traversed in order of increasing distance from the sensor, each is tested for intersection 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 [30]. When a point of intersection is found, it will be the one nearest to the sensor (see [30]) and the algorithm can be stopped.

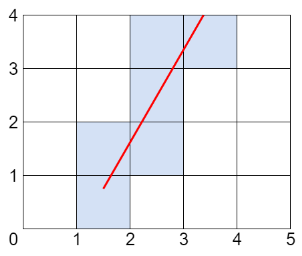

FIGURE 3.3: DEM grid traversal along a line of sight.

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 are not always available for the tracking area of interest. The NASA SRTM data sets provide DEMs with resolutions of 30m [11] and 90m [12] 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.

3.3 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 which can all be modeled in the form given in equation 3.1. However, the notation in equation 3.10 is more specific to the problem.

$$z^m_{s,t}(k) = h^m_s(x^m_s(k), x^m_t(k), \beta_s(k), v_s(k)) \tag{3.10}$$

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 the measurement 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 bias model 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 3.11, 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 based on the state-transition model of the form given by equation 3.2. The measurement conversion function may also depend on some additional global information z_g such as road map information or terrain data (see section 3.5 for an example). The global information can be considered to be constant (if known completely) or a random variable (if there is some uncertainty).

$$z^c_{s,t}(k) = h^c_s(z^m_s(k), \hat{\beta}_s(k), z_g) \tag{3.11}$$

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 3.2.

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). Information may be received in an asynchronous manner. Much of the discussion in sections 3.4 and 3.5 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.

The data association (measurement-to-track at the local node, and track-to-track when performing fusion) is assumed to be known, as the association problem is beyond the scope 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.

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, similar to the one-point KF initialization method from [44]. The objective is to update and refine these bias estimates, beginning with the prior, according to information obtained from multiple sensors. For an example of this type of problem formulation, see section 3.5.

3.4 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 3.3, without additional assumptions that narrows the scope of application. The novelty of the proposed algorithm is that it can be applied generically without significant restrictions in terms of the measurement model, bias model, or fusion architecture. The proposed algorithm can handle operational scenarios that restrict the use of other computationally efficient methods.

3.4.1 Overview of the Proposed Algorithm

The strategy in the proposed algorithm can be divided into two main tasks, contained in the shaded boxes in figure 3.4. This figure shows how an update is performed for a single target and a single sensor. The first task, shaded box on the right, is to update the bias estimate. The bias estimate update 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. The second task, shaded box on the left, is to update 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.

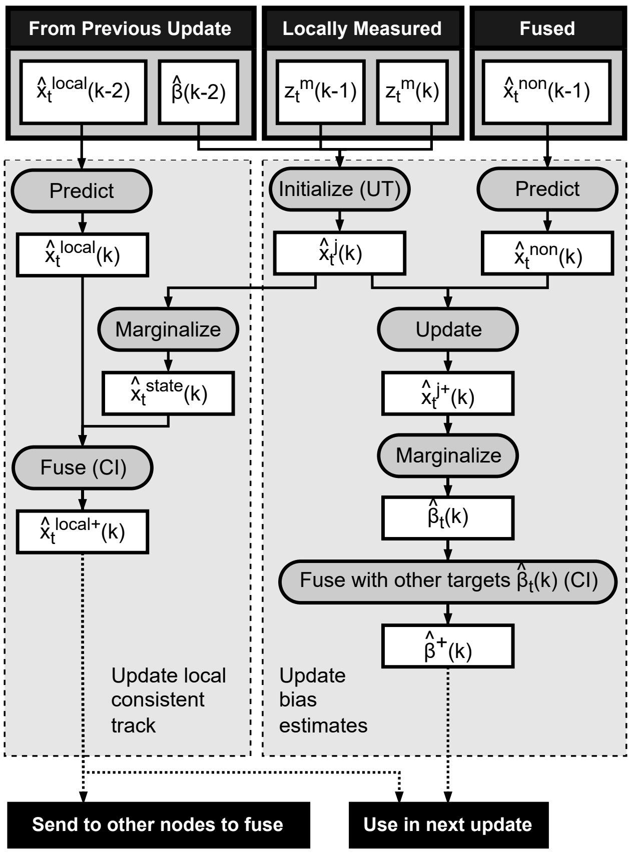

FIGURE 3.4: A flowchart of the proposed algorithm showing the data flow for a sensor node update using a single target identified by t.

The consistent local tracks are updated using CI to retain their consistency despite correlations introduced by bias compensation, without modeling these correlations explicitly. The consistent local tracks are sent to other sensor nodes. These nodes fuse the local tracks together (excluding its own track) to form a consistent non-local track which summarizes the information about the target gotten from other sensors. When a tracking estimate output is required at the local sensor, the consistent local and non-local tracks are fused to provide it. CI is used 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 update for the bias estimate requires forming a joint estimate of the target state and the sensor bias which is accomplished for the instantaneous time k using the UT and the two previous local measurements. The process is repeated at every update to capture the current cross-correlations between the bias parameters and the target state directly at the instant of the update, and is required because the cross-correlations depend not only on the target state (which is tracked) but also on the sensor position (which is not tracked, but measured). The joint estimate is then 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). Since these correlations are not explicitly modeled and the non-local track does not include the bias components in the state vector, information fusion under unknown cross-correlation with unequal state dimension is required.

Another option for updating the bias estimates is to ignore the cross-correlations and use the standard KF update. Ignoring the cross-correlations can result in inconsistent bias estimates, but the cross-correlations are small (in some sense) if the (true) bias uncertainty is small relative to the measurement uncertainty and the only source of correlations is the bias compensation process. An additional alternative for the bias estimate update is proposed in section 3.4.3.

3.4.2 Unscented Transform for Joint Estimate Initialization

As described in section 3.2.2, the UT can be used to form a Gaussian estimate of a non-linear transformed random variable. The UT is also used to initialize a joint estimate of the stacked target state and bias estimate as well as to initialize target states using various motion models. To form the Gaussian estimate, a stacked random variable consisting of the most recent two sensor measurements, the previous bias estimate and any additional global information is used as in equation 3.12. With a two-point initialization scenario, 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 two measurements from the local sensor). The joint distribution parameterized by $\bar{x}^{m,j}_{s,t}, P^{m,j}_{s,t}$ will be the input distribution from which the sigma points are drawn for the UT.

$$\begin{aligned} \bar{x}^{m,j}_{s,t} &= \begin{bmatrix} \bar{z}^m_{s,t}(k)^T & \bar{z}^m_{s,t}(k-1)^T & \bar{\beta}^T_s(k-2) & \bar{z}^T_g \end{bmatrix}^T \\ P^{m,j}_{s,t} &= \begin{bmatrix} R^m_s(k) & 0 & 0 & 0 \\ 0 & R^m_s(k-1) & 0 & 0 \\ 0 & 0 & P_\beta(k-2) & 0 \\ 0 & 0 & 0 & P_g \end{bmatrix} \end{aligned} \tag{3.12}$$

The transformation function f, to transform each of the sigma points and initialize the track state in a joint distribution with the bias, is derived using equation 3.13.

$$\begin{aligned} f(\mathcal{X}_i) &= \begin{bmatrix} \mathcal{P}_i(k) \\ \mathcal{V}_i(k) \\ \mathcal{B}_i \end{bmatrix} \\ \mathcal{P}_i(k) &= h^c_s(\mathcal{Z}^m_i(k), \mathcal{B}_i, \mathcal{Z}^g_i) \\ \mathcal{V}_i(k) &= \frac{\mathcal{P}_i(k) - \mathcal{P}_i(k-1)}{T(k,k-1)} \end{aligned} \tag{3.13}$$

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 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. The bias estimate precedes any measurement used in the UT and is uncorrelated to any of the measurement. For a CV model, the bias estimate can be used for every other measurement. For a constant acceleration (CA) model, the bias estimate is used every third measurement. The time between the measurements is denoted $T(k, k-1)$.

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. Algorithm 1 shows details about drawing sigma points and the rest of the UT algorithm. For higher-order kinematic models, more measurements can be used.

3.4.3 Covariance Intersection Pseudomeaurement Update

The proposed algorithm involves an update step that requires information fusion under unknown cross-correlation with unequal state dimensions, as reviewed in section 3.2.3. An alternative, using the KF update with a pseudomeasurement derived according to the CI algorithm, is proposed. The result is the same estimate in common state components as using the standard CI algorithm (with equal state dimensions on 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 3.14, with subscripts c and u respectively. The same block decomposition applies to $\hat{x}^+$ with the same notation.

$$\begin{aligned} \hat{x} &= \mathcal{N}(\bar{x}, P_x) \\ \bar{x} &= \begin{bmatrix} \bar{x}_c & \bar{x}_u \end{bmatrix}^T \\ P_x &= \begin{bmatrix} P_{cc} & P_{cu} \\ P_{uc} & P_{uu} \end{bmatrix} \end{aligned} \tag{3.14}$$

The CI estimate of the fused common state components $\hat{x}_{ci}$ is computed using the standard CI algorithm, as shown in equation 3.15. It is set to be equal to the posterior update of the common state components.

$$\begin{aligned}\hat{x}_{ci} &= CI(\hat{x}_c, \hat{x}', trace)\\ &= \mathcal{N}(\bar{x}_{ci}, P_{ci})\\ &= \hat{x}_{cc}^{+}\\ &= \mathcal{N}(\bar{x}_{cc}^{+}, P_{cc}^{+})\end{aligned} \tag{3.15}$$

Starting from the above equivalence and the Kalman Filter measurement update equations, a pseudomeasurement z_{eq} with mean $\bar{z}_{eq}$ and covariance R_{eq} is derived. The pseudomeasurement, when used in the standard KF update, results in the same updated (fused) target state estimate as the CI algorithm (the updated estimate of the common state is consistent if the input estimates are consistent). The derivation begins in equation 3.16 below, where the H matrix is a selector of the common state components taking the form $H = [I0]$.

$$\begin{aligned}P^{+} &= (I - PH^T(HPH^T + R_{eq})^{-1}H)P\\ &= (I - PH^T(P_{cc} + R_{eq})^{-1}H)P\\ &= (I - P\begin{bmatrix}(P_{cc} + R_{eq})^{-1}\\ 0_{mat}\end{bmatrix}H)P\\ &= (I - \begin{bmatrix}P_{cc}(P_{cc} + R_{eq})^{-1} & 0\\ P_{uc}(P_{cc} + R_{eq})^{-1} & 0\end{bmatrix})P\end{aligned} \tag{3.16}$$

Only the top-left block of the matrix in equation 3.16, pertaining to the common states, is required to solve for R_{eq}. Considering the top-left block only after right-multiplying P yields equation 3.17.

$$P_{ci} = P_{cc} - P_{cc}(P_{cc} + R_{eq})^{-1}P_{cc} \tag{3.17}$$

Rearranging equation 3.17 to solve for R_{eq}, the final expression for the covariance of the pseudomeasurement is given in equation 3.18. Note that the covariance of the estimate of the common state components must be invertible, as must be $(P_{cc}^{-1} - P_{cc}^{-1}P_{ci}P_{cc}^{-1})$.

$$R_{eq} = (P_{cc}^{-1} - P_{cc}^{-1}P_{ci}P_{cc}^{-1})^{-1} - P_{cc} \tag{3.18}$$

A similar approach is used to derive $\bar{z}_{eq}$ by solving the top block of equation 3.19 corresponding to the common state components. The top block of the vector yields equation 3.20, which is rearranged to solve for $\bar{z}_{eq}$. The final expression for $\bar{z}_{eq}$ is given in equation 3.21.

$$\begin{aligned}\bar{x}^{+} &= \bar{x} + PH^{T}(HPH^{T} + R_{eq})^{-1}(\bar{z}_{eq} - H\bar{x}) \\ &= \bar{x} + \begin{bmatrix} P_{cc}(P_{cc} + R_{eq})^{-1} \\ P_{uc}(P_{cc} + R_{eq})^{-1} \end{bmatrix} (\bar{z}_{eq} - \bar{x}_{c})\end{aligned} \tag{3.19}$$

$$\bar{x}_{ci} = \bar{x}_{c} + P_{cc}(P_{cc} + R_{eq})^{-1}(\bar{z}_{eq} - \bar{x}_{c}) \tag{3.20}$$

$$\bar{z}_{eq} = (I + R_{eq}P_{cc}^{-1})(\bar{x}_{ci} - \bar{x}_{c}) + \bar{x}_{c} \tag{3.21}$$

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. 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.

3.4.4 Bias Compensation

The estimated bias can be compensated at the measurement level. The variance of the bias estimate is also vital for 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 be consistent with the true position of the targets, as illustrated in figure 3.5. The result can either 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, as is the case with the proposed algorithm, and it is assumed to be zero-mean with an appropriately large covariance (consistent with reasonable bias values). Therefore it is important for practical purposes to consider how bias compensation can be achieved such that both the measurements and tracks remain consistent. The following discussion will assume that the bias estimate being used for compensation is consistent with the true bias.

In Fig. 3.5, the bias estimate covariance is not small relative to the measurement covariance. The compensated measurement 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, bias estimate covariance can be considered 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. The effects 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 additional assumptions about the measurement conversion or bias models and only requires a known procedure for mapping points in the joint measurement-bias space to the target position in the state space. Such an approach also lends itself well to the formation of a joint estimate

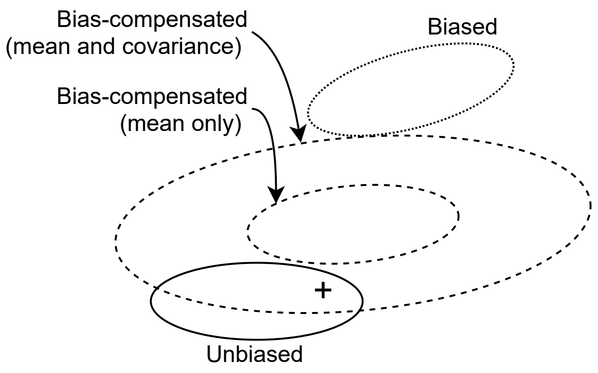

Figure 3.5: An example comparing confidence ellipses of measurements that are unbiased, biased, and bias-compensated (with and without bias-inflation).

including the bias (as is done in section 3.4.2). 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 [59], the correlation of bias-inflated measurements is accounted for by modelling the cross-covariance terms explicitly (with some approximations). If it is not feasible to model the cross-covariance terms explicitly for a given measurement and bias model, the dependence of the bias-inflated measurements is often ignored and the KF is used to track targets. This approach is less optimistic than ignoring the variance of the bias estimate in bias compensation. However, it is still an approximation due to the assumption that the measurements are independent from one another over time. If the bias estimate variance is small, then the approximation may be acceptable since the inflated measurements can be considered independent. But if the bias estimate variance is large (such as at the onset of the bias estimation process), then it is important to consider this correlation so that the tracks remain consistent. The cross-correlation can then be treated as unknown and an approach discussed in section 3.2.3 can be employed for the update (such as CI). It will converge slower than using the KF, but can guarantee consistency. Such an approach is used in the proposed algorithm to maintain the consistent local tracks (as discussed in section 3.4.5 with reference to figure 3.4). With a better knowledge of the bias estimate, the bias estimate variance and the level of correlation between the bias-inflated measurement shrinks. At some point, the performance trade-off between convergence and consistency might favor switching from an assumption of unknown correlation update to independence update.

3.4.5 Proposed Bias Estimation Algorithm

The proposed bias estimate algorithm is described in detail in the form of pseudo-code. The sensor node can either make local observations of targets or receive information about targets (in the form of consistent tracks) from another sensor node. The bias estimation process at a single sensor node s maintains the bias estimate for sensor s, a consistent local track $\hat{x}_t^{local}$ for each target t, and a consistent non-local track $\hat{x}_t^{non}$ (summarizing information received from other sensors) for each target t. The tracks $\hat{x}_t^{non}$ are 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.

A set of variables for all targets t is described in equation 3.22. Note that since these equations refer to a single sensor, the subscript s is dropped from the notation. The exception to this convention 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.

$$\begin{aligned} Z^m(k) &= \{z_t^m(k) | t \in 1 \ldots M(k)\} \\ \hat{X}^{local}(k) &= \{\hat{x}_t^{local}(k) | t \in 1 \ldots M(k)\} \\ \hat{X}^{non}(k) &= \{\hat{x}_t^{non}(k) | t \in 1 \ldots M(k)\} \end{aligned} \tag{3.22}$$

The updates to the bias estimate $\hat{\beta}$ and consistent local track $\hat{x}_t^{local}$ for each sensor node s occur when local measurements are made, as detailed in algorithm 3. The inputs for this algorithm (in addition to those maintained internally) are the measurements Z^m of each target from this time k and the previous measurement time $k-1$, as well as (optionally) some global information z_g such as terrain data. Note that because two measurements are required for the update (with a CV model), the proposed update algorithm should be done on every other measurement update. When two 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 $\hat{x}_t^{non}$ is not yet initialized, the bias update must be skipped for the corresponding target until information from another sensor 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. On line 20 of this algorithm, the update operation is fused under unknown cross-correlation with unequal state dimensions. Several alternatives for the update operation are discussed earlier in section 3.4.5, and are compared through simulations in section 3.5.

The consistent non-local tracks $\hat{x}_t^{non}$ are updated or initialized when tracks are received from another sensor, as described in algorithm 4. The inputs are the consistent

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** $t \in 1 \ldots M(k)$ **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 3.4.2)
7:     $\hat{x}_t^{m,j}(k) = \langle$Use equation 3.12$\rangle$
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** $\hat{\beta}(k), \hat{X}^{local}(k)$
```

non-local tracks $\hat{X}^{non}(k-\tau)$ (maintained from the previous time step) and the consistent local tracks $\hat{X}_s^{local}(k)$ from any other sensor node s. Note that for the latter, "local" is 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 $t \in 1 \ldots M(k)$ 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 on when the information arrives and 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, and can only be applied in scenarios with at least two or more 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 target state information can be bias-compensated and fused in a standard approach as it arrives.

3.4.6 Computational Complexity

The proposed algorithm is intended for online use, so it is essential to have manageable computational requirements. The basic operations that dominate the computational load for this algorithm are computing matrix inversion and the square root of a matrix. The two major existing algorithms used in the proposed algorithm (CI and the UT) use these dominating basic operations.

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 $\mathcal{O}(n^{2.373})$ [83], they have large constant factors that are prohibitive for practical applications unless n is very large [84]. For this discussion, the computational complexity of matrix inversion will be considered to be cubic, though the real-time 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 $\mathcal{O}(n^3)$ [85].

Matrix inversion is the dominating factor of the computational complexity of the CI algorithm. 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 the optimization problem using the bisection method, requiring a constant number of matrix inversions (assuming the required precision to be constant). Therefore 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 does not specify what the function f is, so it must be considered as a variable factor. Therefore 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 for all targets at time index M is performed whenever information is received from other sensors. The information being fused is the target state with its dimension denoted as n_x. Accordingly, the computational complexity of the update operation for all targets is given in equation 3.23. For most practical applications using kinematic models to track non-cooperative targets, the target state vector does not get particularly large. In any case, the computational complexity is similar to a standard tracking application (without bias estimation) and is what must be expected when fusing information from other sensors.

$$\mathcal{O}(ProposedTrackUpdate) = \mathcal{O}(Mn_x^3) \tag{3.23}$$

The proposed measurement update (algorithm 3) performs the bias estimation and involves both CI and the UT. CI is used to update both the local consistent track (state vector of dimension n_x), as well as the bias estimate. The dimension of the bias vector is 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 3.4.2). The dimension of the measurement vector is denoted n_m, and the dimension of the joint distribution inputted into the UT is $2n_m + n_\beta$. The update step, line 20 of this algorithm, can be achieved in several ways. The 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 is dominated by matrix inversion in the dimension of the joint state. Each of these operations is required for each target used in the update. Combining these operations, the computational complexity of the proposed measurement update is given in equation 3.24.

$$\mathcal{O}(ProposedMeasurementUpdate) = \mathcal{O}(M((n_m + n_\beta)^3 + (n_m + n_\beta)f + (n_x + n_\beta)^3)) \quad (3.24)$$

Neither of the complexities in equations 3.23 or 3.24 depend directly on the number of sensors (denoted in this section without a time index as N). The number of sensors only comes into play when algorithm 4 is called, being called once for each update from each sensor. In this sense, the computation time of the proposed algorithm can be considered to grow linearly with the number of sensors N. It is also important that both of these algorithms depend linearly on the number of targets M. Comparing the proposed approach to the ASKF operating on sensors with uniform measurement and bias models, 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 is 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 cubic complexity in the number of sensors but linear in the number of targets. Therefor the proposed algorithm is much more computationally efficient for scenarios with a large number of sensors.

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 cubic complexity in the size of the state vector and linear in the number of targets.

3.5 Simulations and Discussion

To evaluate the proposed algorithm and demonstrate its effectiveness and flexibility, simulations were conducted for terrain-aided tracking using video sensors scenario. The proposed algorithm for bias estimation and compensation is compared against tracking with unbiased measurements, and with bias-ignorant measurements. These comparisons, using unbiased and bias-ignorant measurements, 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 3.4.4. 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 KF approach ignores some sources of correlation over time between the measurements and are based 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 assumes unknown correlation 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.

3.5.1 Scenario

In each Monte Carlo run, the true target and sensor trajectories are regenerated randomly according to their process noises and (static) initial conditions. To generate realistic ground truth for the target trajectories, a high resolution DEM with a resolution of 1m (from [13]) was used. A second DEM with a resolution of 30m (from [11]) was used during tracking, with the difference between the two DEMs representing realistic terrain elevation errors. The true sensor biases 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 [44]. 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 four sensors observing four 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 three 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 $3m$ for the positional components. In addition to the zero-mean noise, the angular measurements are 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 [44]. At each time step, the sensors communicate their tracks to one another after updating their local tracks and bias estimates 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 has a 50% chance to receive information from each other sensor at every time step.

3.5.2 Measurement Models for Terrain-Aided Tracking

The measurement model h_s^m of the form in equation 3.10, defining how the true additive bias affects the measurement, is given below in equation 3.25.

$$\begin{aligned} z_{s,t}^m(k) &= h_s^m(x_s^m(k), x_t^m(k), \beta_s, v_s(k)) \\ &= \begin{bmatrix} x_{s,x}(k) \\ x_{s,y}(k) \\ x_{s,z}(k) \\ x_{t,az}(k) \\ x_{t,el}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \beta_{az} \\ \beta_{el} \end{bmatrix} + v_s(k) \\ &= \bar{z}_{s,t}^m(k) + v_s(k) \\ &= \mathcal{N}(\bar{z}_{s,t}^m(k), R_s(k)) \end{aligned} \tag{3.25}$$

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 3.26.

$$\begin{aligned} z_{s,t}^c(k) &= h_s^c(z_s^m(k), \hat{\beta}_s(k), z_g) \\ \mathcal{Z}_i^c(k) &= h_s^c(\mathcal{Z}_i^m(k), \mathcal{B}_i(k), \mathcal{Z}_i^g) \end{aligned} \tag{3.26}$$

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 3.27.

$$\mathcal{Z}_i^{m,bc}(k) = \begin{bmatrix} \mathcal{Z}_{i,s,x}^m \\ \mathcal{Z}_{i,s,y}^m \\ \mathcal{Z}_{i,s,z}^m \\ \mathcal{Z}_{i,t,az}^m \\ \mathcal{Z}_{i,t,el}^m \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathcal{B}_{i,az} \\ \mathcal{B}_{i,el} \end{bmatrix} \tag{3.27}$$

Each sigma point represents 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, as expressed in equation 3.28.

$$\mathcal{Z}_i^c(k) = IntersectionPoint(\mathcal{Z}_i^{m,bc}(k), DEM, \mathcal{Z}_i^g) \tag{3.28}$$

The DEM intersection algorithm described in [2] is used to compute $\mathcal{Z}_i^c$ (see section 3.2.4). The final argument to the function in equation 3.28 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.

3.5.3 Algorithm and Scenario Variation Nomenclature

The algorithm proposed in section 3.4.5 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 3.4.3, Covariance Intersection (with unequal state dimension) from [70], and Covariance Bounds from [77] 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 3.5.1. 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.

3.5.4 Results

Figure 3.6 show the plots of the RMSE of the bias estimates (Euclidean distance error) over time from the algorithms and scenario variation for four different sensors. The differences between the plots for the various sensors are attributed to the different sensor locations, resulting in a different geometric relationship between the sensor, bias parameters, and the target positions. The differently-positioned sensors vary both in terms of their distance to the targets, and their points of view (affecting how their bias parameters correlate with the target state). In all cases the CIKF algorithm variation results in the lowest RMSE by the end of the simulation period, showing that the CIKF provides an effective balance between the optimistic KF and the pessimistic CI variations. The KF variation also performs well, and has the desirable property that the bias estimate RMSE trends consistently goes downwards over time. This trend is in contrast to the CI variation where the error can sometimes increase before correcting itself, and has a higher steady state error in general. The RMSE of the CIKF bias estimates also sometimes increases before correcting itself, but the magnitude of the increase is smaller and by the end of the simulation period this variation still achieves a lower estimation error. The CI variation lacks confidence in its estimates, giving too much weight to new observations, resulting in estimates which are slow to converge. The KF has the opposite problem, giving too much weight to its existing estimates. The CIKF variation sits somewhere in between. For the RMSE in bias estimate update, it is better to assume that there is no correlation than to assume the correlation to be totally unknown. This is expected since the bias estimation algorithm is structured such that the correlations are minimized. The FC scenario variations (solid lines) are also compared with the LC scenario variations (dashed lines) to investigate the performance of the proposed algorithm under limited communication. Across all sensors and algorithm variations, the LC scenario has slightly higher bias estimation error than the FC scenario, which is due to the delay in acquiring updates from other sensors. The results are very close between the two scenarios for the CIKF and KF algorithm variations, demonstrating that these proposed algorithm variations are effective even when communication is limited.

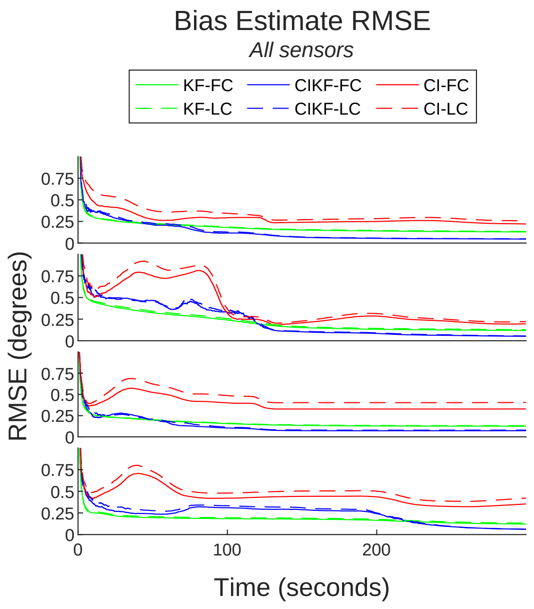

FIGURE 3.6: Root Mean Square Error (RMSE) for the bias estimates $\hat{\beta}_s$ for each sensor s.

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. Estimate consistency 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 3.7 and 3.8 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. All of the proposed algorithm variations produce bias-compensated measurements that are consistent, as been seen in figure 3.7 by noting that the ANEES curves are within the shaded 95% acceptance region. This figure also includes curves for the low-baseline bias-ignorant measurements and the high-baseline unbiased measurements. The bias-compensated measurements show similar consistency to the unbiased measurements, while the ANEES of the bias-ignorant measurements was so much higher such that a logarithmic scale was necessary to see the curve. This discrepancy demonstrates the importance of the bias compensation if consistent measurements are to be achieved. The local consistent tracks maintained by each of the proposed bias estimation algorithm variations are also consistent according to the ANEES test, as seen in figure 3.8. Curves for the unbiased and bias-ignorant trackers are not included in this figure because the local consistent track is a notion introduced in the bias compensation process (which does not occur in either the unbiased or bias-ignorant trackers).

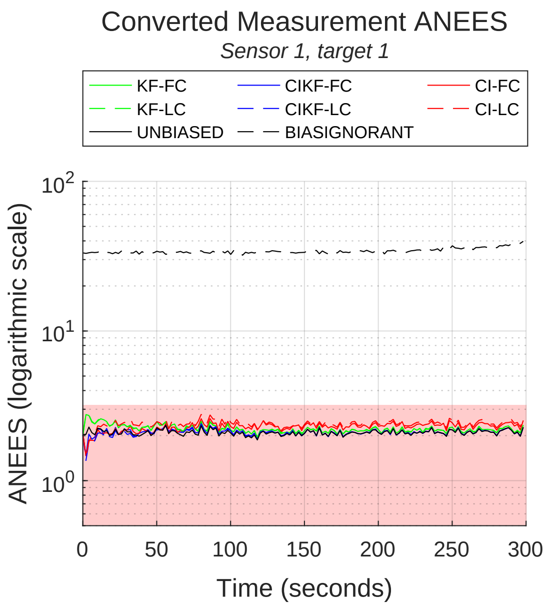

FIGURE 3.7: Average Normalized Estimation Error Squared (ANEES) for the converted bias-compensated measurements $z^c_{1,1}$, with the chi-square test acceptance region shaded.

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. The simplest approach is the standard KF framework operating on the converted bias-compensated measurements originating from local sensor node. Figure 3.9 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 is used due to the substantially higher RMSE of the bias-ignorant tracks. The lowest RMSE among the bias-compensated tracks comes from the CIKF update algorithm, followed closely by the KF variation, and then the CI variation. Similar to 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

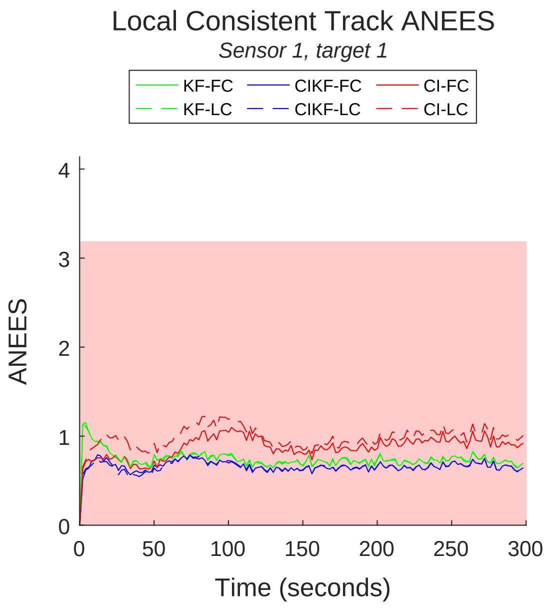

FIGURE 3.8: Average Normalized Estimation Error Squared (ANEES) for the local consistent tracks $\hat{x}_{1,1}^{local}$, with the chi-square test acceptance region shaded.

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 the bias-compensated track according to the bias estimates from CIKF update algorithm outperforms the unbiased tracker by the end of the simulation period. The unbiased tracker is included as a high performance baseline, so this is initially unexpected. As previously noted, the bias-compensated tracking application violates the assumptions of the KF that the measurement noise is zero-mean and independent over time (see sections 3.4.4 and 3.5.2). This violation can lead the filter to be inconsistent and optimistic in the accuracy of its estimates. The optimism is seen clearly in figure 3.10, showing the chi-square test results for these tracks. The ANEES metric 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 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 can be considered for further work. If the proposed algorithm is used in other applications that do not involve terrain models (for example, in stereo video or radar applications) the correlation of measurements over time may be less of an issue.

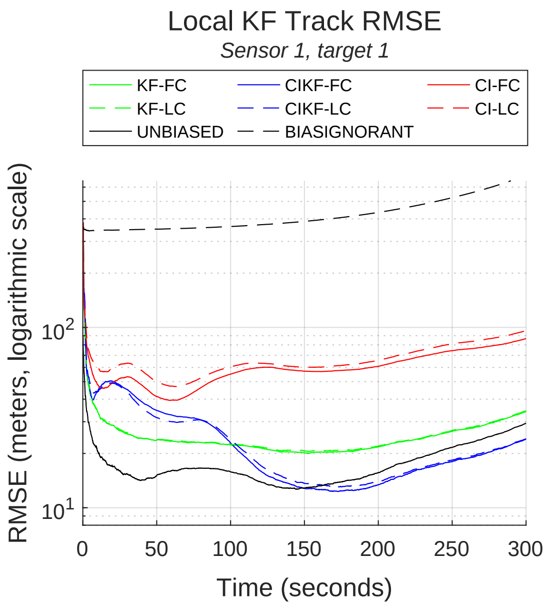

FIGURE 3.9: Root Mean Square Error (RMSE) plotted on a logarithmic scale for the tracks from standard KF operating on bias-compensated measurements $z^c_{1,1}$.

The standard KF cannot produce consistent tracks with measurements that are not independent over time unless this dependence is explicitly accounted for. In general explicit modeling of correlation may not always be possible, and indeed the 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. The consistency is at the expense of the estimation accuracy. This can be seen by comparing the RMSE of the local consistent tracks, shown in figure 3.11, with that of the standard KF tracks, shown in figure 3.9. 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 3.12 reveals that they are similar to the tracks from the standard KF tracker. The chi-square test on the ANEES of these tracks confirms that they are also consistent, as seen in figure 3.13.

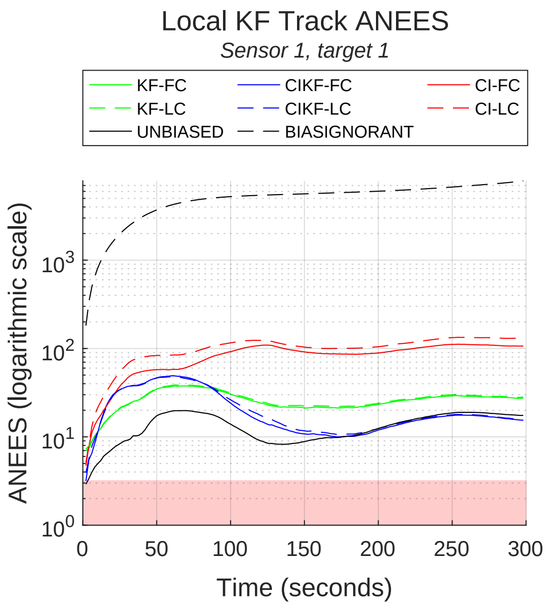

FIGURE 3.10: Average Normalized Estimation Error Squared (ANEES) for the tracks from standard KF operating on bias-compensated measurements $z^c_{1,1}$, with the chi-square test acceptance region shaded.

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 the similarity, the RMSE of the standard KF tracks, local consistent tracks, and fused tracks for all sensors and targets are included in figures 3.14, 3.15, and 3.16, respectively. Unlike the more detailed plots, a linear scale is used 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.

3.5.5 Limitations

The proposed bias estimation algorithm is not theoretically optimal, since various correlations are assumed to be unknown during the bias estimation process, or are neglected in some algorithm variations. The bias estimate update step also requires fusion under unknown correlation with unequal state dimensions. This problem is an area of active research, so future developments here may lead to direct improvements of performance in the proposed algorithm if they can be effectively applied to it. Lastly, bias-compensated tracking involves updating tracks using measurements which have are correlated through the bias compensation process. Although the CI algorithm has been used in this chapter to maintain consistency, RMSE suffered as a result. Future work may look at separating the possibly-correlated and independent components of the bias-compensated measurements and using tools such as Split Covariance Intersection (SCI) [86] to improve bias-compensated tracking performance.

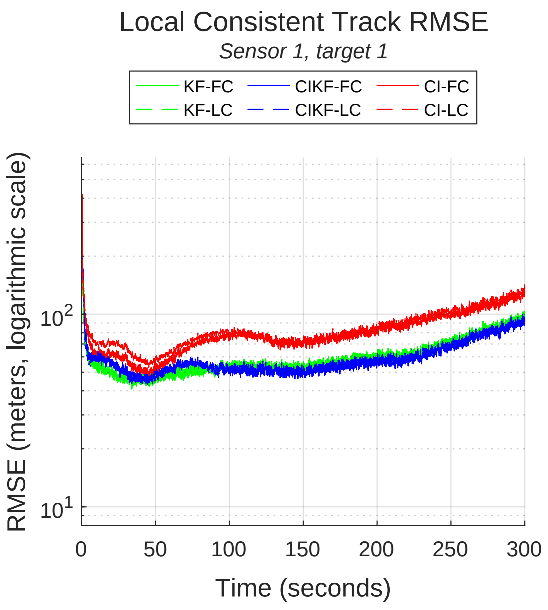

FIGURE 3.11: Root Mean Square Error (RMSE) plotted on a logarithmic scale for the local consistent tracks $\hat{x}_{1,1}^{local}$.

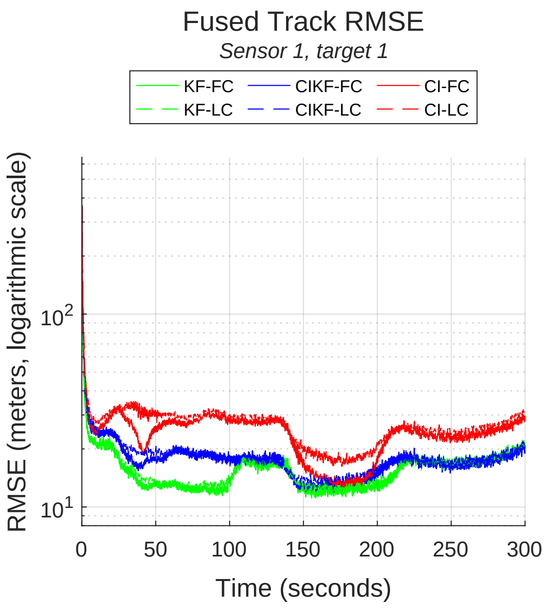

FIGURE 3.12: Root Mean Square Error (RMSE) plotted on a logarithmic scale for the tracks obtained by fusing the local consistent tracks $\hat{x}_{1,1}^{local}$ and non-local consistent tracks $\hat{x}_{1,1}^{nonlocal}$ using CI.

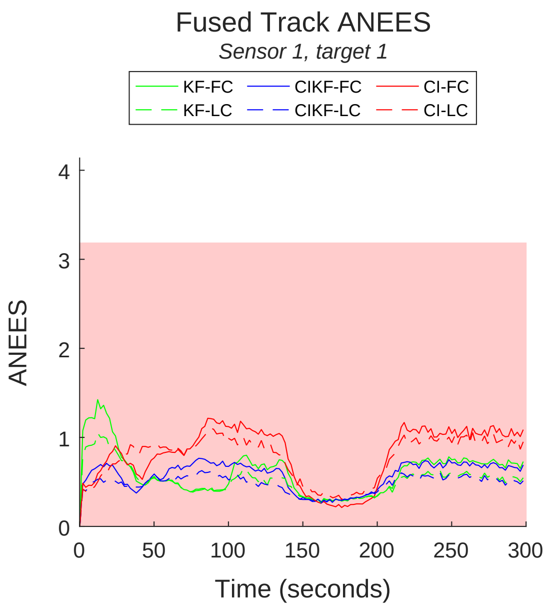

FIGURE 3.13: Average Normalized Estimation Error Squared (ANEES) for the tracks obtained by fusing the local consistent tracks $\hat{x}_{1,1}^{local}$ and non-local consistent tracks $\hat{x}_{1,1}^{nonlocal}$ using CI, with the chi-square test acceptance region shaded.

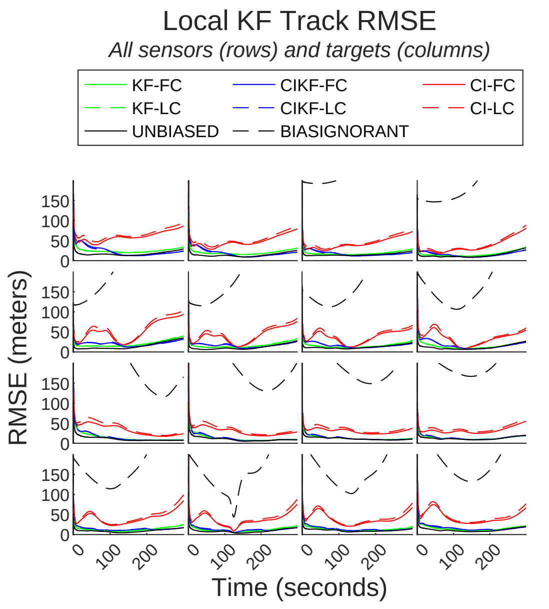

FIGURE 3.14: Root Mean Square Error (RMSE) for the tracks from standard KF operating on bias-compensated measurements $z_{s,t}^{c}$.

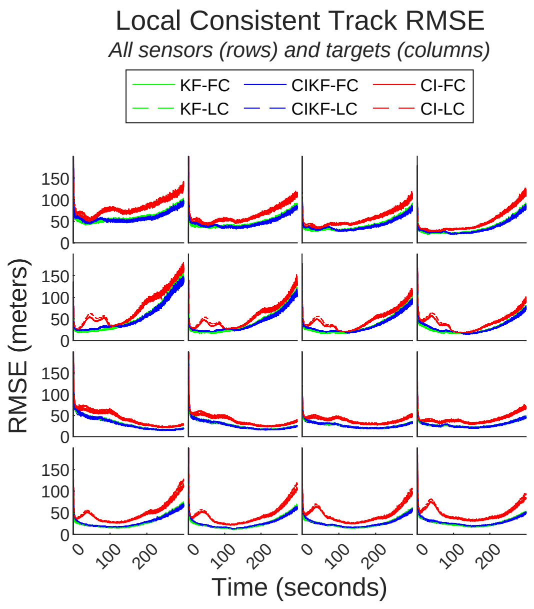

FIGURE 3.15: Root Mean Square Error (RMSE) for the local consistent tracks $\hat{x}_{s,t}^{local}$.

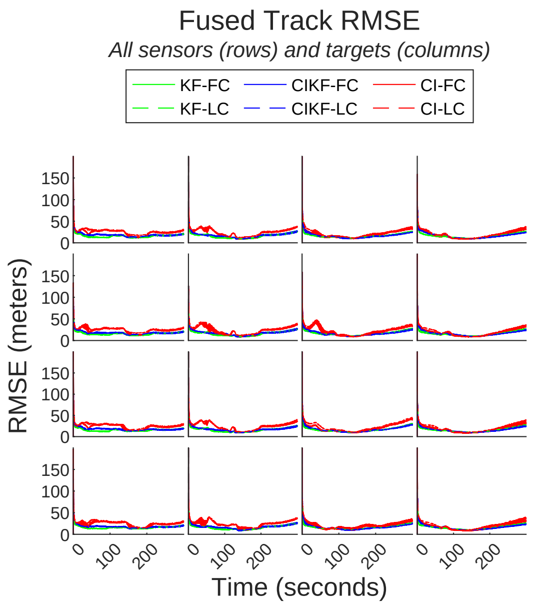

FIGURE 3.16: Root Mean Square Error (RMSE) for the tracks obtained by fusing the local consistent tracks $\hat{x}_{s,t}^{local}$ and non-local consistent tracks $\hat{x}_{s,t}^{nonlocal}$ using CI.

3.6 Conclusion

In this chapter, a computationally efficient algorithm for bias estimation and compensation for practical multisensor-multitarget tracking was presented. The algorithm was achieved by restructuring the estimation problem in order to avoid inconsistency and leveraging on flexible tools such as the Unscented Transform and Covariance Intersection. Covariance Intersection was used to compile state information about a single target and bias estimate information about a single sensor, while the Unscented Transform was used to form joint estimate of the target state and bias estimate. Simulation results of the algorithm showed low bias estimate RMSE for all sensors, consistent bias-compensated measurements, and improved bias-compensated track RMSE comparable to that of a tracker operating on unbiased measurements. The proposed algorithm is suitable for improving tracking accuracy, track probability of detection, and few missed associations.

Although the algorithm showed improvement across several measures, the challenges in bias estimation and compensation are not completely eliminated. Further works on performing fusion under unknown correlation with unequal state dimensions and improving the consistency of bias-compensated tracking are needed to eliminate these challenges.

Chapter 4

Unscented Homography Transform for Tracking in Pixel Coordinates

4.1 Introduction

Imaging sensors such as visible light or infrared (IR) video cameras are frequently used in tracking applications due to their high accuracy, high frame rate, low cost, low bulk, and passive operation. There are a wide range of video target tracking approaches employed in different applications and under different operational conditions. Some methods convert the measured target positions to cartesian coordinates in a global frame of reference [87, 2, 3, 88]. This usually requires information such as measurements of the sensor position and orientation, terrain model, measurements from multiple sensors, or known points of reference, in addition to the video measurements of the target pixel positions. Another option is to track the targets in pixel coordinates [89, 90, 91]. Tracking in pixel coordinates may be used when the global position of the targets is not required, or when additional information to convert the measurements to a global reference frame is not available.

Although tracking in pixel coordinates alleviates the need to determine the global position of targets, it becomes necessary to account for the affects of sensor motion. The homography transform can be used to account for sensor motion by transforming a set of planar points from their positions in one view to their positions in another view. This relies on the assumption that the targets lie at least approximately in the same plane. [1]. This assumption is oftentimes accurate especially when tracking ground targets in flat terrain, hence the homography transform is often used [92, 93, 94].

The homography transform has also been extended to transform multiple sets of planar points, relaxing the limitation that all of the points to be transformed must lie in the same plane [95, 96]. This was achieved by computing multiple homography matrices that are consistent with one another, and using each matrix to transform its corresponding set of planar points. Another limitation of the homography transform is that it is designed to be used on sets of points, with no notion of uncertainty. This limits the use of the homography transform in combination with the Kalman Filter, which most tracking frameworks are based on. The Kalman Filter maintains estimates of target states in the form of Gaussian random variables and also estimates multiple state variables such as velocity, acceleration, and/or turn rate, in addition to position. This makes its integration with the homography transform nontrivial.

With a high frame rate and a relatively accurate detector, it may be sufficient to neglect measurement uncertainty and the specifics of target motion dynamics, avoiding the need for the Kalman Filter and therefore also avoiding this limitation. This practice has been used effectively in practical scenarios [97, 98]. In other scenarios, there may be non-negligible measurement or system uncertainty [99, 100], and high frame rate tracking many not be achievable (for example, due to a computationally-intensive detection process [101] or a large number of tracked targets). Therefore the integration of the Kalman filter with the homography transform remains relevant, but limited work has been done on the subject. This problem is partially addressed in [102]. Their approach allows for the transformation of the full mean of the target state for targets moving according to the Nearly Constant Velocity (NCV) motion model (position and velocity), as well as the elements of the covariance corresponding to the position components of the state. For targets where the NCV model is sufficient, this approach provides a good approximation of the fully-transformed state estimate with uncertainty. The limitations of this approach are that only the position components of the covariance matrix are transformed, and other motion models were not considered.

This chaper focuses on the problem of compensating for sensor motion in tracking scenarios where the assumption that targets are on a nearly-planar surface applies, but the measurement and/or system uncertainty is not negligible, and the frame rate may be low enough to warrant the use of various target kinematic motion models. The approach proposed in this chaper aims to address the remaining limitations of the approach from [102], to allow for the full transformation of the mean and covariance of target state estimates with various motion models.

The first contribution of this chaper is to outline an approach for homography transformation of full target state estimates (in the form of Gaussian random variables) that can be applied with common kinematic motion models used in tracking. The Unscented Transform is used to handle uncertainty, reducing the problem of transforming the distribution of the target state to that of transforming sigma points in the target state space. The second contribution of this chaper is to provide specific derivations of the state vector sigma point transformation functions for three common target kinematic motion models, namely the nearly constant velocity (NCV), nearly constant acceleration (NCA), and nearly coordinated turn (NCT) models.

The remainder of this chaper is organized as follows. Section 4.2 provides relevant mathematical background related to sensor motion compensation in video tracking, probabilistic tracking frameworks, target motion models, and the Unscented Transform. The proposed solution approach, along with all mathematical details, is presented in section 4.3. The performance of this approach is evaluated in section 4.4, which provides details about simulations conducted for this purpose and the performance evaluation results. A conclusion is presented in Section 4.5.

4.2 Background

4.2.1 Homography Transform

The homography transform (HT) can be used to account for camera motion between two video frames when all targets lie within the same plane (or when this closely approximates the true scenario). The homography maps a set of planar points from their position in the previous frame to their corresponding position in the following frame [1]. For tracking ground targets in flat terrain, the set of planar points is the ground plane. This is illustrated in figure 4.1, adapted from [1].

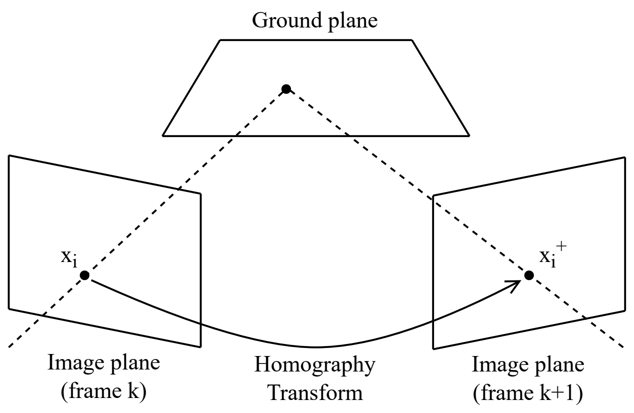

FIGURE 4.1: Illustration of Homography Transformation for mapping planar points form one frame to another. This figure is adapted from [1].

The HT is defined by the 3-by-3 homography matrix H. There are several methods for automatically estimating the homography matrix between two frames [1, 103], so the matrix is assumed to be available (known). If the 2-dimensional image point x_i, with coordinates ξ and η, in the previous frame is expressed in homogeneous coordinates as x_h, then the homography transformation can be applied by left-multiplying the point x_h with H [1]. This is shown in equation 4.1, where the resulting point x_h^+ is the homogeneous coordinate representation of the point in the following frame.

$$\begin{aligned} x_i &= \begin{bmatrix} \xi & \eta \end{bmatrix}^T \\ x_h &= \begin{bmatrix} \xi & \eta & 1 \end{bmatrix}^T \\ x_h^+ &= H x_h \\ &= \begin{bmatrix} c\xi^+ & c\eta^+ & c \end{bmatrix}^T \end{aligned} \tag{4.1}$$

Dividing x_h^+ by its third component c yields the 2-dimensional image coordinates of the point in the following frame x_i^+ (equation 4.2).

$$x_i^+ = \begin{bmatrix} \xi^+ & \eta^+ \end{bmatrix}^T \tag{4.2}$$

The notation in equation 4.3 will be used as a shorthand for the above process throughout this chaper when applying the homography transformation between points in successive frames. When the planar assumption holds, this approach is effective in transforming points to account for camera motion.

$$x_i^+ = homography(x_i) \tag{4.3}$$

Note that in its standard form, the HT can only be used to transform position points and has no notion of uncertainty. In [102] this is partially addressed for the specific case of the CV model by deriving a first-order Taylor series approximation of the transformed position covariance and transformed velocity. The position mean was transformed using the standard homography transform (equation 4.4).

$$p_2 = homography(p_1) \tag{4.4}$$

In [102], the homography matrix H is decomposed as in equation 4.5, and the matrix G is computed as in equation 4.6 and used to transform the velocity mean in equation 4.7.

$$H = \begin{bmatrix} H_{1(2-by-2)} & h_{2(2-by-1)} \\ h^T_{3(1-by-2)} & h_{4(1-by-1)} \end{bmatrix} \tag{4.5}$$

$$G = \frac{(h_3^T p_1 + h_4)H_1 - (H_1 p_1 + h_2)h_3^T}{(h_3^T p_1 + h_4)^2} \tag{4.6}$$

$$v_2 = G v_1 \tag{4.7}$$

The matrix G is also used in [102] to compute the transformed covariance of the position components P_2 from the previous frame covariance P_1, as in equation 4.8.

$$P_2 = G P_1 G^T \tag{4.8}$$

Their work did not address the covariance in the velocity components of the state (nor the velocity-position cross-covariance components) or any additional kinematic models.

4.2.2 Kalman Filter Tracking and Target Kinematic Models

The Kalman filter (KF) [104] is used as the basis for many probabilistic tracking frameworks in a variety of applications [100, 105, 106]. It is the optimal state estimator (minimum mean squared error) in linear systems with Gaussian noises [44]. The estimate takes the form of a Gaussian distribution, as in equation 4.9. The KF is also the underlying filter applied in the interacting multiple model (IMM) filter [44], used to handle maneuvering targets, and in multiple hypothesis tracking (MHT), used to resolve data association ambiguity [107]. These are not discussed in detail here and the reader is referred to the corresponding references for more information.

$$\hat{x} = \mathcal{N}(\bar{x}, P_x) \tag{4.9}$$

The KF uses a discrete-time linear model of the target dynamics with additive zero-mean white Gaussian process noise, as in equation 4.10. The target estimate $\hat{x}$ at time k is predicted to time $k+1$ according to the system model $F(k)$, and white Gaussian noise with covariance $Q(k)$ is added to account for unknown effects on the target state. The conditional probability notation $\hat{x}(k+1|k)$ indicates that this is an estimate of the target state at time $k+1$, based on the information gathered up to time k.

$$\begin{aligned} \bar{x}(k+1|k) &= F(k)\bar{x}(k|k) \\ P_x(k+1|k) &= F(k)P_x(k|k)F(k)^T + Q(k) \end{aligned} \tag{4.10}$$

The state estimate is updated recursively with new information from measurements as they arrive. Once the state estimate has been predicted to the time of the measurement, the update is performed as in equation 4.11.

$$\begin{aligned} \bar{y}(k) &= \bar{z}(k) - H(k)\bar{x}(k+1|k) \\ S(k) &= H(k)P_x(k+1|k)H(k)^T + R(k) \\ K(k) &= P_x(k+1|k)H(k)^T S(k)^{-1} \\ \bar{x}(k+1|k+1) &= \bar{x}(k+1|k) + K(k)\bar{y}(k) \\ P_x(k+1|k+1) &= (I - K(k)H(K))P_x(k+1|k) \end{aligned} \tag{4.11}$$

The measurement model H describes how the state is observed (in terms of the measurement), and the system model F describes how the target state evolves over time and determines which state components are tracked. The system model should be chosen to accurately reflect the target dynamics as much as possible. This chaper focuses on tracking in pixel coordinates, hence the measurements are assumed to be the 2-dimensional pixel position of the target. If the measurements are not in pixel coordinates, a different measurement model will have to be used for other applications (for example, a range-azimuth measurement from a radar or lidar). Note that the measurement model H is not the same as the homography matrix, which uses the same letter for its standard notation. The three target kinematic models discussed here are the nearly constant velocity model, nearly constant acceleration model, and nearly coordinated turn model.

Nearly Constant Velocity Model

In the nearly constant velocity model (NCV), the state estimate consists of the target position and velocity. The state vector of a NCV model with two spatial dimensions (in coordinates ξ, η) is given in equation 4.12.

$$x_{NCV} = \begin{bmatrix} \xi & \eta & \dot{\xi} & \dot{\eta} \end{bmatrix}^T \tag{4.12}$$

In the NCV model, the position and velocity are tracked explicitly and the process noise is modelled by small random accelerations of the target. The corresponding state transition and process noise covariance matrices are given in equation 4.13 [44]. An alternative formulation of the process noise covariance can be used to model process noise which is stronger in the direction of the target motion. This is called a directional nearly constant velocity (DNCV) Model [108].

$$\begin{aligned} F_{NCV}(T) &= \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \Gamma_{NCV}(T) &= \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ 0 & \frac{1}{2}T^2 \\ T & 0 \\ 0 & T \end{bmatrix} \\ Q_{NCV}(T) &= \Gamma_{NCV}\sigma^2_{NCV}\Gamma^T_{NCV} \end{aligned} \tag{4.13}$$

If the measurement consists of the target position, then the corresponding measurement model is given below in equation 4.14, which simply selects the position elements of the state vector.

$$H_{NCV} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.14}$$

Nearly Constant Acceleration Model

The state vector for the nearly constant acceleration (NCA) model consists of the target position, velocity, and acceleration. The state vector for a NCA model with two spatial dimensions is given below in equation 4.15.

$$x_{NCA} = \begin{bmatrix} \xi & \eta & \dot{\xi} & \dot{\eta} & \ddot{\xi} & \ddot{\eta} \end{bmatrix}^T \tag{4.15}$$

The NCA model tracks the position, velocity, and acceleration directly, and uses process noise to model small changes in acceleration. The corresponding state transition and process noise covariance matrices are given in equation 4.16 [44]. As with the NCV/DNCV models, an alternative formulation of the process noise covariance can be used if the noise is stronger in the direction of the target acceleration, resulting in a directional nearly constant acceleration (DNCA) Model [108].

$$\begin{aligned} F_{NCA}(T) &= \begin{bmatrix} 1 & 0 & T & 0 & \frac{1}{2}T^2 & 0 \\ 0 & 1 & 0 & T & 0 & \frac{1}{2}T^2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\ \Gamma_{NCA}(T) &= \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ 0 & \frac{1}{2}T^2 \\ T & 0 \\ 0 & T \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\ Q_{NCA}(T) &= \Gamma_{NCA}\sigma_{NCA}^2\Gamma_{NCA}^T \end{aligned} \tag{4.16}$$

Similar to the NCV model, for a measurement consisting of the target position, the corresponding measurement model (equation 4.17) selects the measured components of the state vector.

$$H_{NCA} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.17}$$

Nearly Coordinated Turn Model

The nearly coordinated turn (NCT) model is used to model targets moving with a nearly constant turn rate and nearly constant speed. The state vector for a NCT model with two spatial dimensions is given in equation 4.18, where Ω is the turn rate.

$$x_{NCT} = \begin{bmatrix} \xi & \eta & \dot{\xi} & \dot{\eta} & \Omega \end{bmatrix}^T \tag{4.18}$$

In the NCT model, the target position, velocity and turn rate are tracked, and the process noise is used to model small changes to the velocity and turn rate. The corresponding state transition and process noise covariance matrices are given in equation 4.19 [44].

$$F_{NCT}(T) = \begin{bmatrix} 1 & 0 & \frac{sin(\Omega T)}{\Omega} & -\frac{1-cos(\Omega T)}{\Omega} & 0 \\ 0 & 1 & \frac{1-cos(\Omega T)}{\Omega} & \frac{sin(\Omega T)}{\Omega} & 0 \\ 0 & 0 & cos(\Omega T) & -sin(\Omega T) & 0 \\ 0 & 0 & sin(\Omega T) & cos(\Omega T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$
$$\Gamma_{NCT}(T) = \begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 \\ T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{bmatrix} \tag{4.19}$$
$$Q_{NCT}(T) = \Gamma_{NCT}\sigma^2_{NCT}\Gamma^T_{NCT}$$

Notice that unlike the NCV and NCA models, the NCT model is nonlinear (F_{NCT} is a nonlinear function of the state component Ω). This is an important distinction since it means that the standard KF cannot be used directly to update the state estimate and its covariance. The extended Kalman filter (EKF), based on a linearization of the state transition model, or the unscented Kalman filter (UKF), based on a Gaussian approximation of a Gaussian random variable that has undergone a nonlinear transformation, can be used instead of the KF. An example of the EKF applied to the NCT model can be found in [44].

When the turn rate is zero, or nearly zero, the target is effectively following a nearly constant velocity model. The limiting form of the NCT model should then be used. This is given in 4.20, adapted from [44].

$$F^{\Omega=0}_{NCT}(T) = \begin{bmatrix} 1 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

The measurement model (assuming measurement of the target position only) selects the appropriate state components, much like in the NCV and NCA models. The corresponding measurement model is given in equation 4.21.

$$H_{NCT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.21}$$

4.2.3 Unscented Transform

Consider the nonlinear transformation in equation 4.22, where x is a Gaussian random variable.

$$\begin{aligned} y &= f(x) \\ x &= \mathcal{N}(\bar{x}, P_x) \end{aligned} \tag{4.22}$$

The unscented transform (UT) can be used to estimate the mean $\bar{y}$ and covariance P_y of the random variable y [60]. Then $\hat{y}$ (equation 4.23) is a Gaussian approximation of y.

$$\hat{y} = \mathcal{N}(\bar{y}, P_y) \tag{4.23}$$

The UT is a very flexible and powerful tool, as the only requirement on the transformation function f is that it is possible to compute transformed points $p_y = f(p_x)$. The UT has been used successfully in a wide range of circumstances [63, 3, 109].

The UT works by computing a set of *sigma points* $\mathcal{X}_i$ from the distribution of x and their corresponding weights W_i. These points are then transformed to compute the transformed sigma points $\mathcal{Y}_i = f(\mathcal{X}_i)$, which are used to estimate the mean and covariance of y. The set of $2n + 1$ sigma points (where n is the dimension of random variable x) and their weights, are computed as in equation 4.24 [60]. The notation $(\sqrt{(n+\kappa)P_x})_i$ refers to the ith column of the matrix square root.

$$\begin{aligned} \mathcal{X}_0 &= \bar{x} & W_0 &= \frac{\kappa}{n+\kappa} \\ \mathcal{X}_i &= \bar{x} + (\sqrt{(n+\kappa)P_x})_i & W_i &= \frac{1}{2(n+\kappa)} \\ \mathcal{X}_{i+n} &= \bar{x} - (\sqrt{(n+\kappa)P_x})_i & W_{i+n} &= \frac{1}{2(n+\kappa)} \end{aligned} \tag{4.24}$$

In equation 4.24, the parameter κ controls the spread of the sigma points. Other sets of sigma points can be used [110, 111, 112] but the approach for transforming a small, deterministic set of points to estimate the transformed distribution parameters remains the same. Once the transformed sigma points $\mathcal{Y}_i$ are computed, the mean and covariance of y are estimated as in equation 4.25.

$$\begin{aligned} \bar{y} &= \Sigma_{i=0}^{2n} W_i \mathcal{Y}_i \\ P_y &= \Sigma_{i=0}^{2n} W_i (\mathcal{Y}_i - \bar{y})(\mathcal{Y}_i - \bar{y})^T \end{aligned} \tag{4.25}$$

4.3 Unscented Homography Transform

In this chaper, the unscented homography transform (UHT) is proposed as a method to transform tracks for pixel-coordinate video tracking. The standard HT is used to transform position points from the reference point of the previous frame to that of the following frame in video tracking. The proposed UHT can be used to transform the entire state vector and its covariance and it is not limited to operating in the position space. This makes the UHT compatible with commonly-used kinematic target motion models, such as the NCV, NCA, and NCT models, and with the standard probabilistic tracking framework. The UHT is used before the prediction step when updating the KF with new measurements, with the rest of the KF tracking framework remaining unchanged.

In the proposed approach, the UT handles the uncertainty in the target state estimate. A set of $2N_s + 1$ Sigma points $\mathcal{X}_i$ are computed from the distribution of the state estimate $\hat{x}$ of dimension N_s, according to the standard UT algorithm. Each sigma point, representing one possible target state, is then encoded as a set of N_t trajectory samples (position points) $\mathcal{P}_{i,j}$ over a short period of time. These trajectory samples can then be transformed to the coordinate reference of the following frame using the standard HT. Once in the appropriate reference frame, the position samples are then used to compute the transformed sigma points (in the space of the estimated state). The standard UT is then used to compute the transformed approximate Gaussian estimate. This process requires defining an appropriate sigma point transformation function f (according to the target motion model and homography matrix) for use in the UT.

4.3.1 Transformation Functions for Common Target Kinematic Motion Models

Sigma point transformation functions f used in the UHT are given for each of the commonly-used target kinematic models. These functions are applied to transform the sigma points $\mathcal{X}_i$ from the original distribution $\hat{x}$ to sigma points $\mathcal{X}_i^+$ in the transformed distribution $\hat{x}^+$ by the UT. Note that the state estimate $\hat{x}$ corresponds to the particular motion model being discussed and the process noise is not considered due to the short duration of the time periods over which the trajectory is considered ($Q \approx \mathbf{0}$).

Nearly Constant Velocity Model

In the NCV model, two trajectory points are required to encode the position and velocity of the target ($N_t = 2$) as four equations are required to solve for the four state components. These points are computed and spaced over a short interval of time ϵ, as in equation 4.26.

$$\begin{aligned} \mathcal{P}_{i,0} &= H_{NCV}\mathcal{X}_i \\ \mathcal{P}_{i,1} &= H_{NCV}F_{NCV}(\epsilon)\mathcal{X}_i \end{aligned} \tag{4.26}$$

The trajectory points $\mathcal{P}_{i,j}$ are then transformed to their corresponding positions in the following frame $\mathcal{P}_{i,j}^+$, according to the standard homography transform.

$$\mathcal{P}_{i,j}^+ = homography(\mathcal{P}_{i,j}) \tag{4.27}$$

From these transformed trajectory points, the transformed sigma points $\mathcal{X}_i^+$ are computed in the space of the estimated state, with reference to the following frame. The position components $\mathcal{P}_{i,0}^+$ are already known, and the velocity components $\mathcal{V}_{i,0}^+$ are computed according to the target motion model F_{NCV}.

$$\begin{aligned} F_{NCV}(\epsilon)\mathcal{X}_i^+ &= \begin{bmatrix} \mathcal{P}_{i,1}^+ \\ \mathcal{V}_i^+ \end{bmatrix} \\ \mathcal{V}_i^+ &= \frac{\mathcal{P}_{i,1}^+ - \mathcal{P}_{i,0}^+}{\epsilon} \\ \mathcal{X}_i^+ &= \begin{bmatrix} \mathcal{P}_{i,0}^+ \\ \mathcal{V}_i^+ \end{bmatrix} \end{aligned} \tag{4.28}$$

The sigma point transformation function used in the UT for the NCV model is defined (according to equations 4.26 through 4.28) as in equation 4.29 below.

$$\mathcal{X}_i^+ = f_{NCV}(\mathcal{X}_i) \tag{4.29}$$

Nearly Constant Acceleration Model

The NCA model is similar to the NCV model, but three trajectory points are required to encode the position, velocity, and acceleration of the target ($N_t = 3$) to solve for the six state components. These trajectory points are given in equation 4.30.

$$\begin{aligned} \mathcal{P}_{i,0} &= H_{NCA}\mathcal{X}_i \\ \mathcal{P}_{i,1} &= H_{NCA}F_{NCA}(\epsilon)\mathcal{X}_i \\ \mathcal{P}_{i,2} &= H_{NCA}F_{NCA}(\epsilon)F_{NCA}(\epsilon)\mathcal{X}_i \end{aligned} \tag{4.30}$$

After transforming the trajectory points to their positions in the following frame (as in equation 4.27), the sigma points of the transformed state can be computed according to the target motion model F_{NCA}.

$$\begin{aligned}
F_{NCA}(\epsilon)\mathcal{X}_i^+ &= \begin{bmatrix} \mathcal{P}_{i,1}^+ \\ \mathcal{V}_{i,1}^+ \\ \mathcal{A}_i^+ \end{bmatrix} \\
F_{NCA}(\epsilon)F_{NCA}(\epsilon)\mathcal{X}_i^+ &= \begin{bmatrix} \mathcal{P}_{i,2}^+ \\ \mathcal{V}_{i,2}^+ \\ \mathcal{A}_i^+ \end{bmatrix} \\
\mathcal{V}_{i,0}^+ &= \frac{\mathcal{P}_{i,1}^+ - \mathcal{P}_{i,0}^+}{\epsilon} \\
\mathcal{V}_{i,1}^+ &= \frac{\mathcal{P}_{i,2}^+ - \mathcal{P}_{i,1}^+}{\epsilon} \\
\mathcal{A}_i^+ &= \frac{\mathcal{V}_{i,1}^+ - \mathcal{V}_{i,0}^+}{\epsilon} \\
\mathcal{X}_i^+ &= \begin{bmatrix} \mathcal{P}_{i,0}^+ \\ \mathcal{V}_{i,0}^+ \\ \mathcal{A}_i^+ \end{bmatrix}
\end{aligned} \tag{4.31}$$

The sigma point transformation function used in the UT for the NCA model is defined (according to equations 4.30 through 4.31) as in equation 4.32 below.

$$\mathcal{X}_i^+ = f_{NCA}(\mathcal{X}_i) \tag{4.32}$$

Nearly Coordinated Turn Model

The NCT model has five state components, so three trajectory points will be required to encode the state ($N_t = 3$) as in equation 4.33. Note that F_{NCT} is also a function of the turn rate Ω which is fixed according to the value of the corresponding sigma point component $\mathcal{O}$.

$$\begin{aligned}
\mathcal{P}_{i,0} &= H_{NCT}\mathcal{X}_i \\
\mathcal{P}_{i,1} &= H_{NCT}F_{NCT}(\epsilon)\mathcal{X}_i \\
\mathcal{P}_{i,2} &= H_{NCT}F_{NCT}(\epsilon)F_{NCT}(\epsilon)\mathcal{X}_i
\end{aligned} \tag{4.33}$$

These trajectory points are then transformed using the homography transform, as with other motion models (see equation 4.27).

Converting the trajectory samples back to the state components for the NCV and NCA models is complex due to F_{NCT} being a nonlinear function of the turn rate. The position components $\mathcal{P}_{i,0}$ are known and the velocity and turn rate components ($\mathcal{V}_{i,0}^+$ and $\mathcal{O}_i^+$ respectively) are obtained from the motion model F_{NCT} as in equation 4.34.

$$
\begin{aligned}
F_{NCT}(\epsilon)\mathcal{X}_i^+ &= \begin{bmatrix} \mathcal{P}_{i,1}^+ \\ \mathcal{V}_{i,1}^+ \\ \mathcal{O}_i^+ \end{bmatrix} \\
F_{NCT}(\epsilon)F_{NCT}(\epsilon)\mathcal{X}_i^+ &= \begin{bmatrix} \mathcal{P}_{i,2}^+ \\ \mathcal{V}_{i,2}^+ \\ \mathcal{O}_i^+ \end{bmatrix}
\end{aligned}
\tag{4.34}
$$

The following notation for the individual sigma point state components will be used (equation 4.35).

$$
\begin{aligned}
\mathcal{P}_{i,j}^+ &= \begin{bmatrix} \xi_{i,j} & \eta_{i,j} \end{bmatrix}^T \\
\mathcal{V}_{i,j}^+ &= \begin{bmatrix} \dot{\xi}_{i,j} & \dot{\eta}_{i,j} \end{bmatrix}^T
\end{aligned}
\tag{4.35}
$$

The state transition (from equation 4.34) for the individual position components is given in equation 4.36.

$$
\begin{aligned}
\xi_{i,1} &= \xi_{i,0} + \frac{sin(\mathcal{O}_i^+\epsilon)}{\mathcal{O}_i^+}\dot{\xi}_{i,0} - \frac{1 - cos(\mathcal{O}_i^+\epsilon)}{\mathcal{O}_i^+}\dot{\eta}_{i,0} \\
\eta_{i,1} &= \eta_{i,0} + \frac{1 - cos(\mathcal{O}_i^+\epsilon)}{\mathcal{O}_i^+}\dot{\xi}_{i,0} + \frac{sin(\mathcal{O}_i^+\epsilon)}{\mathcal{O}_i^+}\dot{\eta}_{i,0}
\end{aligned}
\tag{4.36}
$$

Recall that ϵ can be chosen to be arbitrarily small (small enough to force $\mathcal{O}_i\epsilon$ to also be small). Therefore the small-angle approximations (equation 4.37) for trigonometric functions can be applied.

$$
\begin{aligned}
sin(\theta) &\approx \theta \\
cos(\theta) &\approx 1
\end{aligned}
\tag{4.37}
$$

Using the small-angle approximations, equation 4.36 is simplified as in equation 4.38.

$$
\begin{aligned}
\xi_{i,1} &\approx \xi_{i,0} + \frac{\mathcal{O}_i^+\epsilon}{\mathcal{O}_i^+}\dot{\xi}_{i,0} - \frac{1-1}{\mathcal{O}_i^+}\dot{\eta}_{i,0} \\
&= \xi_{i,0} + \epsilon\dot{\xi}_{i,0} \\
\eta_{i,1} &\approx \eta_{i,0} + \frac{1-1}{\mathcal{O}_i^+}\dot{\xi}_{i,0} + \frac{\mathcal{O}_i^+\epsilon}{\mathcal{O}_i^+}\dot{\eta}_{i,0} \\
&= \eta_{i,0} + \epsilon\dot{\eta}_{i,0}
\end{aligned}
\tag{4.38}
$$

This justifies the approximation of the velocity components using the same approach as in the NCV and NCA models (equation 4.39).

$$\begin{aligned} \mathcal{V}_{i,0}^{+} &\approx \frac{\mathcal{P}_{i,1}^{+} - \mathcal{P}_{i,0}^{+}}{\epsilon} \\ \mathcal{V}_{i,1}^{+} &\approx \frac{\mathcal{P}_{i,2}^{+} - \mathcal{P}_{i,1}^{+}}{\epsilon} \end{aligned} \tag{4.39}$$

Consider the state transition (from equation 4.34) for the individual velocity components, given in equation 4.40.

$$\begin{aligned} \dot{\xi}_{i,1} &= cos(\mathcal{O}_i^{+}\epsilon)\dot{\xi}_{i,0} - sin(\mathcal{O}_i^{+}\epsilon)\dot{\eta}_{i,0} \\ \dot{\eta}_{i,1} &= sin(\mathcal{O}_i^{+}\epsilon)\dot{\xi}_{i,0} + cos(\mathcal{O}_i^{+}\epsilon)\dot{\eta}_{i,0} \end{aligned} \tag{4.40}$$

These equations can be simplified using the small angle approximations from equation 4.37, as in equation 4.41.

$$\begin{aligned} \dot{\xi}_{i,1} &\approx \dot{\xi}_{i,0} - \mathcal{O}_i^{+}\epsilon\dot{\eta}_{i,0} \\ \dot{\eta}_{i,1} &\approx \mathcal{O}_i^{+}\epsilon\dot{\xi}_{i,0} + \dot{\eta}_{i,0} \end{aligned} \tag{4.41}$$

This yields two options for approximating the transformed turn rate $\mathcal{O}_i^{+}$, given in equation 4.42. As long as at least one of the velocity components is not zero, one of these equations can be used to calculate the transformed turn rate.

$$\begin{aligned} \mathcal{O}_i^{+} &\approx \frac{\dot{\xi}_{i,0} - \dot{\xi}_{i,1}}{\epsilon\dot{\eta}_{i,0}} \\ &\approx \frac{\dot{\eta}_{i,1} - \dot{\eta}_{i,0}}{\epsilon\dot{\xi}_{i,0}} \end{aligned} \tag{4.42}$$

The full state transformed sigma points for the NCT model are given in equation 4.43.

$$\mathcal{X}_i^{+} = \begin{bmatrix} \mathcal{P}_{i,0}^{+} \\ \mathcal{V}_{i,0}^{+} \\ \mathcal{O}_i^{+} \end{bmatrix} \tag{4.43}$$

The sigma point transformation function used in the UT for the NCA model is then defined (according to equations 4.33 through 4.43) in equation 4.44.

$$\mathcal{X}_i^{+} = f_{NCT}(\mathcal{X}_i) \tag{4.44}$$

Notice however that if both velocity components are zero, then both of the above equations for the turn rate $\mathcal{O}_i^{+}$ (equation 4.42) are undefined. If the forward direction is unknown, there is ambiguity in the transformed turn rate even if the turn rate is known prior to transformation. Different assumed forward directions result in different transformed turn rates, leaving the problem ill-defined. Zero-velocity targets can be encountered in practical scenarios if a target is actually stationary or if a one-point track initialization is used. Depending on the origin of the zero-velocity target, different approaches can be used to address this.

For a target that is stationary, it may be possible to use the forward direction vector, if available, to transform the turn rate. Some video detectors are able to determine the position of the object being tracked [113]. If the forward direction vector $\mathcal{D}_f$ is known, then the turn rate can be transformed following a similar approach to transformation of the velocity components. The turn is encoded by three position points which are the position of the object $\mathcal{P}_{i,0}$, a point along the line of the forward direction (pre-turn) $\mathcal{D}_f$, and a point along the line of the forward direction (post-turn) $\mathcal{D}_t$. These points are related to $\mathcal{L}_f$ and $\mathcal{L}_t$ in equation 4.45.

$$\begin{aligned}\mathcal{L}_f &= \mathcal{P}_{i,0} + \mathcal{D}_f \\ \mathcal{L}_t &= \mathcal{P}_{i,0} + \begin{bmatrix} cos(\mathcal{O}_i) & -sin(\mathcal{O}_i) \\ sin(\mathcal{O}_i) & cos(\mathcal{O}_i) \end{bmatrix} \mathcal{D}_f \\ &= \mathcal{P}_{i,0} + \mathcal{D}_t\end{aligned} \tag{4.45}$$

The encoded points are transformed using the homography transform (equation 4.27), and the transformed pre-turn and post-turn forward direction vectors are computed as in equation 4.46.

$$\begin{aligned}\mathcal{D}_f^+ &= \mathcal{L}_f^+ - \mathcal{P}_{i,0}^+ \\ \mathcal{D}_t^+ &= \mathcal{L}_t^+ - \mathcal{P}_{i,0}^+\end{aligned} \tag{4.46}$$

The transformed turn rate is the angle between the two transformed directional vectors, as in equation 4.47. Note that this approach is not strictly in agreement with the NCT model, where the turn rate is applied to the *velocity* of the object rather than a forward facing direction. The validity of its application depends on the actual target kinematics, and whether turn rate is a meaningful concept for a stationary target.

$$\mathcal{O}_i^+ = arccos\left(\frac{\mathcal{D}_f^+ \cdot \mathcal{D}_t^+}{|\mathcal{D}_f^+||\mathcal{D}_t^+|}\right) \tag{4.47}$$

In the scenario where zero-velocity targets occur due to a one-point track initialization, the velocity and the turn rate will be initialized to zero even for moving targets. When the turn rate is effectively zero, and the limiting form of the kinematic model (equation 4.20) is used, this implies that the target is travelling in a straight line. Since the homography transformation maps lines to lines, the transformed trajectory will also be a straight line. Therefore the position and velocity components can be computed as in the NCV model, and the turn rate remains zero after transformation. This does not require a non-zero target velocity.

It is also possible that none of these scenarios apply, and that a zero-velocity target exists with an unknown forward direction and a non-zero turn rate. According to the definition of the NCT model, the turn rate is applied to the velocity, so a stationary target could be given an arbitrary turn rate without affecting other state variables. As such, the NCT model is not appropriate for use in the case of a stationary target. However if the model is not explicitly known, for example when using an interacting multiple model (IMM) filter, it may be desirable to have reasonable values to use even when the model is not a good fit. In such a case, it could be assumed that the turn rate is not significantly affected by the homography transformation and assign the pre-transformation turn rate. Alternatively, it could be assumed that if the target is not moving then it is also not turning, and assign a turn rate of zero.

4.4 Simulations

A simulated scenario is considered to evaluate the performance of the proposed approach using Monte Carlo simulation. The scenario consists of three targets, each moving with one of the motion models described in section 4.2.2, observed by a single imaging sensor mounted on a moving platform. The sensor platform is moving with a constant velocity model in three dimensions, and is additionally subject to zero-mean white Gaussian process noise in its orientation angles (pitch, yaw, roll). The sensor measures the positions of the targets in pixel coordinates, subject to zero-mean white Gaussian measurement noise. Uniformly-distributed false detections are present, and random missed detections occur. The parameters used in the simulations are given in table 4.1.

Simulation Parameters	
Duration	20 seconds
Number of Monte Carlo Runs	200
Sensor Parameters	
Sensor Resolution	1920 x 1080
Sensor FOV	100 degrees x 70 degrees
Measurement Standard Deviation	4 pixels
Probability of Detection	0.9
False Alarm Density	10^{-6}
Motion Model Parameters	
Target 1 σ_{NCV}	0.5
Target 2 σ_{NCA}	0.0071
Target 3 σ_{NCT}	0.0710
Sensor σ_{NCV}	1.0
Sensor $\sigma_{yaw,pitch,roll}$	7.0 degrees

TABLE 4.1: Simulation, sensor, and motion model parameters

All of the targets are tracked using an IMM estimator with a mode for each motion model. This corresponds to an assumption that the motion model of each target is not known, but the family of motion models that can occur is known. Data association and track management is handled using MHT.

4.4.1 Results

The proposed approach described in section 4.3 (labelled UHT) is compared against the existing approach from [102] described in section 4.2.1 (labelled WB), as well as against tracking with no sensor motion compensation (labelled NC). Performance is evaluated for each method using the root-mean-square error (RMSE), number of false tracks (NFT), track continuity, and computation time metrics. RMSE is evaluated for target position, speed, and course.

Several scenarios are considered with different sensor frame rates. Frame rates of 1, 5, and 10 frames per second (FPS) are considered. Under some operational conditions, such as limitations in computation resources, large number of targets or computation intensive detection process, the detection and tracking process can be computationally limiting, even if the imaging sensor itself can provide a high frame rate. Therefore, these values for frame rate are chosen to represent different operational scenarios that might be encountered in practice.

The simulation results for the scenario with a frame rate of 1 FPS are summarized in Table 4.2. The values in this table are averaged over all frames during the simulation period (and over all Monte Carlo runs). The best approach in each metric is highlighted in bold, and the rightmost column (ratio) shows the ratio of the best performing approach to the second best approach. This represents the magnitude of the improvement.

As expected, the NC method performs worst in all metrics, as it does not compensate for sensor motion between frames. This demonstrates the importance of compensating for non-negligible sensor motion between frames.

In most metrics, the proposed UHT method performs either better than all others or similar to the WB method. Most notably, the UHT reduced the overall number of false tracks in this scenario by a substantial 56% when compared with WB. In terms of track accuracy (RMSE), it can be seen that WB is generally sufficient for targets moving with an NCV model, but for targets moving with NCA and NCT models the UHT shows notable improvements in at least some state components. This is expected since the WB method transforms the position and velocity components used in NCV but does not address additional state components used in NCA and NCT. Track continuity is similar between the UHT and WB methods, except for the NCT target where continuity is improved with the UHT. It seems that the tracker is able to maintain continuous tracks for NCV and NCA targets while transforming only the position and velocity components. TPD is the same for WB and UHT. The WB method is better on some metrics than the proposed UHT method, but in nearly all cases the magnitude of the difference is $\leq 2\%$ when WB performs best. The only exception to this is in the course RMSE for the NCA target (where WB is 7% better than UHT), but considering that the speed RMSE for

	NC	WB	UHT	Ratio
Number of False Tracks	0.67	0.18	**0.08**	0.44
Position RMSE NCV (pix)	12.94	**4.99**	5.00	1.00
Position RMSE NCA (pix)	13.17	**4.39**	4.45	0.99
Position RMSE NCT (pix)	12.58	6.52	**6.03**	0.93
Speed RMSE NCV (pix/s)	69.07	6.78	**5.92**	0.87
Speed RMSE NCA (pix/s)	87.11	9.66	**5.30**	0.55
Speed RMSE NCT (pix/s)	85.66	22.70	**7.67**	0.34
Course RMSE NCV (deg)	90.66	**26.91**	**26.91**	1.00
Course RMSE NCA (deg)	97.70	**17.96**	19.25	0.93
Course RMSE NCT (deg)	92.65	42.35	**37.77**	0.89
Continuity NCV	0.12	0.45	**0.46**	1.02
Continuity NCA	0.11	**0.68**	0.67	1.01
Continuity NCT	0.13	0.59	**0.67**	1.14
TPD NCV	0.76	**0.91**	**0.91**	1.00
TPD NCA	0.75	**0.90**	**0.90**	1.00
TPD NCT	0.76	**0.91**	**0.91**	1.00

TABLE 4.2: Summary of simulation results for each approach averaged over time with a frame rate of 1 FPS.

the same target is 45% worse than UHT it seems that the UHT still provides the best velocity estimate.

Breaking down the results for RMSE and NFT over time allows for a detailed look at the differences between sensor motion compensation approaches. Figure 4.2 shows the track component RMSE metrics for each compared approach, over time, for each target. The NC method consistently performs the worst, with WB and UHT performing similarly in most circumstances. With the NCT target, the UHT performs better. This aligns with the observations made when inspecting the results summary in Table 4.2.

Figure 4.3 shows the average NFT over time for each of the compared approaches. As expected the NC approach again performs substantially worse than others. Substantially fewer false tracks are consistently seen with the proposed UHT than with the WB approach from the existing literature. This appears to be the most notable benefit of the proposed UHT.

The next scenario compares the various approaches when using a sensor with a frame rate of 5 FPS. The result of these simulations are summarized in table 4.3. Similar trends are seen in this scenario as in the scenario with the frame rate of 1 FPS, but the magnitude of the improvements is somewhat diminished. While NC is still considerably worse, UHT and WB are more similar in terms of track accuracy metrics and the same in terms of track continuity. TPD for all three approaches is now the same. This diminished difference is expected since with a higher sensor frame rate there is less sensor motion between frames, as well as less target motion. Accordingly, the accuracy of both sensor motion compensation and target motion models is less critical to the overall tracking performance. Despite this, UHT still reduced NFT by a substantial 30% when compared

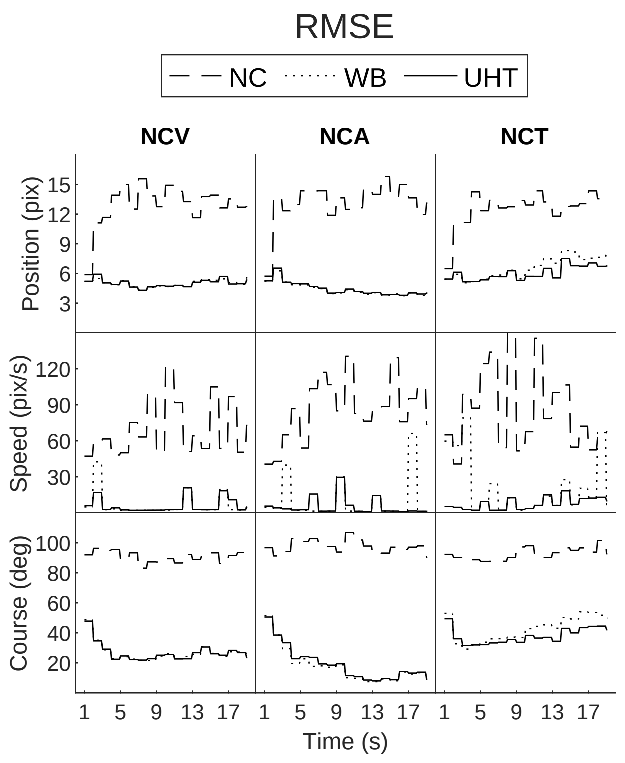

FIGURE 4.2: Root-Mean-Square Error (RMSE) of Position, Speed, and Course for each target over time for each compared approach, with a frame rate of 1 FPS.

with WB, although the NFT was much lower for all approaches in this scenario than in the 1 FPS scenario.

The RMSE metrics broken down over time for the 5 FPS scenario are shown in figure 4.4. Though there are few moments where UHT is noticably more accurate than WB, their accuracy is largely similar. The difference is most prominent for the NCT target, but the magnitude of the difference is diminished compared to the 1 FPS scenario.

Figure 4.5 shows the average NFT over time for each approach in the 5 FPS scenario. Due to the higher data rate and lower NFT among all approaches, the advantages of the UHT are less visible in this figure. Nonetheless, the UHT can be seen to have the lowest average NFT most of the time.

Lastly, a scenario with a sensor frame rate of 10 FPS is considered. The performance evaluation results from this scenario are summarized in table 4.4. Continuing the trend observed when comparing the 5 FPS scenario to the 1 FPS scenario, the differences between the approaches are further diminished for the 10 FPS scenario when compared to the 5 FPS scenario. As the frame rate is increased, the performance of all three

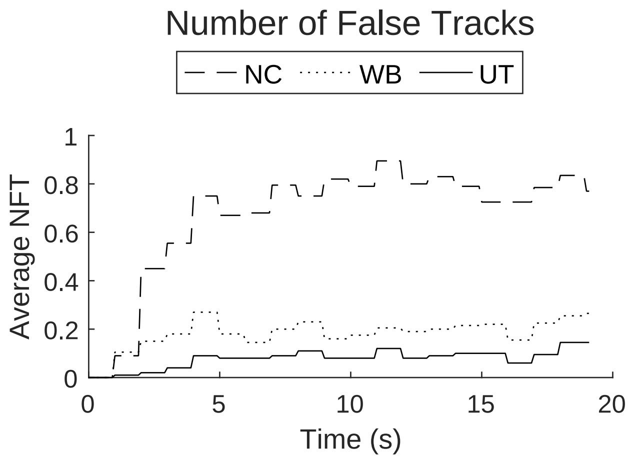

FIGURE 4.3: Average Number of False Tracks (NFT) over time for each compared approach, with a frame rate of 1 FPS.

approaches becomes more similar. With a frame rate of 10 FPS the accuracy of the NC approach remains inferior to others, but track continuity and NFT are now largely similar. As in the 5 FPS scenario, TPD is the same for all approaches.

When examining the RMSE metrics over time (figure 4.6) the same observation can be made. Accuracy of WB and UHT are now virtually the same, with the gap to the less accurate NC narrowing when compared to scenarios with lower frame rates. Also interesting is that accuracy metrics are converging among the three targets. With a frame rate of 10 FPS and the specified target and sensor motion dynamics (see table 4.1), the transformation of the position and velocity components only (as in the WB approach) is largely sufficient to compensate for sensor motion regardless of the target motion model. With a very high frame rate the sensor motion between frames would become negligible, but this may not be practical to achieve.

Figure 4.7 shows the average NFT over time for the scenario with a sensor frame rate of 10 FPS. Although UT performs slightly better in some isolated moments, most of the time all approaches are performing similarly.

The computation time for each approach was also recorded, with similar results for each scenario. Average (mean) computation time per frame for each approach is shown in figure 4.8 for the 10 FPS scenario (due to the similarity of results, computation times for the other two scenarios are omitted for brevity). This demonstrates that computation time is the cost of using the proposed approach, with the UHT taking just under 21% longer to run than WB, while WB took only just over 1% longer to run than NC. An increase in runtime for UHT when compared with WB is expected since the UHT must consider all state components when transforming the tracks. While WB transforms only a state vector with 4 elements and a 2-by-2 sub-matrix of the covariance matrix

	NC	WB	UHT	Ratio
Number of False Tracks	0.0130	0.0123	**0.0086**	0.70
Position RMSE NCV (pix)	7.21	**3.25**	3.27	0.99
Position RMSE NCA (pix)	7.33	**3.28**	3.30	0.99
Position RMSE NCT (pix)	7.38	3.76	**3.58**	0.95
Speed RMSE NCV (pix/s)	17.98	4.84	**4.41**	0.91
Speed RMSE NCA (pix/s)	19.28	5.42	**4.89**	0.90
Speed RMSE NCT (pix/s)	17.35	5.62	**5.00**	0.89
Course RMSE NCV (deg)	76.81	**35.59**	36.14	0.98
Course RMSE NCA (deg)	82.16	**41.14**	41.87	0.98
Course RMSE NCT (deg)	77.47	37.37	**36.63**	0.98
Continuity NCV	0.27	**0.32**	**0.32**	1.00
Continuity NCA	0.32	**0.39**	**0.39**	1.00
Continuity NCT	0.30	**0.38**	**0.38**	1.00
TPD NCV	**0.96**	**0.96**	**0.96**	1.00
TPD NCA	**0.96**	**0.96**	**0.96**	1.00
TPD NCT	**0.96**	**0.96**	**0.96**	1.00

Table 4.3: Summary of simulation results for each approach averaged over time with a frame rate of 5 FPS.

(regardless of the motion model), the UHT transforms up to a 6-by-6 covariance matrix (in the case of the NCA motion model). Note that the UHT implementation is not highly optimized for computation speed, so it is likely that this gap could be narrowed with further effort. In any case, the UHT remains computationally feasible, but there is some increased computational cost when compared with WB and NC.

4.5 Conclusion

In this chaper the unscented homography transform (UHT) is proposed to address the challenges of uncertainty and non-position state components in the traditional homography transform by transforming the entire target state and covariance matrix. This was accomplished by using the unscented transform to handle the estimate uncertainty, reducing the problem to the derivation of a sigma point transformation function that can be applied to the full target state vector.

The performance of the proposed UHT is evaluated through simulations and compared with the existing extension (WB) and a low baseline tracker that neglects sensor motion (NC). The scenario consisted of targets moving with three motion models, and considered different sensor frame rates. The UHT reduced false tracks and improved NCT motion model track accuracy, when compared with WB. Across other metrics the performance of UHT and WB where comparable. The NC tracker performed worse or similar in all metrics. As the frame rate increased the magnitude of the performance differences between all three approaches diminished.

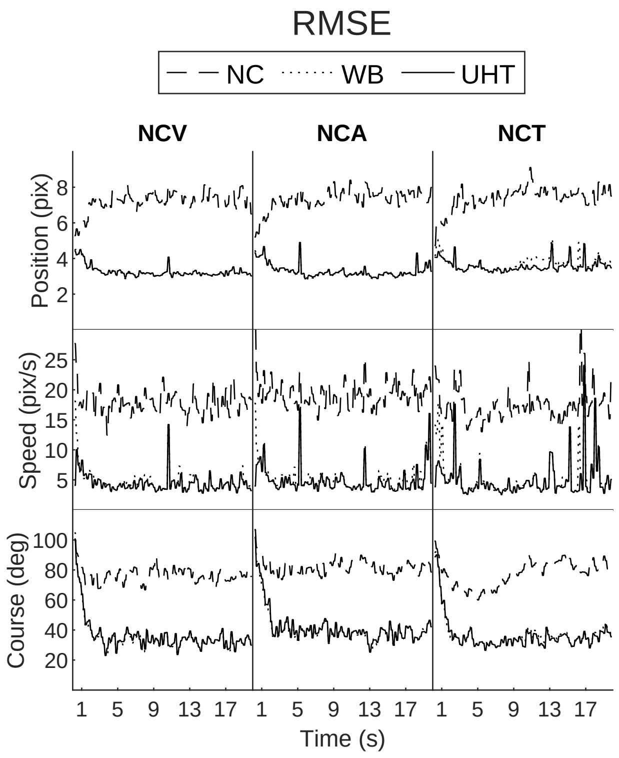

FIGURE 4.4: Root-Mean-Square Error (RMSE) of Position, Speed, and Course for each target over time for each compared approach, with a frame rate of 5 FPS.

The proposed UHT offers an effective and flexible way to transform full target state estimates in pixel coordinates between different points of view, as long as a valid homography can be obtained. A remaining limitation of the proposed approach is that the targets must be approximately co-planar. Future work may address this by reconciling the proposed approach with existing work using multiple consistent homography matrices to transform points in different planes.

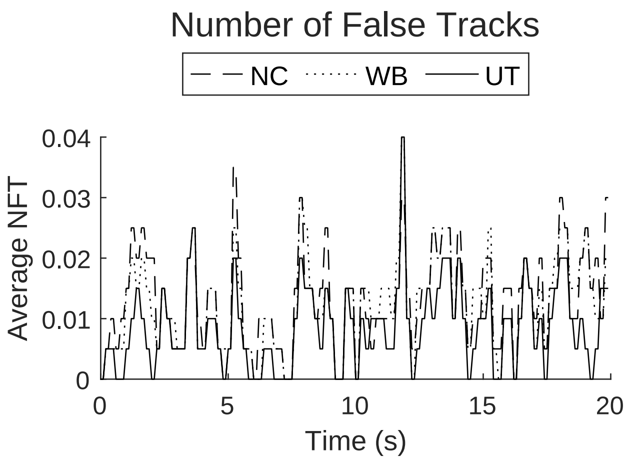

FIGURE 4.5: Average Number of False Tracks (NFT) over time for each compared approach, with a frame rate of 5 FPS.

	NC	WB	UHT	Ratio
Number of False Tracks	0.0078	0.0079	**0.0075**	0.95
Position RMSE NCV (pix)	5.13	**2.88**	2.90	0.99
Position RMSE NCA (pix)	5.20	**2.93**	2.96	0.99
Position RMSE NCT (pix)	5.24	3.06	**2.99**	0.98
Speed RMSE NCV (pix/s)	13.08	6.00	**5.70**	0.95
Speed RMSE NCA (pix/s)	13.69	6.26	**5.88**	0.94
Speed RMSE NCT (pix/s)	11.87	4.98	**4.70**	0.94
Course RMSE NCV (deg)	72.26	**46.98**	47.32	0.99
Course RMSE NCA (deg)	76.29	**54.69**	55.12	0.99
Course RMSE NCT (deg)	73.07	43.67	**43.36**	0.99
Continuity NCV	0.22	**0.23**	**0.23**	1.00
Continuity NCA	**0.24**	**0.24**	**0.24**	1.00
Continuity NCT	0.25	**0.26**	**0.26**	1.00
TPD NCV	**0.98**	**0.98**	**0.98**	1.00
TPD NCA	**0.98**	**0.98**	**0.98**	1.00
TPD NCT	**0.98**	**0.98**	**0.98**	1.00

TABLE 4.4: Summary of simulation results for each approach averaged over time with a frame rate of 10 FPS.

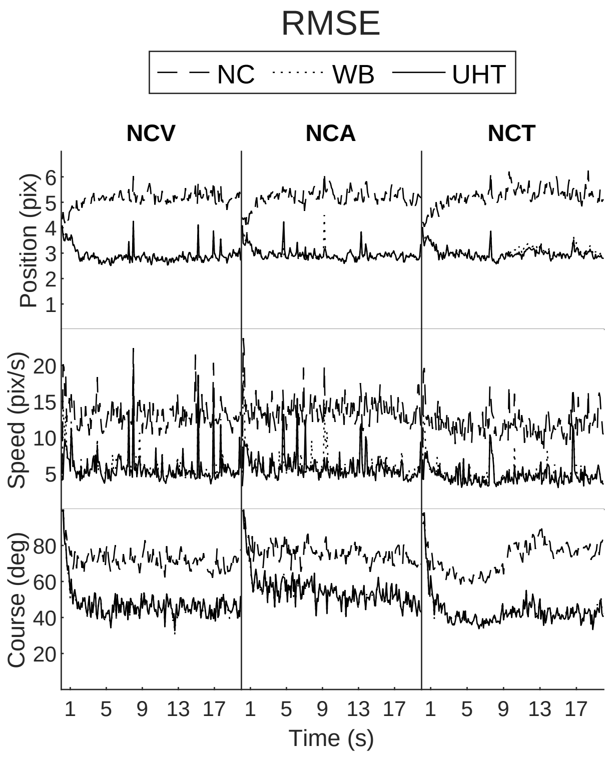

FIGURE 4.6: Root-Mean-Square Error (RMSE) of Position, Speed, and Course for each target over time for each compared approach, with a frame rate of 10 FPS.

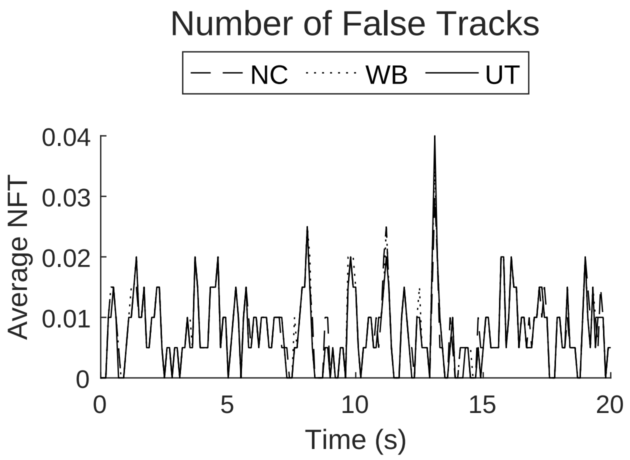

FIGURE 4.7: Average Number of False Tracks (NFT) over time for each compared approach, with a frame rate of 10 FPS.

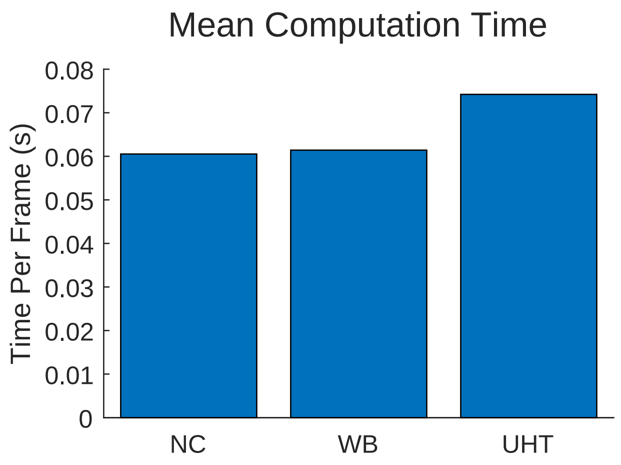

FIGURE 4.8: Average (mean) computation time per frame for each compared approach, recorded for the scenario with a frame rate of 10 FPS.

Chapter 5

Conclusion

The objective of this thesis was to address practical problems in tracking that limit the effectiveness of existing methods or the scope of their application. This has been achieved by developing approaches that relax assumptions about the both the tracking systems and operational conditions in which they will be applied. Improved methods were proposed in terrain-aided tracking, sensor bias estimation and measurement compensation, and sensor motion compensation for tracking in pixel coordinates. The performance of each proposed approach was evaluated through simulation and compared with existing state of the art methods.

In terrain-aided tracking the approach proposed in this thesis relaxed the assumption of a Gaussian distribution on the converted measurement, instead using the more general Gaussian Mixture distribution to model the converted measurement. The Gaussian Mixture distribution was parameterized using image processing techniques to cluster samples from the converted target position distribution and estimate the number of mixture components. The different algorithm variations used either the cluster sample mean and covariance or Expectation Maximization to compute the parameters of each mixture component. Due to the algorithm's ability to dynamically estimate the number of mixture components, the Gaussian Mixture will automatically fall back to being equivalent with the Gaussian assumption (using a single mixture component) when the flat earth assumption applies. When compared in a simulated scenario with an existing approach from the literature that models the converted measurement using a Gaussian distribution, the proposed approach resulted in improved track accuracy for targets tracked in peaky terrain without any degradation of performance in flat terrain. This allows for more robust tracking performance when terrain is peaky or the grazing angle to targets is low.

The approach proposed in this thesis to estimate and compensate for sensor biases does not rely on the assumptions of particular sensor or bias models, homogeneous sensors, number of sensors, fusion architecture, high communication rate, full communication availability, or synchronous communication. It also has a reasonable computational cost, growing only linearly with the number of sensors and again linearly with the number of targets. This makes its scope of practical application much wider than existing approaches in the literature, which are either strictly limited to specific use cases or have a computational load which is not practical under many conditions. This was achieved by restructuring the estimation problem to avoid sources of inconsistency, and by using the flexible Unscented Transform and Covariance Intersection tools. In a simulated scenario using a terrain-aided measurement model (that could not be handled by existing bias estimation algorithms) the proposed approach for bias estimation and compensation improved track accuracy to levels comparable to a tracker operating on unbiased measurements. This was also found to be true for a scenario with limited communication, demonstrating the proposed algorithm's ability to operate effectively without full communication. The tracker operating on unbiased measurements was used as a high baseline due to the lack of any existing approaches that could handle the relevant operational conditions. The proposed approach offers an option for estimating and compensating for sensor biases in scenarios where there previously was none, and substantially eases some challenges of integration for many practical, complex tracking systems.

The extension to the homography transformation proposed for sensor motion compensation when tracking in pixel coordinates can be applied to three common kinematic motion models to fully transform the target state estimate between frames. In addition to these kinematic models, for which derivations are provided, the general approach is designed to be readily extensible to other models. This was achieved in two steps. First, the unscented transform was used to handle uncertainty, reducing the problem of transforming the distribution of the state estimate to that of transforming a set of realizations of the state vectors. Second, the state vector realizations were transformed by encoding them as sets of position points, transforming the position points using the standard homography transformation, and then decoding them into realizations of the transformed state. The performance of the proposed approach was compared in a simulated scenario with an existing approach which only partially transforms the state estimates. Transforming the full state resulted in improved tracking performance if the target motion and sensor motion between frames were substantial, when compared with partial state transformation. The most notable improvements were in the reduced number of false tracks and improved track accuracy and continuity for a target moving with a nearly coordinated turn model when the tracking frame rate was low. This allows for more robust tracking performance with target motion models other than NCV, lower frame rates, and greater sensor motion between frames.

5.1 Future Work

One remaining challenge is the consistent handling of errors in terrain data in the context of terrain-aided tracking. The algorithm in chapter 2 addressed sensor measurement errors, but the error in the terrain data was modelled by adding to the vertical component of the converted measurement noise and its correlation over time was not addressed. Chapter 3 delved further into this issue, investigating the possibility of assuming unknown correlation of measurements over time to produce consistent tracks, but this consistency was achieved at the cost of reduced track accuracy. Future work may investigate the possibility of achieving a more optimal balance between consistency and accuracy when tracking with correlated measurements. An alternative route may be to extend the approach presented in chapter 3 to model the terrain error as a location-dependent bias for each target and estimate this bias to correct for it. A similar approach could be investigated in situations where the flat earth assumption is a good approximation, but still results in some residual modelling error.

Another remaining challenge is improved fusion under unknown correlation with unequal state vectors. This has been applied to sensor bias estimation in chapter 3 as a means of improving estimates of the uncommon state components (the bias estimates) by using information about the common state components (the target state). This chapter proposed one approach that was used effectively, but it is not optimal nor guaranteed to be consistent. Further work on this problem is needed to address these challenges.

The unscented homography transform was proposed in chapter 4 to effectively compensate for sensor motion when tracking co-planar targets in pixel coordinates using a single sensor, but it may be possible to extend this to use cases including multiple sensors and targets that are not co-planar. Future work may investigate using the proposed transform to align tracks between two or more imaging sensors in track-to-track fusion applications. To accommodate targets that do not all lie within the same plane, future work may seek to reconcile the proposed transformation with existing work in multiple homography estimation.

Bibliography

[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

[2] D. Schonborn, T. Kirubarajan, R. Tharmarasa, M. McDonald, and M. Bradford, "Terrain-aided tracking using a gaussian mixture measurement model," *To be submitted to IEEE Transactions on Aerospace and Electronic Systems (July 2022)*, 2022.

[3] D. Schonborn, T. Kirubarajan, and R. Tharmarasa, "Sensor bias estimation and measurement compensation for multisensor-multitarget tracking," *To be submitted to IEEE Transactions on Aerospace and Electronic Systems (July 2022)*, 2022.

[4] ——, "Unscented homography transform for low frame rate tracking in pixel coordinates," *To be submitted to IEEE Transactions on Aerospace and Electronic Systems (July 2022)*, 2022.

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

[6] 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.

[7] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt *et al.*, "A system for video surveillance and monitoring," *VSAM final report*, pp. 1–68, 2000.

[8] M. Mallick, "Geolocation using video sensor measurements," in *2007 10th International Conference on Information Fusion.* IEEE, 2007, pp. 1–8.

[9] V. Karimi, R. Mohseni, and M. R. Khosravi, "An edge computing framework based on ofdm radar for low grazing angle target tracking," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–9, 2020.

[10] G. Rongier, C. Rude, T. Herring, and V. Pankratius, "Generative modeling of insar interferograms," *Earth and Space Science*, vol. 6, no. 12, pp. 2671–2683, 2019.

[11] 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

[12] 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/

[13] 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.

[14] A. M. Fosbury, J. L. Crassidis, T. Singh, and C. Springen, "Ground target tracking using terrain information," in *2007 10th International Conference on Information Fusion*. IEEE, 2007, pp. 1–8.

[15] J. Lancaster, S. Blackman, and L. Yu, "Imm/mht tracking with an unscented particle filter with application to ground targets," in *Signal and Data Processing of Small Targets 2007*, vol. 6699. International Society for Optics and Photonics, September 2007, p. 669919.

[16] B. Ristic, S. Arulampalam, and N. Gordon, "Beyond the kalman filter: Particle filters for tracking applications." Artech house, 2003, ch. 10, pp. 215–238.

[17] H. Sidenbladh and S.-L. Wirkander, "Tracking random sets of vehicles in terrain," in *2003 Conference on Computer Vision and Pattern Recognition Workshop*, vol. 9. IEEE, 2003, pp. 98–98.

[18] R. Collins, Y. Tsin, J. R. Miller, and A. Lipton, "Using a dem to determine geospatial object trajectories," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-98-19, January 1998.

[19] Y. Kim and H. Bang, "Utilization of terrain elevation map in slam for unmanned aircraft," in *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*. IEEE, 2015, pp. 57–62.

[20] R. E. Wolfe, M. Nishihama, A. J. Fleig, J. A. Kuyper, D. P. Roy, J. C. Storey, and F. S. Patt, "Achieving sub-pixel geolocation accuracy in support of modis land science," *Remote Sensing of Environment*, vol. 83, no. 1-2, pp. 31–49, 2002.

[21] A. J. Davison, "Mobile robot navigation using active vision," Ph.D. dissertation, Oxford University, 1998.

[22] C. Xu, D. Huang, and J. Liu, "Target location of unmanned aerial vehicles based on the electro-optical stabilization and tracking platform," *Measurement*, vol. 147, p. 106848, 2019.

[23] B. Tufan and A. Bayri, "Emitter localization based on angle of arrival measurements using digital terrain elevation data," in *2012 20th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2012, pp. 1–4.

[24] K. Dästner, B. Köhler, and F. Opitz, "Fusion of ir/ccd video streams and digital terrain models for multi target tracking." in *GI Jahrestagung*, 2009, pp. 2334–2342.

[25] M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density," in *European conference on computer vision*. Springer, 1996, pp. 343–356.

[26] 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.

[27] Q. D. Team. (2019) Qgis geographic information system, open source geospatial foundation project. [Online]. Available: http://qgis.osgeo.org

[28] J. Li, G. Taylor, and D. B. Kidner, "Accuracy and reliability of map-matched gps coordinates: The dependence on terrain model resolution and interpolation algorithm," *Computers & Geosciences*, vol. 31, no. 2, pp. 241–251, 2005.

[29] 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.

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

[31] C. K. Reddy, H.-D. Chiang, and B. Rajaratnam, "Trust-tech-based expectation maximization for learning finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 7, pp. 1146–1157, 2008.

[32] B. Umale and M. Nilav, "Overview of k-means and expectation maximization algorithm for document clustering," *International Journal of Computer Applications*, vol. 63, no. 13, pp. 578–588, 2014.

[33] G. Hamerly and C. Elkan, "Learning the k in k-means," in *Proceedings of the 16th International Conference on Neural Information Processing Systems*, ser. NIPS'03. Cambridge, MA, USA: MIT Press, 2003, p. 281–288.

[34] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[35] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4696–4704.

[36] K. Li, B. Habtemariam, R. Tharmarasa, M. Pelletier, and T. Kirubarjan, "Multitarget tracking with doppler ambiguity," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 4, pp. 2640–2656, 2013.

[37] T. Yamada, T. Ito, H. Izumi, S. Murayama, Y. Sawayama, and Y. Obata, "Multidimensional multiple hypothesis tracking with a gaussian mixture model to suppress grating lobes," in *2017 20th International Conference on Information Fusion (Fusion)*. IEEE, 2017, pp. 1–8.

[38] P. R. Östergård, "A new algorithm for the maximum-weight clique problem," *Nordic Journal of Computing*, vol. 8, no. 4, pp. 424–436, 2001.

[39] A. Ravankar, Y. Kobayashi, A. Ravankar, and T. Emaru, "A connected component labeling algorithm for sparse lidar data segmentation," in *2015 6th International Conference on Automation, Robotics and Applications (ICARA).* IEEE, 2015, pp. 437–442.

[40] L. He, Y. Chao, K. Suzuki, and K. Wu, "Fast connected-component labeling," *Pattern recognition*, vol. 42, no. 9, pp. 1977–1987, 2009.

[41] S. Liu, H. Li, Y. Zhang, and B. Zou, "Multiple hypothesis method for tracking move-stop-move target," *The Journal of Engineering*, vol. 2019, no. 19, pp. 6155–6159, 2019.

[42] C. Sanderson and R. Curtin, "An open source c++ implementation of multi-threaded gaussian mixture models, k-means and expectation maximisation," in *2017 11th International Conference on Signal Processing and Communication Systems (ICSPCS).* IEEE, 2017, pp. 1–8.

[43] 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.

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

[45] 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.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

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

[59] 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.

[60] 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.

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

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

[63] 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.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

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

[69] 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.

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

[71] 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.

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

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] 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.

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

[83] 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.

[84] 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.

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

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

[87] Y. Watanabe, P. Fabiani, and G. Le Besnerais, "Simultaneous visual target tracking and navigation in a gps-denied environment," in *2009 International Conference on Advanced Robotics.* IEEE, 2009, pp. 1–6.

[88] P. Chen, Y. Dang, R. Liang, W. Zhu, and X. He, "Real-time object tracking on a drone with multi-inertial sensing data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 131–139, 2017.

[89] F. S. Leira, T. A. Johansen, and T. I. Fossen, "Automatic detection, classification and tracking of objects in the ocean surface from uavs using a thermal camera," in *2015 IEEE aerospace conference.* IEEE, 2015, pp. 1–10.

[90] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "Uav cinematography constraints imposed by visual target tracking," in *2018 25th IEEE International Conference on Image Processing (ICIP).* IEEE, 2018, pp. 76–80.

[91] P. Rosello and M. J. Kochenderfer, "Multi-agent reinforcement learning for multi-object tracking," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 1397–1404.

[92] S. Li and D.-Y. Yeung, "Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[93] A. Desai, D.-J. Lee, and M. Zhang, "Using accurate feature matching for unmanned aerial vehicle ground object tracking," in *International Symposium on Visual Computing.* Springer, 2014, pp. 435–444.

[94] K. M. Abughalieh, B. H. Sababha, and N. A. Rawashdeh, "A video-based object detection and tracking system for weight sensitive uavs," *Multimedia Tools and Applications*, vol. 78, no. 7, pp. 9149–9167, 2019.

[95] Z. L. Szpak, W. Chojnacki, and A. van den Hengel, "Robust multiple homography estimation: An ill-solved problem," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2132–2141.

[96] W. Chojnacki and Z. L. Szpak, "Full explicit consistency constraints in uncalibrated multiple homography estimation," in *Asian Conference on Computer Vision.* Springer, 2018, pp. 659–675.

[97] J. Pang, L. Qiu, X. Li, H. Chen, Q. Li, T. Darrell, and F. Yu, "Quasi-dense similarity learning for multiple object tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 164–173.

[98] S. Gu, Y. Zheng, and C. Tomasi, "Efficient visual object tracking with online nearest neighbor classifier," in *Asian Conference on Computer Vision.* Springer, 2010, pp. 271–282.

[99] H. Wang, H. S. Stone, and S.-F. Chang, "Facetrack: Tracking and summarizing faces from compressed video," in *Multimedia Storage and Archiving Systems IV*, vol. 3846. SPIE, 1999, pp. 222–234.

[100] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, "Video object tracking using adaptive kalman filter," *Journal of Visual Communication and Image Representation*, vol. 17, no. 6, pp. 1190–1208, 2006.

[101] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[102] J. H. White and R. W. Beard, "The homography as a state transformation between frames in visual multi-target tracking," *Brigham Young University ScholarsArchive*, 2019.

[103] T. Nguyen, S. W. Chen, S. S. Shivakumar, C. J. Taylor, and V. Kumar, "Unsupervised deep homography: A fast and robust homography estimation model," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2346–2353, 2018.

[104] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[105] Y. Kang, C. Roh, S.-B. Suh, and B. Song, "A lidar-based decision-making method for road boundary detection using multiple kalman filters," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4360–4368, 2012.

[106] Z. Xue, Y. Zhang, C. Cheng, and G. Ma, "Remaining useful life prediction of lithium-ion batteries with adaptive unscented kalman filter and optimized support vector regression," *Neurocomputing*, vol. 376, pp. 95–102, 2020.

[107] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[108] T. Kirubarajan, Y. Bar-Shalom, K. R. Pattipati, and I. Kadar, "Ground target tracking with variable structure imm estimator," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 1, pp. 26–46, 2000.

[109] N. Peng, S. Zhang, X. Guo, and X. Zhang, "Online parameters identification and state of charge estimation for lithium-ion batteries using improved adaptive dual unscented kalman filter," *International Journal of Energy Research*, vol. 45, no. 1, pp. 975–990, 2021.

[110] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[111] J. R. Van Zandt, "A more robust unscented transform," in *Signal and Data Processing of Small Targets 2001*, vol. 4473. International Society for Optics and Photonics, 2001, pp. 371–380.

[112] H. M. Menegaz, J. Y. Ishihara, G. A. Borges, and A. N. Vargas, "A systematization of the unscented kalman filter theory," *IEEE Transactions on automatic control*, vol. 60, no. 10, pp. 2583–2598, 2015.

[113] R. Juranek, A. Herout, M. Dubská, and P. Zemcik, “Real-time pose estimation piggybacked on object detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2381–2389.