Cooperative Estimation for a Vision-Based Multiple Target Tracking System

Joshua Y. Sakamaki
Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Electrical and Computer Engineering Commons

BYU ScholarsArchive Citation
Sakamaki, Joshua Y., "Cooperative Estimation for a Vision-Based Multiple Target Tracking System" (2016). All Theses and Dissertations. 6072.
https://scholarsarchive.byu.edu/etd/6072

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Cooperative Estimation for a Vision-Based Multiple Target Tracking System

Joshua Y. Sakamaki

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of
Master of Science

Randal W. Beard, Chair
Michael D. Rice
Timothy W. McLain

Department of Electrical & Computer Engineering
Brigham Young University
June 2016

Copyright © 2016 Joshua Y. Sakamaki
All Rights Reserved
ABSTRACT

Cooperative Estimation for a Vision-Based Multiple Target Tracking System

Joshua Y. Sakamaki
Department of Electrical & Computer Engineering, BYU
Master of Science

In this thesis, the Recursive-Random Sample Consensus (R-RANSAC) algorithm is applied to a vision-based, cooperative target tracking system. Unlike previous applications, which focused on a single camera platform tracking targets in the image frame, this work uses multiple camera platforms to track targets in the inertial or world frame. The process of tracking targets in the inertial frame is commonly referred to as geolocation.

In practical applications sensor biases cause the geolocated target estimates to be biased from truth. The method for cooperative estimation developed in this thesis first estimates the relative rotational and translational biases that exist between tracks from different vehicles. It then accounts for the biases and performs the track-to-track association, which determines if the tracks originate from the same target. The track-to-track association is based on a sliding window approach that accounts for the correlation between tracks sharing common process noise and the correlation in time between individual estimation errors, yielding a chi-squared distribution. Typically, accounting for the correlation in time requires the inversion of a \( Nn_x \times Nn_x \) covariance matrix, where \( N \) is the length of the window and \( n_x \) is the number of states. Note that this inversion must occur every time the track-to-track association is to be performed. However, it is shown that by making a steady-state assumption, the inverse has a simple closed-form solution, requiring the inversion of only two \( n_x \times n_x \) matrices, and can be calculated offline. Distributed data fusion is performed on tracks where the hypothesis test is satisfied. The proposed method is demonstrated on data collected from an actual vision-based tracking system.

A novel method is also developed to cooperatively estimate the location and size of occlusions. This capability is important for future target tracking research involving optimized path planning/gimbal pointing, where a geographical map is unavailable. The method is demonstrated in simulation.

Keywords: cooperative estimation, multiple target tracking, geolocation, recursive RANSAC, bias estimation, track-to-track association, track fusion, occlusion estimation
ACKNOWLEDGMENTS

First and foremost, I would like to thank my parents for supporting me throughout my educational endeavors. They, along with the rest of my family and friends, were always there to offer words of encouragement.

I would like to thank Dr. Randy Beard for being a wonderful advisor and mentor throughout my graduate years. Our discussions have instilled in me the importance of being mathematically rigorous and his counsel has been invaluable to my development as a researcher.

I’d also like to thank the members of the BYU MAGICC Lab for all they’ve done to help support my efforts. Particularly, I’d like to acknowledge Peter Niedfeldt, Kyle Ingersoll, and Patrick DeFranco for their work, which my thesis was able to build upon.

This project was funded by the Air Force Research Laboratory through the grant number FA8651-13-1-0005 with program manager Dr. Will Curtis.
# TABLE OF CONTENTS

**LIST OF FIGURES** ............................... vi

**NOMENCLATURE** ............................... vii

Chapter 1  Introduction ............................ 1

Chapter 2  Background of Methods Used ............... 6
  2.1  Target Model ................................. 6
  2.2  Computer Vision Background ..................... 7
      2.2.1  Foreground Detection ...................... 7
      2.2.2  Camera Model ............................. 9
  2.3  Geolocation ................................. 10
  2.4  Recursive-RANSAC Background ................... 10
      2.4.1  Random Sample Consensus (RANSAC) ........... 10
      2.4.2  Recursive-RANSAC ......................... 11
  2.5  Cooperative Architecture ....................... 13

Chapter 3  Cooperative Estimation .................... 15
  3.1  Track Bias Estimation .......................... 15
      3.1.1  Initialization of Optimizer ................. 17
      3.1.2  Updating the Rotational and Translational Biases 22
  3.2  Track-to-Track Association ...................... 22
      3.2.1  Hypothesis Test ........................... 22
      3.2.2  Covariance Matrix $\mathbf{P}_0$ ................ 25
      3.2.3  Recursion Equation for Test Statistic ........ 33
      3.2.4  Invertibility of $\mathbf{P}_0$ ................ 35
  3.3  Track Fusion ................................ 36
      3.3.1  Updating the Consensus Set After Track Fusion .. 40
  3.4  Conclusion .................................. 40

Chapter 4  Results ................................. 42
  4.1  Simulation Results ............................. 42
  4.2  Video Results ................................ 44
      4.2.1  Test With Stationary Platforms ............... 44
      4.2.2  Test With UAV Platforms ..................... 50
  4.3  Discussion of Biases Over Time .................. 51
  4.4  Conclusion .................................. 53

Chapter 5  Occlusion Estimation ..................... 55
  5.1  Method .................................. 55
  5.2  Simulation Results ............................. 59
  5.3  Conclusion .................................. 60
Chapter 6  Conclusion ......................................................... 63
  6.1  Summary of Contributions ........................................ 63
  6.2  Recommendations for Future Work .............................. 64

REFERENCES ................................................................. 66

Appendix A  Proof of Theorem 1 ......................................... 70

Appendix B  Partial Derivatives for Gradient Descent Method ...... 72

Appendix C  Hardware ...................................................... 75
  C.1  Camera .............................................................. 75
  C.2  UAV Platform ...................................................... 77
# List of Figures

1.1 Possible tracking scenario with multiple targets and occlusions .......................... 2
1.2 Block diagram of the complete vision-based target tracking system for two vehicles 3
2.1 Results of the moving platform foreground detector ........................................... 9
2.2 Illustration of various architectures for multi-vehicle systems ............................ 14
3.1 Illustration of the rotational bias between two tracks ......................................... 21
3.2 The $L_2$ norm of the cross-covariance $P_{ij}$ converges to a steady-state value over time 30
3.3 The test statistic represented at two subsequent time steps ............................... 34
4.1 Example of associated tracks, randomly generated for each simulation .................. 43
4.2 Monte Carlo simulations testing the bias estimation and track-to-track association 44
4.3 Calculated test statistic versus theoretical chi-squared distribution ................. 45
4.4 Tracking scenario with two stationary cameras and two ground targets ............ 46
4.5 Geolocated tracks from test with stationary cameras .................................... 47
4.6 Results of bias estimation with associated tracks ........................................... 47
4.7 Results of bias estimation with unassociated tracks ....................................... 48
4.8 Results of track-to-track association for test with stationary cameras .............. 48
4.9 Results of track fusion .................................................................................. 49
4.10 Track fusion through a simulated occlusion .................................................... 50
4.11 Tracking scenario with two maneuvering cameras and one ground target .......... 51
4.12 Plot of paths taken by maneuvering camera platforms .................................... 51
4.13 Results of track-to-track association for test with maneuvering camera platforms 52
4.14 Translational and rotational biases over time from test with stationary cameras 52
4.15 Illustration of the time-varying measurement noise associated with geolocation applications .............................................................. 53
5.1 The likelihood update region for a given UAV-track pair .................................. 57
5.2 Illustration of the sigmoid function .................................................................. 58
5.3 The line of sight vector for estimating occlusion height .................................... 59
5.4 Simulated scenario for testing occlusion estimation ......................................... 60
5.5 Likelihood map generated by occlusion estimation .......................................... 61
5.6 Binary map generated by occlusion estimation ............................................... 61
C.1 Sample of images used in camera calibration ............................................... 75
C.2 Results of undistortion process ..................................................................... 76
C.3 3DR Y6 multicopter used in actual UAV test ................................................. 77
NOMENCLATURE

$V_i$ The $i^{th}$ vehicle in the network
$x_{ij}^i$ State of true signal from $i^{th}$ vehicle expressed in the frame of the $j^{th}$ vehicle
$\hat{x}_{ij}^i$ State estimate from $i^{th}$ vehicle expressed in the frame of the $j^{th}$ vehicle
$\tilde{x}$ Relative estimation error between two tracks
$\tilde{\tilde{x}}$ Vector of relative estimation errors
$\hat{p}_{ij}^i$ State estimate (2-D position) from $i^{th}$ vehicle expressed in the frame of the $j^{th}$ vehicle
$\tilde{p}$ Relative estimation error between two tracks (position only)
$\tilde{\tilde{p}}$ Vector of relative estimation errors (position only)
$P_i^i$ Estimation error covariance from the $i^{th}$ vehicle expressed in the frame of the $j^{th}$ vehicle
$P_{ij}^i$ Cross-covariance between associated tracks from the $i^{th}$ and $j^{th}$ vehicles
$P_{i,j,\infty}$ Steady-state cross-covariance
$P_{p,\infty}$ Steady-state prediction error covariance
$P_{e,\infty}$ Steady-state estimation error covariance
$P$ Covariance of the relative estimation error
$P_\infty$ Steady-state covariance of the relative estimation error
$n_x$ Dimension of state vector
$p_{n}, p_{e}$ North/east positions
$A$ State transition matrix
$C$ Measurement observation matrix
$y$ Measurement
$Q$ Covariance of the process noise
$\sigma_Q$ Parameter used to scale process noise covariance
$\xi$ Process noise
$R$ Covariance of the measurement noise
$\sigma_R$ Parameter used to scale measurement noise covariance
$\nu$ Measurement noise
$K$ Kalman gain
$K_{\infty}$ Steady-state Kalman gain
$Vol_{ij}^i$ Volume of steady-state estimation error ellipsoid
$Vol_{ij}^i$ Volume of ellipsoid from fused estimates, while accounting for cross-covariance
$\mu$ Scaling between magnitude of estimation error covariance and cross-covariance
$I$ Identity matrix of size $n_x \times n_x$
$\bar{I}$ Identity matrix of size $Nn_x \times Nn_x$
$\bar{0}$ Vector of zeros, of size $n_x \times 1$
$O_x, O_y$ Pixel location of camera optical axis
$\bar{e}_x, \bar{e}_y$ Pixel measurements in standard image reference frame
$e_x, e_y$ Pixel measurements in reference frame used for geolocation
$f$ Camera focal length in pixels
$k$ Intrinsic matrix
$\chi$ Consensus set
$N_w$ Length of measurement history window used in R-RANSAC
$M$ Number of stored models
CHAPTER 1. INTRODUCTION

Unmanned aerial vehicles (UAV) provide unique perspectives that can be of benefit in intelligence, surveillance, search and rescue, and reconnaissance applications. With the recent rise in UAV technology has come an increased focus on robust, vision-based target tracking from UAV platforms. When equipped with a video camera, an unmanned aerial vehicle (UAV) is capable of tracking ground targets from hundreds, and sometimes thousands of meters away. In most applications, an operator specifies which targets are to be tracked. A tracking algorithm, typically referred to as a multiple target tracker (MTT), then attempts to estimate the states of these targets and arranges the state estimates into continuous sequences, which are known as tracks. In an ideal situation, the tracking system is then completely autonomous, requiring no further input from the operator to continuously track the specified targets. However, in practice the robustness of the MTT diminishes as the sensor noise increases, the number of targets increases, the distance between targets decreases, or targets are frequently occluded. These conditions often cause the system to experience track fragmentation, where the tracks becomes discontinuous. When track fragmentation occurs the system stops tracking certain targets, which requires the operator to re-specify these targets. Therefore, the tracking process can be a highly interactive task, requiring monitoring and adjustments from an operator.

Recursive-Random Sample Consensus (R-RANSAC) is a novel multiple target tracking algorithm that was developed at Brigham Young University. Niedfeldt described the main theory of R-RANSAC in [1, 2] and compared it to state-of-the-art MTT algorithms such as nearest neighbor, joint probabilistic data association (JPDA), multiple hypothesis tracking (MHT), and probabilistic hypothesis density filter (PHD). The author was able to demonstrate R-RANSAC’s exceptional tracking performance, especially in environments with cluttered measurements. In [3] Ingersoll made various improvements to R-RANSAC and applied the algorithm to tracking scenarios in-
volving a single, stationary camera. This work was extended by DeFranco to scenarios with a single, maneuvering camera platform [4].

Unlike previous applications, the work in this thesis uses multiple camera platforms to cooperatively track targets. In general, the use of a cooperative system can increase the probability of a given target being seen by at least one UAV, increase the number of targets that can be collectively tracked by the system, and decrease the uncertainty in target states through track fusion. Such a system increases track continuity even in environments with multiple occlusions, which is essential for a fully autonomous tracking system. A possible tracking scenario which could benefit from a multi-vehicle system is presented in Figure 1.1. Additionally, the handoff problem is another tracking scenario which requires cooperative estimation.

It should be noted that the methods developed by Ingersoll and DeFranco perform tracking in the image frame, meaning that the states of each target are expressed in terms of pixels. Target tracking in the image frame has a significant challenge when the camera platform is moving. If we define the target dynamics within the image frame and the camera then moves, the dynamic model of the target changes. This makes it difficult to propagate tracks, which is especially important when targets leave the camera’s field of view. The work in this thesis overcomes this challenge by incorporating a technique called geolocation. In geolocation, all target measurements are trans-

---

1 In the handoff problem, a UAV tracks a target of interest. When the vehicle is low on fuel a second UAV is deployed. For some time the two vehicles simultaneously and cooperatively track the target. The main tracking responsibility is handed off to the second vehicle, at which time the first vehicle leaves.
formed from the dynamic image frame to the static inertial frame, essentially enabling the system to output the targets’ GPS locations. Aside from making the propagation of tracks easier, this technique opens the door to future research with learning algorithms that can be used to predict target behavior. The geolocation technique used in this thesis makes use of a flat-earth assumption and is highlighted in [5] and [6].

A drawback of vision-based geolocation is that it not only requires the pixel measurements of the target locations, but it also requires measurements of the UAV and gimbal states, as seen in the block diagram of Figure 1.2. These measurements are often subject to sensor biases, which can be introduced through systematic errors in the camera calibration, UAV positional/attitude data, or gimbal angles. The sensor biases cause the estimates produced by R-RANSAC to be biased from truth and make it difficult for the system to determine when a target is being simultaneously detected by multiple vehicles, which is a necessary condition for proper track fusion.

Therefore, in Figure 1.2 the tracks from different vehicles are fed into a bias estimation block, where the rotational and translational biases are estimated between the track pairs. Once the biases have been estimated the objective is to determine which tracks from one vehicle correspond to the tracks from the other vehicle. This process is often referred to as track-to-track correlation [7] or track-to-track association [8].

In [9] the authors present a method for track-to-track association that relies on the conditional probability densities to develop assignment cost functions. Optimization techniques are
used to find the associations that minimize the overall cost.References [10–13] use a similar approach to estimate the biases and perform the track-to-track associations simultaneously. However, these methods require complex assignment algorithms that can be computationally intensive, and in many cases are not suitable for real-time applications. Moreover, they only consider translational biases between tracks and do not account for rotational biases, which can also have a large influence on the estimates. In [14] the authors formulate the problem using a method called reference topology, where rotational sensor biases are accounted for. However, this method assumes that the same set of targets are observed by each vehicle. Note that none of the methods mentioned have been tested using a vision-based tracking system.

In [6] and [15] the authors explore methods that estimate and account for sensor biases in a geolocating, vision-based tracking system. However, in both works only one target is considered and the track-to-track association problem is not addressed. Moreover, the bias estimation schemes require that the vehicles fly in specific orbits around the targets, which in practice may not be feasible or ideal. In [16] a vision-based estimation algorithm is presented that enables a system of multiple UAVs to cooperatively geolocate ground targets with high levels of accuracy. However, the authors assume that there are no sensor biases and the track-to-track association problem is not considered.

The organization of this thesis is as follows. In Chapter 2 we present background of the methods used within the current vision-based multiple target tracking system. This includes a discussion of the foreground detection, geolocation process, and the multiple target tracking algorithm R-RANSAC. As listed below, the remainder of the thesis presents novel contributions to the state-of-the-art in cooperative, vision-based multiple target tracking.

1. In Section 3.1 we develop a bias estimation technique that is able to account for both the relative rotational and translational biases that exist between tracks from different vehicles.

2. In Section 3.2 we present a statistical test for determining if two tracks from different vehicles originate from the same target, and derive the necessary covariance matrices under a steady-state assumption. Here we show that such an assumption greatly reduces the computation required in the calculation of the test statistic.
3. In Section 3.3 we derive fusion equations that account for the estimated biases, and present the equations in both state and information space.

4. In Chapter 4 we present the main results of the cooperative estimation methods from both simulated and actual video data. Here we show that the methods are effective in the presence of actual UAV sensor biases and noise.

5. In Chapter 5 we present a novel method for cooperatively estimating the size and location of occlusions in real-time, providing a bridge between the current target tracking capabilities and future work with optimal path planning/gimbal pointing.

Finally, in Chapter 6 we summarize our findings and make recommendations for future research.
CHAPTER 2. BACKGROUND OF METHODS USED

2.1 Target Model

Throughout this thesis it is assumed that the multi-vehicle system is composed of homogeneous trackers, thus each vehicle uses the same state transition matrix $A$, observation matrix $C$, process noise covariance $Q$, and measurement noise covariance $R$ to model the target dynamics. The target states are defined in the north and east directions, given by $x = [p_n, p_e, \dot{p}_n, \dot{p}_e, \ddot{p}_n, \ddot{p}_e]^T$, where $p_n$ and $p_e$ are the north and east positions respectively, and dots above the variables denote differentiation with respect to time. The size of the state vector is therefore $n_x = 6$. The target dynamics are modeled using the linear system, defined as

$$x[t] = Ax[t-1] + \xi[t]$$

$$y[t] = Cx[t] + v[t],$$

where $\xi \sim N(O, Q)$ and $v \sim N(O, R)$.

In this thesis a Weiner process acceleration model is used [17], which defines $A$, $Q$, $C$, and $R$ as

$$A = \begin{pmatrix}
1 & 0 & dt & 0 & \frac{dt^2}{2} & 0 \\
0 & 1 & 0 & dt & 0 & \frac{dt^2}{2} \\
0 & 0 & 1 & 0 & dt & 0 \\
0 & 0 & 0 & 1 & 0 & dt \\
0 & 0 & 0 & 0 & 1 & dt \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

$$Q = \sigma_Q^2 \begin{pmatrix}
\frac{dt^5}{20} & 0 & \frac{dt^4}{8} & 0 & \frac{dt^3}{6} & 0 \\
0 & \frac{dt^5}{20} & 0 & \frac{dt^4}{8} & 0 & \frac{dt^3}{6} \\
\frac{dt^4}{8} & 0 & \frac{dt^3}{3} & 0 & \frac{dt^2}{2} & 0 \\
0 & \frac{dt^4}{8} & 0 & \frac{dt^3}{3} & 0 & \frac{dt^2}{2} \\
\frac{dt^3}{6} & 0 & \frac{dt^2}{2} & 0 & dt & 0 \\
0 & \frac{dt^3}{6} & 0 & \frac{dt^2}{2} & 0 & dt
\end{pmatrix}$$

$$C = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$R = \sigma_R^2 \begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix},$$

where $\sigma_Q$ and $\sigma_R$ are the standard deviations of the process and measurement noise, respectively.
where \( dt \) is size of the time step, \( \sigma^2_Q \) is used to scale \( Q \) to achieve desired tracking performance, and \( \sigma^2_R \) is determined experimentally through sensor calibration.

2.2 Computer Vision Background

2.2.1 Foreground Detection

In vision-based target tracking a video camera is used to provide pixel measurements for the targets, which in this thesis are assumed to be moving. The way in which a vision system extracts pixel locations of these targets is through a foreground detector, which essentially identifies pixels that are associated with moving objects (foreground) and those that are not (background).

There are a variety of methods for foreground detection. One of the simplest is to subtract the current image from an image with an empty background. As targets enter the scene this method is able to fully segment the image and identify the targets as foreground. However, this method is unable to adjust to changes in lighting conditions between the two images, which is likely to occur as time goes on. Moreover, the image of the empty background must be acquired and stored prior to tracking, which is not feasible for many tracking applications.

Another method is frame-to-frame subtraction, where the current image is subtracted from either the previous image frame, or from a frame that is several time steps before the current frame. Since the time difference between the two images is quite small in comparison to the previous method, this foreground detector is better able to adjust to changes in lighting conditions. Also, it does not require any images to be obtained prior to tracking. However, this method struggles to fully segment the foreground at higher frame rates and leaves ghosts of the targets in the image at lower frame rates.

Both of the methods presented assume that the two images being compared are in the same reference frame. This is obviously true when the camera is stationary. On the other hand, a moving camera platform causes the two images to have different coordinate systems. Performing the subtraction on these images would cause a large amount of background pixels to be falsely classified as foreground. To rectify this, the motion of the camera must be accounted for by identifying the transformation between the frames. Finding this transformation is performed by calculating the \textit{homography} between the two image frames.
Homography is a fundamental concept taken from the computer vision community. It represents the projective mapping from one reference frame to another and is referenced in [4, 18, 19]. The first step in calculating the homography between two images is to find feature points within each image. Once the features are found the next step is to match the features across the images. The goal of feature matching is to create feature point pairs by assigning a pixel location from one image, to a corresponding pixel location in the other image. For this thesis the process of finding and matching features is again highlighted in [4], where the author uses Shi and Tomasi’s method [20] to find good features to track, and the Kanade-Lucas-Tomasi algorithm to match features across images [21].

There are a couple things to note about the calculation of the homography. The first is that it assumes a flat-earth model. This assumption often seems somewhat restrictive, however, it was found to be a reasonable assumption for moving-target detection [4]. The second is that it requires at least four feature point pairs, three of which cannot be collinear. As one might imagine, the process of finding and matching features is nontrivial and often inaccurate, which could lead to an erroneous calculation of the homography. One way to deal with the mentioned uncertainty is to calculate the homography using many feature point pairs. Doing so creates an over-constrained problem that yields much better results.

Once the homography has been calculated the older image can be transformed into the current image’s coordinate system. The frame-to-frame subtraction method can then be applied to the two images to detect the foreground pixels. Clusters of foreground pixels, which are often referred to as blobs by those in the computer vision community, can then be identified. Various methods for identifying blobs are presented in [3]. The centroids of these blobs then act as pixel measurements for the targets.

Results of the foreground detector can be seen in Figure 2.1. Here the algorithm detects features within the image, which are depicted as green dots, and tracks the movement of the features across frames, which is shown by the yellow vector lines. This information is used to calculate the homography between the frames. Once the motion of the camera has been accounted for, the location of the targets can be determined, which are displayed by the yellow boxes.
Figure 2.1: Results of the moving platform foreground detector. The green dots are the feature points from the current frame. The yellow vector lines indicate the movement of those feature points from the previous frame. The yellow boxes indicate the target locations.

2.2.2 Camera Model

The pixel measurements produced by the foreground detector \((\bar{\epsilon}_x, \bar{\epsilon}_y)\) are specified in a reference frame with the origin at the top left of the image, the horizontal axis pointing to the right, and the vertical axis pointing down. This convention is widely used within the computer vision community. However, the camera model used in this thesis is illustrated in [5] and requires that the reference frame’s origin be at the pixel location of the optical axis \((O_x, O_y)\). Moreover, it requires that the horizontal axis be specified as pointing to the right, while the vertical axis be specified as pointing up.

The pixel location that is aligned with the optical axis can be estimated through a camera calibration. Note that the camera calibration also yields the camera’s focal length, which is necessary for the geolocation process. The pixel location of each target in the new reference frame are calculated as

\[
\begin{pmatrix}
\bar{\epsilon}_x \\
\bar{\epsilon}_y
\end{pmatrix}
= \begin{pmatrix}
1 & 0 \\
0 & -1
\end{pmatrix}
\begin{pmatrix}
\bar{\epsilon}_x - O_x \\
\bar{\epsilon}_y - O_y
\end{pmatrix}.
\]
2.3 Geolocation

The literature presents many different ways to accomplish the task of geolocating ground targets. For example, in [22,23] the authors use information from multiple time steps to essentially construct stereo vision. In [5] the authors assume a flat-earth model and use an extended Kalman filter to perform the geolocation. Similarly, in [15] an extended information filter is used to estimated the location of a given target. Although effective, these and many other common methods are suited for single target applications as they assume perfect data association, or that all of the pixel measurements are associated with the given target.

Since the objective in this thesis is to track multiple targets in clutter, the assumption of perfect data association cannot be made. Thus, a method that simply transforms all of the measurements from the image frame to the inertial frame is needed. In [5] the authors formulate a method for calculating a single estimate of the instantaneous target location. This method assumes a flat earth model, a simplistic pinhole camera projection model, and known UAV and gimbal states. Despite being quite sensitive to measurement errors, this method does not assume any type of data association. In essence, any measurement seen in the image frame is simply transformed to provide a corresponding measurement in the inertial frame. These inertial measurements are then input to R-RANSAC which takes care of the filtering and data associations.

2.4 Recursive-RANSAC Background

As previously mentioned, tracking is performed in the inertial frame using the R-RANSAC algorithm, which is built upon RANSAC. Here, we review the RANSAC and R-RANSAC algorithms as background information for subsequent discussions of the cooperative tracking system.

2.4.1 Random Sample Consensus (RANSAC)

Least squares is a regression analysis method that estimates signal parameters by minimizing the sum of the squares, or $L_2$ norm, of the errors. This method is effective in estimating signal parameters when the errors are zero mean. However, since the calculation of this metric takes into account all measurements within the dataset the solution is greatly affected by measurement outliers.
Random sample consensus (RANSAC) was presented in [24] as a method for estimating signal parameters in the presence of measurement outliers. The RANSAC algorithm is as follows. First, a model is chosen to represent the data. For example, a line can be used to model a target’s linear trajectory. Second, RANSAC randomly selects a minimum subset of measurements. The number of measurements required is based on the selected model, and for a line the minimum subset is represented by two points. Third, a hypothesis is generated by estimating the signal parameters based on the minimum subset. Fourth, using an inlier threshold a region is specified around the hypothesis, which is then used to characterize all other measurements as either inliers or outliers. The inliers are added to the hypothesis’ consensus set. Finally, after iterating through steps 2-4, the hypothesis with the largest consensus set is chosen as the one that best represents the data. A smoothing step can then be performed by applying the least squares method to all inliers of the chosen model.

RANSAC has found primary application in the computer vision community as a method for rejecting outliers when calculating image homographies. However, RANSAC has two inherent weaknesses: 1) it is suited for post-processing as it is designed to run on a batch of data, and 2) it can only estimate the parameters of a single signal. These drawbacks are overcome by the use of Recursive-RANSAC.

2.4.2 Recursive-RANSAC

Recursive-RANSAC was originally designed to take advantage of the exceptional filtering characteristics of RANSAC, while estimating the parameters of multiple signals in real-time. R-RANSAC works by storing a set of $M$ hypothesis tracks, or models, where each model is described by its state estimate $\hat{x}$, error covariance $P$, and consensus set $\Upsilon$. Along with the model set, R-RANSAC also stores a measurement history window, which contains the past $N_w$ measurement scans, where each scan is defined as the set of all measurements at a given time step.
At each time step $k$, all models are propagated forward in time using the prediction step of the Kalman filter, defined in [25] as

$$\hat{x}[k|k-1] = A\hat{x}[k-1|k-1]$$  \hspace{1cm} (2.3) \\
$$P[k|k-1] = AP[k-1|k-1]A^T + Q.$$  \hspace{1cm} (2.4)

An inlier region $I_R$ is then specified around each state estimate using an inlier threshold $\tau_R$, such that

$$I_R = \{y : \|y - C\hat{x}[k|k-1]\|_2 < \tau_R, \}

where $y$ represents a measurement. For each model, all measurements in the current scan are classified as either inliers or outliers, depending on whether or not they lie within the track’s inlier region. The inliers are used to update the track. The original R-RANSAC algorithm does this by using the measurement update step of the Kalman filter, defined in [25] as

$$K = P[k|k-1]C^T(CP[k|k-1]C^T + R)^{-1}$$  \hspace{1cm} (2.5) \\
$$\hat{x}[k|k] = \hat{x}[k|k-1] + K(y - C\hat{x}[k|k-1])$$  \hspace{1cm} (2.6) \\
$$P[k|k] = (I - KC)P[k|k-1],$$  \hspace{1cm} (2.7)

where $K$ is the Kalman gain and $I$ is an identity matrix with the same dimensions as $A$. The inliers are then added to the given track’s consensus set. It should be noted that R-RANSAC is quite modular and can accommodate various filtering algorithms. In [3] the author demonstrates the use of the nearest neighbor, all neighbors, probabilistic data association (PDA), and joint probabilistic data association (JPDA) filters. It was found that the PDA and JPDA filters are superior in performance, however, JPDA requires significantly more computation. Therefore, in this thesis the measurement update step from the PDA filter is used. Although not presented here, the PDA filter algorithm can be found in [26] and [27].

When a measurement is found to be an outlier to all existing models, R-RANSAC does not discard the measurement. Instead the outlier is used to initialize new tracks using an iterative
process that is based on RANSAC. A random, minimum subset of measurements is selected from the measurement history window. Note that the outlier measurement is included in this subset. This minimum subset is then used to form a hypothesis model and the measurement history window is searched for all measurements that fall within the model’s inlier region. This process is repeated to produce w hypothesis models. Upon completion, the hypothesis with the largest support is propagated to the current time step and along the way is updated by the measurements in its consensus set. The model is then added to the model set. Once all models have been updated, R-RANSAC manages the model set by identifying valid tracks (i.e. those with large support), merging redundant tracks, and removing tracks with low support.

At each time step, all consensus sets are updated by removing the measurements that have left the measurement history window. An inlier ratio $\rho$ is then computed for each track as $\rho = \frac{|\Upsilon|}{N_w}$, where $|\Upsilon|$ is the cardinality of the consensus set. In essence, $\rho$ represents the amount of support that a particular track has and if it is larger than a specified inlier ratio threshold $\tau_\rho$ then the track is presumed to be a possible valid track. In addition, a lifetime threshold $\tau_T$ can be specified as the minimum number of time steps that the model needs to have existed for it to be presumed a valid track.

### 2.5 Cooperative Architecture

As the objective of this thesis is the development of a cooperative estimation system, it is beneficial to provide a short discussion on the various architectures for multi-vehicle systems. In general there are three types of architectures [28]: centralized, hierarchal, and decentralized.

In the centralized architecture each sensor sends its raw measurements to a central processor, which performs all computations to estimates the target states. This process is illustrated in Figure 2.2(a). The benefit of a centralized architecture is that since the processor has access to all sensor data it can provide a global estimate. However, the central processor is subjected to large computational loads, and there is a possibility of catastrophic failure of the entire system if the central processor fails.

In the hierarchal architecture, local fusion centers process data from sensors to produce local state estimates. These estimates are passed to a central processor which fuses all local state estimates to generate a global estimate (see Figure 2.2(b)). In some cases, the central processor
Figure 2.2: Illustration of communication in (a) centralized, (b) hierarchal, and (c) decentralized architectures. The arrows represent communication directions. Note that a decentralized system may not always be fully-connected, with bi-directional communication lines as illustrated here.

This type of architecture distributes the computational load, however, the entire system is still vulnerable to a failure in the central processor. Moreover, communication bottlenecks often occur in such a system.

In a decentralized architecture there is no central processing unit. Instead, the system consists of a network of nodes, each with its own sensor and processor. Estimates of the target states are generated by the nodes and can be communicated across the network. Figure 2.2(c) illustrates a fully-connected, decentralized system with bi-directional communication lines. However, a variety of network topologies exist. Fusion occurs at each node between the local estimate and information obtained from neighboring nodes. A disadvantage of a decentralized architecture is that since the data is distributed throughout the network it is difficult to monitor the system. Moreover, depending on the network configuration and the fusion technique used the local estimates may not be equivalent to a global estimate, and may even differ from each other. However, with the lack of central processing, decentralized systems have been shown to be scalable (no centralized computational bottlenecks), flexible (can accommodate dynamic changes to the network), modular, and robust [29]. Therefore, a decentralized system is assumed for this thesis.
CHAPTER 3.  COOPERATIVE ESTIMATION

In this chapter the main cooperative estimation methods are presented. Section 3.1 describes the optimization problem used to perform the bias estimation. Section 3.2 presents the statistical test, which determines if two tracks from different vehicles are associated and derives the necessary covariance matrix for such test to attain a desired probability of detection. Section 3.3 derives the fusion equations, while accounting for the estimated translational and rotational biases.

3.1 Track Bias Estimation

For notational purposes let $V_i$ represent the $i^{th}$ vehicle in the system. Moreover, let

$$\hat{p}_{i}^{j}[m|n] \triangleq \begin{pmatrix} \hat{p}_{i,n}^{j}[m|n] \\ \hat{p}_{i,e}^{j}[m|n] \end{pmatrix}$$

refer to the two-dimensional position estimate vector produced by $V_i$ at time step $m$ given measurements up to and including time step $n$, as expressed in the $j^{th}$ vehicle’s reference frame. The state estimate $\hat{x}_{i}^{j}[m|n]$ and error covariance $P_{i}^{j}[m|n]$ are defined similarly. Without loss of generality, we will assume that each vehicle only produces one track, and that it is unknown whether the tracks are associated. This assumption further simplifies the notation.

Consider the case where vehicles $V_i$ and $V_j$ each have a windowed set of $N$ estimates for a particular target and therefore we have in memory the state estimates, $\hat{x}_{i}^{j}[k|k]$ and $\hat{x}_{j}^{j}[k|k]$, where $k = t - N + 1, \ldots, t$ and $t$ is the current time step. From these the corresponding position estimates $\hat{p}_{i}^{j}[k|k]$ and $\hat{p}_{j}^{j}[k|k]$ can be extracted. Without truth data we cannot estimate the individual biases produced by each vehicle. The best that we can do is to estimate the relative translational bias vector $\beta_{i}^{j}$, and the relative rotation matrix $R_{i}^{j}$, that transforms the $i^{th}$ vehicle’s position estimate
into the $j^{th}$ vehicle’s reference frame as

$$\hat{p}_i^j[k|k] = R_i^j(\hat{p}_i^j[k|k] + \beta_i^j).$$

(3.1)

Note that the rotation matrix and bias vector can also be used to transform the $j^{th}$ vehicle’s estimate into the $i^{th}$ vehicle’s reference frame as in

$$\hat{p}_i^j[k|k] = R_i^jT \hat{p}_j^i[k|k] - \beta_i^j.$$

In this thesis we assume that the rotation matrix is simply a function of a constant relative rotational bias angle $\theta$ between the two tracks about the negative down axis

$$R_i^j \triangleq \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}. $$

(3.2)

Moreover, the translational biases are assumed to be just in position, and are defined as

$$\beta_i^j \triangleq \begin{pmatrix} \beta_{pn} \\ \beta_{pe} \end{pmatrix},$$

(3.3)

where $\beta_{pn}$ and $\beta_{pe}$ are constants.

The optimal rotation matrix $R_i^{j*}$ and the optimal translational bias vector $\beta_i^{j*}$ are found by solving an optimization problem which minimizes the squared error of the position data in the two tracks when expressed in the frame of the $i^{th}$ vehicle. Defining

$$\tilde{p}[k|k] = (R_i^{j*T} \hat{p}_j^i[k|k] - \beta_i^j) - \hat{p}_i^j[k|k]$$

(3.4)

$$\tilde{p}_i^j[k|k] - \hat{p}_i^j[k|k]$$

(3.5)

yields the objective

$$J = \sum_{k=t-N+1}^{t} \tilde{p}[k|k]^T \tilde{p}[k|k],$$

16
which is minimized in the optimization problem

\[
(R_j^*, \beta_j^*) = \arg\min_{R_j^*, \beta_j^*} J. \tag{3.6}
\]

The results of Equation 3.6 are used to align the tracks (entire state estimates) from the two vehicles in a common reference frame as

\[
\hat{x}_j[k|k] = \bar{R}_j^i (\hat{x}_i[k|k] + \bar{\beta}_j^i), \tag{3.7}
\]

where

\[
\bar{R}_j^i \triangleq \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \otimes R_i^j \tag{3.8}
\]

and

\[
\bar{\beta}_j^i \triangleq \begin{pmatrix} \beta_j^i^T & 0 & 0 & 0 \end{pmatrix}^T. \tag{3.9}
\]

Here, $\otimes$ represents the Kronecker product. It should be noted that obtaining a solution from the optimizer does not guarantee that the tracks are associated. To determine whether the tracks originate from the same target requires solving the track-to-track association problem, which is discussed in the next section.

### 3.1.1 Initialization of Optimizer

It should be noted that the optimization problem defined in Equation 3.6 contains local minima, thus, convergence on the proper solution requires the use of a global search or multi-start method, both of which are extremely slow. Another way to deal with the problem of local minima is to give the optimizer an initial guess for the translational and rotational biases. Two methods for obtaining an initial guess are presented in this section.
Initialization Using Homography

Homography is a fundamental concept taken from the computer vision community. It represents the projective mapping from one reference frame to another and is well documented in [18, 19]. The first step in calculating the homography between two images is to find feature points within each image. Once the features are found the next step is to match the features across the images. The goal of feature matching is to be able to assign a pixel location from one image, to a corresponding pixel location in the other image. As one might imagine, the process of finding and matching features is nontrivial and often inaccurate. This is usually due to the fact that corresponding features are not always detected in both images, and sometimes unrelated features are matched with each other. Moreover, when calculating the homography using feature points that are defined in the image plane, the intrinsic parameters of the camera must be known.

In our case, we would like to calculate the homography between two sequences of data, \( \hat{p}_i[k][k] \) and \( \hat{p}_j[k][k] \), where \( k = t - N + 1, \ldots, t \). Each data point is treated as a feature and since the data sets are collected over the same time interval we know that \( \hat{p}_i[t - N + 1][t - N + 1] \) corresponds with \( \hat{p}_j[t - N + 1][t - N + 1] \), \( \hat{p}_i[t - N + 2][t - N + 2] \) corresponds with \( \hat{p}_j[t - N + 2][t - N + 2] \), etc...

Thus, the difficulty and uncertainty of feature matching described earlier is overcome. Moreover, since the features here are defined in an object plane we do not need the intrinsic parameters of the camera to calculate the homography.

Here we will assume that \( \hat{p}_i = (\hat{p}_{i,n}, \hat{p}_{i,e}, \hat{p}_{i,d})^T \). We desire to find a homography matrix, \( H \), such that \( \hat{p}_j \approx H \hat{p}_i \), where

\[
H = \begin{bmatrix}
H_{11} & H_{12} & H_{13} \\
H_{21} & H_{22} & H_{23} \\
H_{31} & H_{32} & H_{33}
\end{bmatrix}.
\]
Defining $p^i_{j,n} = \frac{\hat{p}^i_{j,n}}{\hat{p}^i_{j,d}}$ and $p^i_{j,e} = \frac{\hat{p}^i_{j,e}}{\hat{p}^i_{j,d}}$ as in [18] we have

$$p^i_{j,n} = \frac{H_{11}\hat{p}^i_{j,n} + H_{12}\hat{p}^i_{j,e} + H_{13}\hat{p}^i_{j,d}}{H_{31}\hat{p}^i_{j,n} + H_{32}\hat{p}^i_{j,e} + H_{33}\hat{p}^i_{j,d}}$$

$$p^i_{j,e} = \frac{H_{21}\hat{p}^i_{j,n} + H_{22}\hat{p}^i_{j,e} + H_{23}\hat{p}^i_{j,d}}{H_{31}\hat{p}^i_{j,n} + H_{32}\hat{p}^i_{j,e} + H_{33}\hat{p}^i_{j,d}}.$$

Rearranging the above equations and letting $\hat{p}^i_{1,d} = \hat{p}^i_{j,d} = 1$ (since it is assumed that both tracks lie on the same plane) yields

$$-H_{11}\hat{p}^i_{j,n} - H_{12}\hat{p}^i_{j,e} - H_{13} + H_{31}\hat{p}^i_{j,n}(\hat{p}^j_{j,n}) + H_{32}\hat{p}^i_{j,n}(\hat{p}^j_{j,n}) + H_{33}(\hat{p}^j_{j,n}) = 0$$

$$-H_{21}\hat{p}^i_{j,n} - H_{22}\hat{p}^i_{j,e} - H_{23} + H_{31}\hat{p}^i_{j,n}(\hat{p}^j_{j,e}) + H_{32}\hat{p}^i_{j,n}(\hat{p}^j_{j,e}) + H_{33}(\hat{p}^j_{j,e}) = 0.$$

From these equations we can see that by letting

$$h = [H_{11}, H_{12}, H_{13}, H_{11}, H_{12}, H_{13}, H_{11}, H_{12}, H_{13}]$$

$$a_1 = [-\hat{p}^i_{j,n}, -\hat{p}^i_{j,e}, -1, 0, 0, 0, \hat{p}^i_{j,n}(\hat{p}^j_{j,n}), \hat{p}^i_{j,e}(\hat{p}^j_{j,n}), (\hat{p}^j_{j,n})]$$

$$a_2 = [0, 0, 0, -\hat{p}^i_{j,n}, -\hat{p}^i_{j,e}, -1, \hat{p}^i_{j,n}(\hat{p}^j_{j,e}), \hat{p}^i_{j,e}(\hat{p}^j_{j,e}), (\hat{p}^j_{j,e})],$$

we then have

$$a_1h = 0$$

$$a_2h = 0.$$ 

Given the sequences of data points, $\hat{p}^i_{j}[k|k]$ and $\hat{p}^i_{j}[k|k]$, where $k = t - N + 1, \ldots, t$ we form a system of linear equations

$$Ah = 0,$$ 

(3.11)
where

\[
A = \begin{pmatrix}
    a_1^T [t - N + 1]
    
a_2^T [t - N + 1]
    
    \vdots
    
a_1^T [t]
    
a_2^T [t]
\end{pmatrix}.
\]

Equation 3.11 is used to solve for \( h \) using linear least squares. The elements of \( h \) are rearranged as in Equation 3.10, to acquire the homography matrix \( H \). Using the method highlighted in chapter 4 of [19], the homography matrix can be decomposed into the translational bias vector \( \beta_i \) and rotation matrix \( R_{i,0} \), which act as initial guesses for the optimizer.

The benefit of this homography method is that it is extremely computationally efficient. This might lead one to ask the question, why even use an optimizer if the homography method is able to produce estimates of the biases at a faster rate? There are two main reasons for this. First, the homography method is much more sensitive to measurement noise than the optimization approach presented in Equation 3.6. Second, the homography method is unable to find a solution when calculating the homography between straight lines. Thus, it would fail to converge on the correct solution when the targets are traveling in straight paths over the entire window of \( N \).

**Initialization Using Trigonometry**

Consider the case where the two dimensional position vector for each track, \( \hat{p}_i \) and \( \hat{p}_j \), are rotated and biased from each other, as depicted by Figure 3.1. Notice that \( u \) and \( v \) are the vectors that point from the initial to the final locations in the window, which can be stated mathematically as

\[
u \triangleq \hat{p}_i [t] - \hat{p}_i [t - N + 1 | t - N + 1] \quad (3.12)
\]
\[
v \triangleq \hat{p}_j [t] - \hat{p}_j [t - N + 1 | t - N + 1]. \quad (3.13)
\]
Figure 3.1: The vectors $\mathbf{u}$ and $\mathbf{v}$ are defined by the initial and end locations of each track. The angle $\theta$ is found using trigonometry.

Extending Equations 3.12 and 3.13 to three dimensions under the flat-earth assumption gives the vectors

\[
\mathbf{U} \triangleq \begin{pmatrix} \mathbf{u} \\ 0 \end{pmatrix},
\]

\[
\mathbf{V} \triangleq \begin{pmatrix} \mathbf{v} \\ 0 \end{pmatrix},
\]

which are used to find $\theta$ as

\[
\theta = \tan^{-1} \left( \frac{\mathbf{U} \times \mathbf{V}}{\mathbf{U} \cdot \mathbf{V}} \right). \tag{3.14}
\]

The initial guess for the rotation matrix $R_{i,0}^j$ is found by substituting Equation 3.14 into Equation 3.2. The initial guess for the bias vector $\beta_{i,0}^j$ is found by rotating $\hat{p}_j^i$ with $R_{i,0}^j \,^T$ and calculating the average translational bias over the window, as in

\[
\beta_{i,0}^j = \frac{1}{N} \sum_{k=t-N+1}^{t} R_{i,0}^j \,^T \hat{p}_j^i[k|k] - \hat{p}_j^i[k|k].
\]
3.1.2 Updating the Rotational and Translational Biases

An assumption made in Section 3.1 is that $\theta$, $\beta_{pn}$, and $\beta_{pe}$ are constants. Such an assumption implies that the biases can be calculated once, and then used to align tracks at future time steps. However, in practice the biases vary with time. One way of handling this is to solve the optimization problem at every time step, or whenever the window is slid. Again, the optimizer can be initialized using either the homography or trigonometry methods. Alternatively, the optimizer can be initialized with the previously estimated values of the biases.

Another way of handling the time-varying biases is by performing a few iterations of a simple gradient-descent-based method. This requires the partial derivatives $\frac{\partial J}{\partial \theta}$, $\frac{\partial J}{\partial \beta_{pn}}$, and $\frac{\partial J}{\partial \beta_{pe}}$ which are presented in Appendix B. The update equations for the biases are then

$$
\theta^+ = \theta - \alpha_{\theta} \frac{\partial J}{\partial \theta}
$$

$$
\beta_{pn}^+ = \beta_{pn} - \alpha_{pn} \frac{\partial J}{\partial \beta_{pn}}
$$

$$
\beta_{pe}^+ = \beta_{pe} - \alpha_{pe} \frac{\partial J}{\partial \beta_{pe}}
$$

where $\alpha_{\theta}$, $\alpha_{pn}$, and $\alpha_{pe}$ dictate the size of the step taken along the direction of the gradient [30].

3.2 Track-to-Track Association

In this section a statistical test is developed to determine if two tracks from different vehicles represent the same target.

3.2.1 Hypothesis Test

Here, the track-to-track association problem is formulated as a hypothesis test. The two hypotheses are

$$
H_0 : \text{The two tracks are associated}
$$

$$
H_1 : \text{The two tracks are not associated}
$$
We assume that $N$ consecutive relative estimation errors are available, each defined as

$$\tilde{x}[k|k] \triangleq \hat{x}_j^j[k|k] - \hat{x}_j^j[k|k]$$

(3.15)

and organize the errors in the vector

$$\bar{\tilde{x}}[t] = \begin{pmatrix} 
\tilde{x}[t - N + 1|t - N + 1] \\
\tilde{x}[t - N + 2|t - N + 2] \\
\vdots \\
\tilde{x}[t|t] 
\end{pmatrix},$$

(3.16)

where Equation 3.15 is based on the entire state estimate: position, velocity, and acceleration. Moreover, it can be noted that Equation 3.15 is defined in the $j^{th}$ vehicle’s frame, however, this is arbitrary. Equivalent results are attained when defining it in the $i^{th}$ vehicle’s frame. Under the null hypothesis $\bar{\tilde{x}}[t] \sim N(\bar{0}, \bar{P}_0)$ and under the alternative hypothesis $\bar{\tilde{x}}[t] \sim N(\bar{0}, \bar{P}_1)$. The covariance matrix $\bar{P}_0$ is known and is given below. On the other hand, the covariance matrix $\bar{P}_1$ is not known, and depends on the (unknown) true difference between two unassociated tracks.

In the ideal case, where both $\bar{P}_0$ and $\bar{P}_1$ are known, the test statistic that follows from the log-likelihood ratio is [31]

$$L = \bar{\tilde{x}}[t]^T (\bar{P}_0^{-1} - \bar{P}_1^{-1}) \bar{\tilde{x}}[t].$$

(3.17)

However, because $\bar{P}_1$ is unknown, this test statistic is unusable. Recall that the square of a standard normal random variable follows a chi-squared distribution [32]. This also applies to the square of a multivariate vector with elements that are independent and identically distributed as zero-mean Gaussian with unit variance. The square of the Mahalanobis distance, here defined as

$$D[t] = \bar{\tilde{x}}[t]^T \bar{P}_0^{-1} \bar{\tilde{x}}[t],$$

(3.18)

is a test statistic which effectively de-correlates the elements and forces the covariance to be identity, resulting in the distribution under $H_0$ of $D[t] \sim \chi^2_{Nn_x}$ (a central chi-square random variable with $Nn_x$ degrees of freedom) [31].

23
To see this, define $e[t] \triangleq \tilde{W}_0 \tilde{x}[t]$, where $\tilde{W}_0$ is an invertible transformation matrix such that

$$e[t]^T e[t] = \tilde{x}[t]^T \tilde{W}_0^T \tilde{W}_0 \tilde{x}[t] \sim \chi^2_{N_n_x}. \quad (3.19)$$

This implies that $e[t] \sim N(\bar{0}, \bar{I})$, where $\bar{I}$ is the $N_n_x \times N_n_x$ identity matrix. From this we know that

$$E\{e[t]e[t]^T\} = E\{\tilde{W}_0 \tilde{x}[t] \tilde{x}[t]^T \tilde{W}_0^T\}$$

$$= \tilde{W}_0 E\{\tilde{x}[t] \tilde{x}[t]^T\} \tilde{W}_0^T$$

$$= \tilde{W}_0 \bar{P}_0 \tilde{W}_0^T$$

$$= \bar{I}.$$

Solving for $\bar{P}_0$ gives

$$\bar{P}_0 = \tilde{W}_0^{-1} \tilde{W}_0^{-T}$$

$$= (\tilde{W}_0^T \tilde{W}_0)^{-1},$$

which yields the Cholesky factorization

$$\bar{P}_0^{-1} = \tilde{W}_0^T \tilde{W}_0. \quad (3.20)$$

Substitution of Equation 3.20 into Equation 3.19 yields the test statistic (Equation 3.18).

Because the likelihood ratio is an increasing function of $D[t]$, by the Karlin-Rubin theorem, the following test is a uniformly most powerful test for testing $H_0$ against $H_1$ [31]:

$$\phi(D[t]) = \begin{cases} 1 & D[t] > D_\alpha \\ 0 & D[t] \leq D_\alpha \end{cases}, \quad (3.21)$$

where $\phi(D[t]) = 1$ means $H_0$ is rejected and $\phi(D[t]) = 0$ means $H_0$ is not rejected. The decision threshold is found as follows. For a given false alarm probability

$$\alpha = P(\phi(D[t]) = 1 \mid H_0) = P(D[t] > D_\alpha \mid H_0), \quad (3.22)$$
$D\alpha$ is computed from

$$\alpha = 1 - F_{D|H_0}(D\alpha) = 1 - \int_0^{D\alpha} \frac{1}{\Gamma((Nn_x/2)\gamma)}x^{((Nn_x/2)\gamma-1)}e^{-x/2}dx,$$

where $\Gamma$ represents the Gamma function. Note that under $H_0$ this produces a probability of detection $P_D = 1 - \alpha$.

### 3.2.2 Covariance Matrix $\bar{P}_0$

Under $H_0$, $\bar{x}[k|k] \sim N(0, P[k|k])$ where the covariance is defined as

$$P[k|k] \triangleq E \left\{ \bar{x}[k|k]\bar{x}[k|k]^T \right\}.$$  \hspace{1cm} (3.24)

Recall that the estimation error for each track is defined as

$$\begin{align*}
\hat{x}_i^j[k|k] &= \bar{x}_i^j[k|k] - x_i^j[k] \\
\hat{x}_j^i[k|k] &= \bar{x}_j^i[k|k] - x_j^i[k],
\end{align*}$$

which can be rearrange as

$$\begin{align*}
\hat{x}_i^j[k|k] &= \bar{x}_i^j[k|k] + x_i^j[k] \\
\hat{x}_j^i[k|k] &= \bar{x}_j^i[k|k] + x_j^i[k].
\end{align*}$$  \hspace{1cm} (3.25) \hspace{1cm} (3.26)

Substitution into Equation 3.15 yields

$$\begin{align*}
\bar{x}[k|k] &= (\bar{x}_i^j[k|k] + x_i^j[k]) - (\bar{x}_j^i[k|k] + x_j^i[k]) \\
&= (\bar{x}_i^j[k|k] - \bar{x}_j^i[k|k]) + (x_i^j[k] - x_j^i[k]).
\end{align*}$$  \hspace{1cm} (3.27)
Assuming that the two tracks are associated, \( x_i^j = x_j^j \), Equation 3.27 becomes

\[
\bar{x}[k|k] = \bar{x}_i^j[k|k] - \bar{x}_j^j[k|k].
\] (3.28)

Substitution into Equation 3.24 yields

\[
P[k|k] = E\{ (\bar{x}_i^j[k|k] - \bar{x}_j^j[k|k]) (\bar{x}_i^j[k|k] - \bar{x}_j^j[k|k])^T \}
= E\{ \bar{x}_i^j[k|k] \bar{x}_i^j[k|k]^T \} + E\{ \bar{x}_j^j[k|k] \bar{x}_j^j[k|k]^T \} - E\{ \bar{x}_i^j[k|k] \bar{x}_j^j[k|k]^T \} - E\{ \bar{x}_j^j[k|k] \bar{x}_i^j[k|k]^T \}
= P_i^i[k|k] + P_j^j[k|k] - P_{ij}^i[k|k] - P_{ji}^j[k|k],
\] (3.29)

where

\[
P_{ij}^i[k|k] = \bar{R}_i^i P_i^i[k|k] (\bar{R}_i^i)^T
\] (3.30)

\[
P_{ji}^j[k|k] = (P_{ij}^i[k|k])^T.
\] (3.31)

Here the simplification of Equation 3.30 to Equation 3.31 is a result of the assumption that the measurement and process noises are the same in each direction. A similar derivation of Equation 3.29 can be found in [7].

The calculation of \( P[k|k] \) can be simplified by assuming that at the time of the hypothesis test the filters have converged to steady-state covariances. Thus, \( P_\infty \) is of interest, where

\[
\lim_{k \to \infty} P[k|k] \to P_\infty.
\] (3.32)

For homogeneous trackers the prediction and estimation error covariances of the filters converge to steady-state values as in

\[
\lim_{k \to \infty} P_i^i[k|k - 1], P_j^j[k|k - 1] \to P_{p,\infty},
\] (3.33)

\[
\lim_{k \to \infty} P_i^i[k|k], P_j^j[k|k] \to P_{e,\infty}.
\] (3.34)
We desire $P_{e,\infty}$ but must first find the steady-state prediction error covariance $P_{p,\infty}$, which satisfies the discrete algebraic Riccati equation

$$P_{p,\infty} = A [P_{p,\infty} - P_{p,\infty}C^T (CP_{p,\infty}C^T + R)^{-1}CP_{p,\infty}]A^T + Q.$$  \hfill (3.35) 

Here, we also note that the Kalman gain for each vehicle, $K^j_i$ and $K^j_j$, reaches a steady-state value as in

$$\lim_{k \to \infty} K^j_i[k], \quad K^j_j[k] \to K_\infty,$$  \hfill (3.36) 

where

$$K_\infty = P_{p,\infty}C^T (CP_{p,\infty}C^T + R)^{-1}.$$  \hfill (3.37) 

The steady-state estimation error covariance can then be found as

$$P_{e,\infty} = (I - K_\infty C)P_{p,\infty},$$  \hfill (3.38) 

where Equations 3.37 and 3.38 are from the measurement update step of the Kalman filter. Therefore, by assuming that the filters have reached steady state the first two terms of Equation 3.29 can be replaced by $2P_{e,\infty}$.

At this point much of the literature further simplifies the calculation of $P[k|k]$ by assuming that the estimation errors from the two vehicles are independent, and consequently that the cross-covariance terms in Equation 3.29, $P^j_i[k|k]$ and $P^j_j[k|k]$, are zero \[11–13, 33, 34\]. However, this assumption is not correct since if the two tracks represent the same target they inherently share a common process noise. In \[35\] the authors present two methods for finding the cross-covariance terms.

The first of these methods is an approximation that is found by examining the volumes of two different steady-state estimation error ellipsoids, which correspond to 85% probability mass of the Gaussian distributions. The first ellipsoid is representative of the uncertainty when the estimates from both sources are fused together, while accounting for the cross-covariance, which is given by
Equation 3.29. The second is representative of the uncertainty of an individual estimate, \( P_j^i \) or \( P_j^j \). The authors note that the ratio of the first volume to the second, \( Vol_{ij}^j/Vol_{ij}^i \), remains nearly constant over varying levels of process noise.

Here the authors assume that the track fusion is performed using a linear estimation equation. This technique is highlighted in [36] for the one-dimensional case, and is extended to multi-dimensions in the equations

\[
\hat{x}[k|k] = \hat{x}_i^j[k|k] + \left[ P_i^j[k|k] - P_{ij}^j[k|k] \right] \left[ P_i^j[k|k] + P_j^j[k|k] - P_{ij}^j[k|k] - P_{ji}^i[k|k] \right]^{-1} \left[ \hat{x}_j^j[k|k] - \hat{x}_i^j[k|k] \right]
\]

(3.39)

\[
\hat{P}[k|k] = P_i^j[k|k] - \left[ P_i^j[k|k] - P_{ij}^j[k|k] \right] \left[ P_i^j[k|k] + P_j^j[k|k] - P_{ij}^j[k|k] - P_{ji}^i[k|k] \right]^{-1} \left[ P_i^j[k|k] - P_{ij}^j[k|k] \right],
\]

(3.40)

which will be discussed in Section 3.3. Assuming that \( P_i^j[k|k] = P_j^j[k|k] \) and \( P_{ij}^j[k|k] = \mu P_i^j[k|k] = \mu P_j^j[k|k] \), substitution into Equation 3.40 yields the fused covariance

\[
\hat{P}[k|k] = \frac{1 + \mu}{2} P_i^j[k|k].
\]

The corresponding ellipsoid then has a volume defined as

\[
Vol_{ij}^j = \left( \frac{1 + \mu}{2} \right)^{\frac{n_x}{2}} Vol_i^j.
\]

(3.41)

In the literature the authors examine this method for a two-dimensional case, where \( Vol_{ij}^j/Vol_i^j \approx 0.7 \). For \( n_x = 2 \), Equation 3.41 becomes

\[
Vol_{ij}^j = \frac{1 + \mu}{2} Vol_i^j.
\]
This implies that

\[ \mu = 2 \left( \frac{\text{Vol}_i^j}{\text{Vol}_j^i} \right) - 1 \]

\[ \approx 2(0.7) - 1 \]

\[ = 0.4. \]

Although presented here for two dimensions such a method can be extended to approximate the cross-covariance of higher dimensional cases.

Alternatively, the authors also present a recursion method for finding the cross-covariance term \( P_{ij}^k[k|k] \) as

\[
P_{ij}^k[k|k] \triangleq E \{ \tilde{x}_i^j[k|k] \tilde{x}_j^i[k|k] \} = (I - K_i^j[k]C)(AP_{ij}^k|k-1|k-1]A^T + Q)(I - K_j^i[k]C)^T. \tag{3.42}
\]

Again, assuming homogeneous trackers and that the filters have reached steady-state at the time of the hypothesis test the recursion becomes

\[
P_{ij}^k[k|k] = (I - K_\infty C)(AP_{ij}^k|k-1|k-1]A^T + Q)(I - K_\infty C)^T. \tag{3.43}
\]

Here it is reasonable to initialize the cross-covariance to zero since typically the initial estimates are equal to the initial measurements, which have independent errors. It should be noted that Equation 3.43 also converges over time to

\[
\lim_{k \to \infty} P_{ij}^k[k|k] \to P_{ij,\infty},
\]

as shown in Figure 3.2. The covariance (Equation 3.29) at steady-state then becomes

\[
P_{\infty} = 2P_{e,\infty} - P_{ij,\infty} - (P_{ij,\infty})^T
\]

\[
= 2(P_{e,\infty} - P_{ij,\infty}). \tag{3.44}
\]
Figure 3.2: The $L_2$ norm of the cross-covariance $P_{ij}$ converges to a steady-state value over time.
For this plot $\sigma_Q = 1$.

The covariance matrix $\bar{P}_0$ accounts for the temporal correlation between the error measurements:

$$\bar{P}_0 = E\{\bar{x}[t]\bar{x}[t]^T\},$$  \hspace{1cm} (3.45)

where $\bar{x}[t]$ is given by Equation 3.16. The form of $\bar{P}_0$ is

$$\bar{P}_0 = \begin{bmatrix} P[t - N + 1||t - N + 1] & P[t - N + 1||t - N + 2] & \ldots & P[t - N + 1||t] \\ P[t - N + 2||t - N + 1] & P[t - N + 2||t - N + 2] & \ldots & P[t - N + 2||t] \\ \vdots & \vdots & \ddots & \vdots \\ P[t||t - N + 1] & P[t||t - N + 2] & \ldots & P[t||t] \end{bmatrix}. \hspace{1cm} (3.46)$$

Notice here that $P[g||k] \triangleq E\{\bar{x}[g]g\bar{x}[k]k^T\}$, which is not to be confused with $P[g||k] \triangleq E\{\bar{x}[g]k\bar{x}[g]k^T\}$. Under the steady-state assumption of Equation 3.32, the diagonals of Equation 3.46 are equal to $P_\infty$. To account for the correlation over time, the off-diagonals must be found.

Consider the case where $g = k + 1$, so that we are interested in finding $E\{\bar{x}[k + 1||k + 1]\bar{x}[k]k^T\}$. In [35] the authors present an equation for the estimation error $\bar{x}[k + 1||k + 1]$ in terms
of $\bar{x}[k|k]$, which with the steady-state assumption is

$$
\bar{x}[k+1|k+1] = [I - K_\infty C]A\bar{x}[k|k] + [I - K_\infty C]\bar{\xi}[k] - K_\infty v[k+1],
$$

(3.47)

where $\bar{\xi}$ and $v$ are the process and measurement noises. From this it can be shown that

$$
E\{\bar{x}[k+1|k+1]\bar{x}[k|k]^T\} = E\left\{\left([I - K_\infty C]A\bar{x}[k|k] + [I - K_\infty C]\bar{\xi}[k] - K_\infty v[k+1]\right)\bar{x}[k|k]^T\right\}
$$

$$
= E\left\{[I - K_\infty C]A\bar{x}[k|k]\bar{x}[k|k]^T + [I - K_\infty C]\bar{\xi}[k]\bar{x}[k|k]^T - K_\infty v[k+1]\bar{x}[k|k]^T\right\}
$$

$$
= [I - K_\infty C]AE\{\bar{x}[k|k]\bar{x}[k|k]^T\} + [I - K_\infty C]\bar{\xi}[k]E\{\bar{x}[k|k]^T\} - K_\infty v[k+1]E\{\bar{x}[k|k]^T\}
$$

$$
= [I - K_\infty C]AE\{\bar{x}[k|k]\bar{x}[k|k]^T\}
$$

$$
= [I - K_\infty C]AP_\infty
$$

$$
= GP_\infty,
$$

(3.48)

where $G = [I - K_\infty C]A$.

Now, consider the case where $g = k + 2$, so that we are interested in finding $E\{\bar{x}[k+2|k+2]\bar{x}[k|k]^T\}$. In this situation Equation 3.47 can be used recursively to find $\bar{x}[k+2|k+2]$ in terms of $\bar{x}[k|k]$ as in

$$
\bar{x}[k+2|k+2] = [I - K_\infty C]A\bar{x}[k+1|k+1] + [I - K_\infty C]\bar{\xi}[k+1] - K_\infty v[k+2]
$$

$$
= G\bar{x}[k+1|k+1] + [I - K_\infty C]\bar{\xi}[k+1] - K_\infty v[k+2]
$$

$$
= G\left(G\bar{x}[k|k] + [I - K_\infty C]\bar{\xi}[k] - K_\infty v[k+1]\right) + [I - K_\infty C]\bar{\xi}[k+1] - K_\infty v[k+2]
$$

$$
= GG\bar{x}[k|k] + G[I - K_\infty C]\bar{\xi}[k] - GK_\infty v[k+1] + [I - K_\infty C]\bar{\xi}[k+1] - K_\infty v[k+2].
$$

(3.49)

From this it can be seen that

$$
E\{\bar{x}[k+2|k+2]\bar{x}[k|k]^T\} = E\left\{(GG\bar{x}[k|k] + G[I - K_\infty C]\bar{\xi}[k] - GK_\infty v[k+1] + [I - K_\infty C]\bar{\xi}[k+1]
$$

$$
- K_\infty v[k+2])\bar{x}[k|k]^T\right\}
$$

$$
= GGE\{\bar{x}[k|k]\bar{x}[k|k]^T\}
$$

$$
= G^2P_\infty,
$$

31
which can be generalized to

\[ E \{ \tilde{x}[k+\ell|k+\ell] \tilde{x}[k|k]^T \} = G^\ell P_\infty. \]

This result is illustrated in [8] without the steady-state assumption. Moreover, note that

\[
E \{ \tilde{x}[k|k] \tilde{x}[k+\ell|k+\ell]^T \} = (E \{ \tilde{x}[k+\ell|k+\ell] \tilde{x}[k|k]^T \})^T \\
= (G^\ell P_\infty)^T \\
= P_\infty (G^\ell)^T.
\]

Therefore, in steady-state \( \bar{P}_0 \) can be rewritten to account for the correlation of the estimation errors over time as

\[
\bar{P}_0 = \begin{pmatrix}
    P_\infty & P_\infty G^T & P_\infty (G^2)^T & \ldots & P_\infty (G^{N-1})^T \\
    G P_\infty & P_\infty & P_\infty G^T & \ldots & P_\infty (G^{N-2})^T \\
    G^2 P_\infty & G P_\infty & \ddots & \ldots & \vdots \\
    \vdots & \vdots & \ddots & P_\infty G^T \\
    G^{N-1} P_\infty & G^{N-2} P_\infty & \ldots & G P_\infty & P_\infty
\end{pmatrix}.
\]  

(3.50)

The structure of Equation 3.50 is convenient as it produces an inverse with a tri-diagonal block form, as highlighted in the following theorem:

**Theorem 1** Consider the symmetric, positive semidefinite block matrix \( \bar{P}_0 \) defined by Equation 3.50, where \( P_\infty \) is also symmetric, positive semidefinite. The inverse of the matrix \( \bar{P}_0 \) is of the form

\[
\bar{P}_0^{-1} = \begin{pmatrix}
    U & V & 0 & \ldots & 0 \\
    V^T & W & V & \ldots & 0 \\
    0 & V^T & \ddots & \ddots & \vdots \\
    \vdots & \vdots & W & V \\
    0 & 0 & \ldots & V^T & Y
\end{pmatrix},
\]  

(3.51)
where

\[ U = P_\infty^{-1} + G^T Y G \]  \quad (3.52)
\[ V = -G^T Y \]  \quad (3.53)
\[ W = Y + G^T Y G \]  \quad (3.54)
\[ Y = (P_\infty - GP_\infty G^T)^{-1}. \]  \quad (3.55)

A proof of Theorem 1 is provided in Appendix A.

Typically, the calculation of \( \bar{P}_0^{-1} \) requires the inversion of an \( N_{nx} \times N_{nx} \) matrix, and must occur every time the test statistic is calculated. However, as seen in Theorem 1, by making the steady-state assumption the inverse of \( \bar{P}_0 \) only needs to be calculated once, and can occur offline. Moreover, the calculation of \( \bar{P}_0^{-1} \) only requires the inversion of two \( n_x \times n_x \) matrices, found in Equations 3.52 and 3.55.

In addition to decreasing the computation required for \( \bar{P}_0^{-1} \), the use of this theorem allows for a recursion equation to be developed for \( D \) in Equation 3.18, which can be used to update the test statistic if the window is sliding.

### 3.2.3 Recursion Equation for Test Statistic

To simplify the notation, in this section we will redefine the estimation error as \( \tilde{x}_k \triangleq \tilde{x}[k|k] \). By using Equation 3.51, Equation 3.18 can be expanded to

\[
\tilde{x}[t]^T \bar{P}_0^{-1} \tilde{x}[t] = \left[ \tilde{x}_{t-N+1}^T U \tilde{x}_{t-N+1} + \tilde{x}_{t-N+2}^T V \tilde{x}_{t-N+2} \right] \\
+ \left[ \tilde{x}_{t-N+1}^T V \tilde{x}_{t-N+2} + \tilde{x}_{t-N+2}^T W \tilde{x}_{t-N+2} + \tilde{x}_{t-N+3}^T V \tilde{x}_{t-N+3} \right] \\
+ \left[ \tilde{x}_{t-N+2}^T V \tilde{x}_{t-N+3} + \tilde{x}_{t-N+3}^T W \tilde{x}_{t-N+3} + \tilde{x}_{t-N+4}^T V \tilde{x}_{t-N+4} \right] \\
\vdots \\
+ \left[ \tilde{x}_{t-2}^T V \tilde{x}_{t-1} + \tilde{x}_{t-1}^T W \tilde{x}_{t-1} + \tilde{x}_t^T V \tilde{x}_t \right] \\
+ \left[ \tilde{x}_{t-1}^T V \tilde{x}_t + \tilde{x}_t^T Y \tilde{x}_t \right]. \tag{3.56}
\]
Figure 3.3 illustrates the manner in which values from the test statistic at time $t$ are carried over to $t + 1$. From this, it is clear to see that the complete recursion is

$$D[t + 1] = D[t] - (d'_{t-N+1} + d''_{t-N+2} + d'''_{t}) + (d'_{t-N+2} + d''_{t} + d'''_{t+1}).$$  \hfill (3.57)
It should be noted that the original window contains \( N \) time steps, however, Equation 3.57 requires that a window of \( N + 1 \) time steps be maintained. To avoid this change in the window size the recursion equation can be broken up into two steps. First, after the hypothesis test at time \( t \) an intermediate value for \( D[t] \) is calculated as

\[
D[t]^+ = D[t] - (d'_{t-N+1} + d''_{t-N+2} + d'''_{t}).
\]

At time \( t + 1 \) the test statistic can then be updated using

\[
D[t + 1] = D[t]^+ + (d'_{t-N+2} + d''_{t} + d'''_{t+1}).
\]

### 3.2.4 Invertibility of \( \overline{P}_0 \)

There are two factors that can negatively affect the invertibility of \( \overline{P}_0 \): a decrease in the size of the time step \( dt \), and a decrease in the level of the process noise covariance \( Q \). Recall that the matrix \( G \) is used to model the time correlation between subsequent estimation errors and is defined as \( G = (I - K_\infty C)A \). We know that in a Kalman filter

\[
\lim_{Q \to 0} K_\infty \to 0,
\]

which implies that \( G \to A \). Furthermore, from Section 2.1 we see that

\[
\lim_{dt \to 0} A \to I,
\]

which signifies that \( A \) becomes idempotent \( (A^2 = A) \). This implies that as the size of the time step and the level of the process noise covariance decrease, \( G \) approaches identity, making \( \overline{P}_0 \) (Equation 3.50) singular and non-invertible. Additional discussion of the affects of both the time step size and process noise level is provided in Section 4.1, where Monte Carlo simulation results are presented.

As mentioned previously, the process noise level is tuned to attain desired tracking performance, however, this value is entirely dependent on the target dynamics, which we have no
control over. Thus, in order to overcome the possible non-invertibility of \( \tilde{P}_0 \) we increase the time difference between subsequent estimation errors. This time difference is expressed as \( \ell \) time steps. Note here that we do not increase \( dt \) directly, as this would have negative effects on the tracking performance of R-RANSAC. Instead we simply use the relative estimation error at every \( \ell^{th} \) time step when constructing Equation 3.16. Doing so reduces the observed time correlation in the test statistic, and actually enhances the power of the test [8, 37]. In such a case, Equation 3.16 becomes

\[
\tilde{x}[t] = \begin{pmatrix}
\hat{x}[t - \ell(N-1)|t - \ell(N-1)] \\
\hat{x}[t - \ell(N-2)|t - \ell(N-2)] \\
\vdots \\
\hat{x}[t - \ell|t - \ell] \\
\hat{x}[t]|t
\end{pmatrix}.
\] (3.58)

Notice that when \( \ell = 1 \), Equation 3.58 is equivalent to Equation 3.16.

The same procedure for finding the test statistic is used, however, \( G = (I - K_\infty C)A \) is replaced with \( G = [(I - K_\infty C)A]^{\ell} \). Moreover, the recursion for the test statistic (Equation 3.57) simply becomes

\[
D[t + \ell] = D[t] - (d'_{t-\ell(N-1)} + d''_{t-\ell(N-2)} + d'''_{t}) + (d'_{t-\ell(N-2)} + d''_{t} + d'''_{t+\ell}).
\] (3.59)

Note, the recursion requires that the window is slid by \( \ell \) time steps.

### 3.3 Track Fusion

If \( H_0 \) is accepted for a given pair of tracks, \((\hat{x}_i^j, P_i^j)\) and \((\hat{x}_j^j, P_j^j)\), the objective is to combine or fuse the estimates and covariances. Equations 3.39 and 3.40, which were presented previously, can be used to perform the track fusion. However, these equations are suited for a centralized multi-agent architecture. As the objective of this thesis is a decentralized system, an alternative fusion technique is presented.

The chosen method takes advantage of the fusion properties of information filters. The information filter is mathematically equivalent to the Kalman filter, however, data is represented in information space instead of state space. Recall that the filter from \( V_i \) yields the mean \( \hat{x}_i^j[t|t] \) and
estimation error covariance $P_i[t|t]$. The information filter instead produces an information matrix
$\Omega_i[t|t]$ and information vector $Z_i[t|t]$, where

$$\Omega_i[t|t] = P_i[t|t]^{-1}$$

$$Z_i[t|t] = P_i[t|t]^{-1}[t|t] \hat{x}_i[t|t].$$

The equations for the discrete time information filter are shown in [28] as

**Prediction Step**

$$\Omega_i[t|t - 1] = (A \Omega_i[t - 1|t - 1] A^T + Q)^{-1}$$

$$Z_i[t|t - 1] = \Omega_i[t|t - 1] A \Omega_i[t - 1|t - 1] Z_i[t - 1|t - 1]$$

**Measurement Update Step**

$$\Omega_i[t|t] = \Omega_i[t|t - 1] + \omega_i[t]$$

$$Z_i[t|t] = Z_i[t|t - 1] + \zeta_i[t],$$

where $\omega_i[t]$ and $\zeta_i[t]$ are defined as

$$\omega_i[t] = C^T R^{-1} C$$

$$\zeta_i[t] = C^T R^{-1} y_i[t]$$

and represent the contributions to the information matrix and information vector from the measurement $y_i[t]$.

For the track fusion to be performed on $V_i$, $\omega_j[t]$ and $\zeta_j[t]$ are found using the above equations and are sent from the $V_j$ to $V_i$. Assuming that there is no rotational or translational biases between the $V_i$ and $V_j$’s reference frames, ie... $\bar{R}^j_i = I$ and $\bar{\beta}^j_i = 0$, the fusion becomes

$$\Omega_i[t|t]^+ = \Omega_i[t|t] + \omega_j[t]$$

$$Z_i[t|t]^+ = Z_i[t|t] + \zeta_j[t].$$
Notice that the track fusion is performed by simply adding the measurement update contributions. This method can be applied to tracks that are rotated and translated by $\bar{R}_j^i \neq I$ and $\bar{\beta}_j^i \neq 0$.

The fusion on $P_i^j[t|t]$ is obtained by applying the following steps:

1. Transform $P_i^j[t|t]$ into the $V_j$'s frame using $\bar{R}_j^i$. Recall that this is $P_i^j[t|t] = \bar{R}_j^i P_i^j[t|t](\bar{R}_j^i)^T$

2. Take the inverse to produce an information matrix

3. Add $\omega_j^i[t]$

4. Take the inverse to produce covariance matrix

5. Transform back to the $V_i$'s frame using $(\bar{R}_j^i)^T$

Applying these steps produces the fusion equation for the covariance

\[
P_i^j[t|t]^+ = (\bar{R}_j^i)^T \left( P_i^j[t|t] + \omega_j^i[t] \right)^{-1} \bar{R}_j^i \\
= (\bar{R}_j^i)^T \left( (\bar{R}_j^i P_i^j[t|t](\bar{R}_j^i)^T)^{-1} + \omega_j^i[t] \right)^{-1} \bar{R}_j^i \\
= (\bar{R}_j^i)^T \left( \bar{R}_j^i P_i^j[t|t](\bar{R}_j^i)^T + \omega_j^i[t] \right)^{-1} \bar{R}_j^i \\
= \left( (\bar{R}_j^i)^T \bar{R}_j^i P_i^j[t|t](\bar{R}_j^i)^T \bar{R}_j^i + (\bar{R}_j^i)^T \omega_j^i[t] \bar{R}_j^i \right)^{-1} \\
= \left( P_i^j[t|t] + (\bar{R}_j^i)^T \omega_j^i[t] \bar{R}_j^i \right)^{-1}. \tag{3.70}
\]

Inverting both sides of Equation 3.70 yields the update equation for the information matrix

\[
\Omega_i^j[t|t]^+ = \Omega_i^j[t|t] + (\bar{R}_j^i)^T \omega_j^i[t] \bar{R}_j^i. \tag{3.71}
\]

It can be noted that when the measurement noise is equal in each direction, Equation 3.71 reduces to Equation 3.68 with $\omega_j^i[t]$ replaced by $\omega_j^i[t]$.

Similarly, to perform the fusion on $\hat{x}_i^j[t|t]$ the following steps are applied:

1. Transform $\hat{x}_i^j[t|t]$ into $V_j$'s frame using $\bar{R}_j^i$ and $\bar{\beta}_j^i$

2. Multiply by $P_i^j[t|t] = \bar{R}_j^i P_i^j[t|t]^{-1}(\bar{R}_j^i)^T$ to produce an information vector

38
3. Add $\zeta_j^t[t]$

4. Multiply by $P_i^j[t | t]^{-1} = \bar{R}_j^i P_i^j[t | t]^{-1} (\bar{R}_j^i)^T$ to convert back to state space

5. Transform back to $V_i$’s frame using $(\bar{R}_j^i)^T$ and $(-\bar{\beta}_j^i)$

Following this procedure produces the fusion equation for the state estimate

$$\hat{x}^j_i[t | t]^+ = (\bar{R}_j^i)^T \left[ P_i^j[t | t]^{-1} [t | t] \bar{R}_j^i (\hat{x}^j_i[t | t] + \bar{\beta}_j^i) + \zeta_j^t[t] \right] - \bar{\beta}_j^i$$

$$= (\bar{R}_j^i)^T \left[ \bar{R}_j^i P_i^j[t | t]^{-1} [t | t] (\bar{R}_j^i)^T \bar{R}_j^i (\hat{x}^j_i[t | t] + \bar{\beta}_j^i) + \zeta_j^t[t] \right] - \bar{\beta}_j^i$$

$$= P_i^j[t | t]^{-1} [t | t] (\hat{x}^j_i[t | t] + \bar{\beta}_j^i) + (\bar{R}_j^i)^T \zeta_j^t[t] - \bar{\beta}_j^i.$$

(3.72)

Therefore, the update equation for the information vector is

$$Z_j^i[t | t]^+ = Z_j^i[t | t] + (\bar{R}_j^i)^T \zeta_j^t[t] + (\Omega_j^i[t | t] - I) \bar{\beta}_j^i.$$

One thing to note is that Equations 3.66 and 3.67 only hold if the measurement update step in R-RANSAC is from the Kalman filter. As mentioned in Section 2.4, other filtering algorithms, such as the PDA filter, can be used. Instead of deriving information space representations of these alternative filters it can be noted from Equations 3.64 and 3.65 that

$$\omega_j^t[t] = \Omega_j^i[t | t] - \Omega_j^i[t | t - 1]$$

(3.73)

$$\zeta_j^t[t] = Z_j^i[t | t] - Z_j^i[t | t - 1]$$

(3.74)

which can also be represented as

$$\omega_j^t[t] = P_i^j[t | t]^{-1} [t | t] - P_i^j[t | t - 1]$$

(3.75)

$$\zeta_j^t[t] = P_i^j[t | t]^{-1} [t | t] \bar{x}_j^t[t | t] - P_i^j[t | t - 1] \bar{x}_j^t[t | t - 1].$$

(3.76)

Therefore, the measurement update contributions in information space can be found using the state estimate and covariance before and after the measurement update step [35].
It’s important to note that instead of communicating the measurement update contributions, some distributed fusion techniques communicate the entire information matrix and vector between vehicles. The challenge with such an approach is removing redundant information incurred by various network topologies. In [38, 39] the authors discuss a popular method referred to as track-let fusion, which removes such redundancies and has been found to be optimal when there is no process noise. Similarly, in [40] the author proposes an algorithm known as the information consensus filter to handle this situation. However, the technique proposed in this section does not have this issue since the fused information is based solely on the measurement contributions, which are independent and uncorrelated.

3.3.1 Updating the Consensus Set After Track Fusion

As mentioned in Section 2.4, every time a track is updated with a measurement that measurement is added to the model’s consensus set $\Upsilon$. The inlier ratio (which is between 0 and 1 in the current implementation) is calculated from the consensus set and is essentially a measure of how much support the model has. When a track is updated using information from another vehicle the consensus set must be updated to reflect the increase in model support. This is done by adding an $inf$ to the consensus set. Each $inf$ is time stamped and removed from $\Upsilon$ at appropriate times. The reason for adding an $inf$ is to distinguish elements in the consensus set that are due to the model being updated locally from those that are caused by the model being updated by another vehicle. This is important for Section 5. It is worth noting that updating the consensus set based on information received from other vehicles allows the inlier ratio to be greater than 1.

3.4 Conclusion

In this chapter we presented specific methods that enable a cooperative system to track geolocated ground targets. The bias estimation method was formulated as an optimization problem, which solved for the optimal rotation matrix and translational bias vector that minimized the squared error between two tracks over a window of time. Two techniques for obtaining initial guesses of the biases were presented as ways to overcome the multiple local minima that exist in the solution space of the optimization problem. The track-to-track association method was based
on a sliding window approach, with the square of the Mahalanobis distance as the test statistic. It was shown that by making a steady-state assumption the inverse of the covariance matrix, which is required for the calculation of the test statistic, has a convenient tri-diagonal block structure and can be calculated offline. A novel recursion method was also developed, which decreases the computation required to recalculate the test statistic as the window is slid. Finally, the equations for track fusion were derived and take into account the estimated rotational and translational biases. These equations were presented in both state and information space.
CHAPTER 4. RESULTS

In this chapter the results of the cooperative estimation methods from Chapter 3 are presented. In Section 4.1 Monte Carlo simulations are performed to validate the bias estimation and track-to-track association. These methods, along with the track fusion, are demonstrated using video obtained from stationary (Section 4.2.1) and actual UAV platforms (Section 4.2.2). Lastly, Section 4.3 provides a short discussion on the validity of one of the governing assumption from Section 3.1, based on observations from the video results.

4.1 Simulation Results

A Monte Carlo simulation was performed to test the bias estimation and track-to-track association under the assumption of $H_0$ (two associated tracks). The performance of the algorithms was analyzed for varying process noise levels $\sigma_Q$, window lengths $N$, and time steps between subsequent estimation errors $\ell$. For each test a random target trajectory was generated using Equation 2.1, where $dt = 0.01$ seconds. Two tracks were then produced using a Kalman filter. The tracks contained a relative rotational and translational bias, which were determined randomly (see Figure 4.1). The bias estimation and track-to-track association were performed on these tracks. For a given value of $\sigma_Q$, $N$, and $\ell$, 250 tests were performed and the test statistic was compared to the threshold, which was calculated based on $\alpha = 0.05$. The results of the entire simulation, where $\sigma_Q \in [0.1, 10, 100]$, $N \in [10, 50, 100]$, and $\ell \in [2, 10, 50]$, is shown in Figure 4.2.

For many of the test scenarios $P_D$ was approximately equal to the designed value of $1 - \alpha = 0.95$. This was indication that the calculated test statistic followed the theoretical chi-squared distribution with $N\alpha$ degrees of freedom, which is evident for example in Figure 4.3 for $(\sigma_Q, N, \ell) = (10, 100, 10)$. Such scenarios validate the calculation of $\bar{P}_0$ and $\bar{P}_0^{-1}$. However, it can also be seen that for several test scenarios, the test statistic failed to attain the desired probability of detection, and thus did not follow the theoretical distribution. This behavior was due to $\bar{P}_0$ becoming singular.
Figure 4.1: Example of associated tracks, randomly generated during the Monte Carlo simulations. The tracks are separated by relative rotational and translational biases.

(as discussed in Section 3.2.4) and was a function of $\ell$ and $\sigma_Q$. Note that when the process noise level was very small ($\sigma_Q = 0.1$), even a large time step ($\ell = 50$) produced incorrect results. For moderate and high process noise levels ($\sigma_Q = 1, 10, 100$) it can be seen that increasing $\ell$ yielded the desired probability of detection, and thus produced the correct distribution. It is interesting to note though that as $\bar{P}_0$ becomes singular the result is a decrease in the calculated test statistic.

Moreover, it can be noted that the plot associated with the case of $(\sigma_Q, N, \ell) = (100, 10, 2)$ did not follow the trends that have been previously mentioned. The cause of this anomaly seemed to be that the optimizer used for the bias estimation struggled to converge to proper solutions, which sometimes resulted in extremely large estimation errors and a low $P_D$. This breakdown in the bias estimation was primarily due to the high level of process noise and was remedied by an increase in $\ell$ and/or $N$.

Lastly, a Monte Carlo simulation was performed under the assumption of $H_1$ (two unassociated tracks). Although not presented in a figure, for all variations of $\sigma_Q$, $N$, and $\ell$ the algorithm was able to correctly reject $H_0$ with probability $P_R = 1$.

\footnote{While tuning R-RANSAC for various tracking scenarios it has been observed that for most practical applications, where the objective is to track maneuvering targets, the process noise is typically at a moderate to high level.}
Figure 4.2: Monte Carlo simulations testing the bias estimation and track-to-track association for randomly generated associated tracks. Each graph contains 250 individual tests and compares the calculated test statistic $D$ (blue) with the threshold $D_{\alpha}$ (red) for different values of $\sigma_Q$, $N$, and $\ell$. The calculated probability of detection $P_D$ is also shown for each scenario. For these tests $\alpha = 0.05$.

4.2 Video Results

In this section the results obtained from actual video are presented.

4.2.1 Test With Stationary Platforms

The complete cooperative estimation system was tested in a tracking scenario that involved two stationary cameras (simulating a hovering scenario), each viewing the tracking area from a different point of view, as seen in Figure 4.4.
Figure 4.3: Normalized histogram of the test statistic for \((\sigma_Q, N, \ell) = (10, 100, 10)\), compared to the PDF of the theoretical chi-squared distribution with \(Nn_x = 600\) degrees of freedom.

For this test the position and orientation of the cameras were calculated by determining the mapping of known inertial coordinates to their corresponding locations in the image frame. The images were processed as a rate of 10Hz \((dt = 0.1\) seconds) and a foreground detector based on the KLT method [4] was used to produce pixel measurements for each target, which were transformed into the inertial frame (geolocated). Note that the inertial frame was specified using a north-east-down (NED) frame of reference. These measurements in the inertial frame were input to R-RANSAC, which produced states for each target.

The tracks produced by each target can be seen in the Figure 4.5, where the green tracks are from one vehicle \((V_1)\), while the cyan tracks are from the other vehicle \((V_2)\). This figure clearly illustrates the rotational and translational biases that separate the associated tracks.

A window of \(N = 10\) state estimates was stored with \(\ell = 10\). This window was used to calculate the rotational and translational biases as in Section 3.1. The rotation matrix and bias vector were then used to transform the tracks from one vehicle into the reference frame of the
other vehicle, as in Equation 3.7. Applying the bias estimation to two associated tracks can be seen in Figure 4.6. It is clear that the bias estimation technique was effective in transforming both tracks into a common reference frame, which is vital for performing the track-to-track association. The application of the bias estimation to two unassociated tracks is shown in Figure 4.7. Notice that despite the tracks being unassociated the optimizer still returned a rotation matrix and bias vector that minimized the squared error.
Figure 4.5: The two targets from Figure 4.4 are geolocated by two cameras. The circles represent the target locations at the current time step, while the trails represent the track history. Green denotes the tracks from $V_1$, and cyan represents the tracks from $V_2$. Due to sensor biases, the associated tracks are biased from each other.

Figure 4.6: Two associated tracks, before (left) and after (right) the bias estimation.

At every time step the window was slid, the bias estimation applied, and the track-to-track association performed. The threshold for the test statistic was based on $\alpha = 0.05$. The results of the track-to-track association over the entire video sequence can be seen in Figure 4.8. Note that the track-to-track association was performed between a single track from $V_1$ with the two tracks from $V_2$, which yielded an associated track pair as well as an unassociated track pair. For each column (the left and right columns representing the associated and unassociated cases, respectively) the top
Figure 4.7: Two unassociated tracks, before (left) and after (right) the bias estimation.

Figure 4.8: Track-to-track association between two associated tracks and two unassociated tracks. For each column the top plots represent the determined association over time, where a 1 indicates that $H_0$ was rejected and a 0 indicates that $H_0$ was not rejected. The bottom plots show the test statistic over time (blue) compared to the threshold (red).

...plots represent the determined association, corresponding to Equation 3.21, over time. Recall that a 1 means that $H_0$ was rejected for the track pair, while a 0 indicates that $H_0$ was not rejected. The bottom plots show the test statistic over time, compared to the threshold. As seen, over the entire video sequence the track-to-track association algorithm was able to correctly accept and reject $H_0$ with $P_D = 1$ and $P_R = 1$.

For the complete system, once $H_0$ was accepted for a given track pair the tracks were fused. With the fully-connected network the fusion decreased the covariance of each track and brought the two tracks more into consensus, i.e., $\bar{x}[k|k]$ decreased. The effect of the track fusion can be seen
Figure 4.9: Results of the track fusion. On the left two associated tracks are aligned, however, no track fusion is performed. On the right are the same tracks, however, with track fusion applied over the entire window. It is clear to see that the track fusion was effective in reducing the relative error between the two tracks.

in Figure 4.9. The left plot shows two associated tracks that were aligned using the bias estimation technique, with no track fusion. Notice that there were several areas where the two tracks did not fully line up. Over the entire window ($N = 10$, $\ell = 10$) the RMS error of the position states between the two tracks was found to be 0.364 m. On the other hand, the right plot shows the tracks with track fusion applied over the entire window. Here it can be seen that the fusion caused the two tracks to be more aligned. As a result, the RMS error over the window decreased to 0.037 m. The difference in the RMS errors between the two scenarios is approximately an order of magnitude.

Another benefit of the fusion process is that it allows a vehicle to receive information about targets that they themselves are not detecting, which is beneficial for environments with occlusions. Consider the scenario where $V_1$ is tracking a particular target. When $t \leq 25$ seconds, $V_1$ updates the track using its own local measurements. At $t = 25$ seconds the bias estimation and track-to-track association methods are performed. When $25 < t \leq 35$ seconds the track is updated with the local measurements plus the measurement contributions from $V_2$ (track fusion). When $35 < t \leq 45$ seconds, $V_1$ stops receiving measurements for the given target, simulating an occlusion. However, the track continues to be updated using the measurement contributions form the other vehicle. At $45 < t \leq 55$ seconds, the target is no longer occluded from $V_1$ and the track is once again updated using information from both vehicles. When $55 < t \leq 65$ seconds, the track is updated by $V_1$ only. The results of this scenario are illustrated in Figure 4.10.
Figure 4.10: A single target tracked from $V_1$. In the first section (green), $V_1$ tracks the target using its own local measurements. In the second section (blue), the track is updated by the local measurements from $V_1$ and the measurement update contributions from $V_2$. In the third section (red), $V_1$ no longer receives measurements, simulating an occlusion. In the fourth section (black), the track is again updated with information from both vehicles. In the last section (magenta), the track is updated by $V_1$ only. Track fusion allows the vehicle to continually track the target even through the occlusion.

From this scenario we can see that the method for track fusion allows $V_1$ to continually track the target through a simulated occlusion with no track fragmentation. The ability of a vehicle to receive information about targets that they themselves are not detecting is one of the main advantages of a cooperative system, and will be leveraged in Chapter 5 to cooperatively estimate the size and location of occlusions.

4.2.2 Test With UAV Platforms

A test was performed with data collected from actual UAV platforms. A description of the hardware used can be found in Appendix B. The test included a single target and two UAVs, which viewed the tracking area from a different angle (see Figure 4.11). However, unlike the previous test the UAV platforms were not stationary. A plot of the flight paths can be seen in Figure 4.12. Note that the video sequences contained a single target, therefore the data was used to validate the
method under the assumption of $H_0$ only. The results from the test are summarized in Figure 4.13. As seen, the algorithm was effective in associating the two tracks from the different vehicles and had a probability of detection $P_D = 1$. These results affirm the effectiveness of the method in the presence of actual UAV sensor biases and noise.

4.3 Discussion of Biases Over Time

Recall that one of the assumptions made in Section 3.1 is that the translational and rotational biases are constant. However, as we plot the estimated values of $\beta_{pn}$, $\beta_{pe}$, and $\theta$ (Figure 4.14)
Figure 4.13: Track-to-track association between two associated tracks. For each column the top plots represent the determined association over time, where a 1 indicates that $H_0$ was rejected and a 0 indicates that $H_0$ was not rejected. The bottom plots show the test statistic over time (blue) compared to the threshold (red).

Figure 4.14: Translational and rotational biases over time from test with stationary camera platforms.

From the test in Section 4.2.1, we find that the estimated biases are actually time-varying. The reason for this is that in Section 2.1 the target model assumed that the measurement noise covariance is constant and symmetric in the north/east directions. However, in a geolocation application this is not necessarily true. To see this, consider Figure 4.15. Here, for simplicity we use the angular
Figure 4.15: When viewing a target from directly above, the angular noise $\delta$ projects a measurement noise in the inertial frame of $d_1$. When viewing a target from an angle, $\delta$ projects to $d_2$. The fact that $d_2 \neq d_1$ indicates that the measurement noise in a geolocation process is time-varying and depends on the UAV and target position.

perturbation $\delta$ to represent all sensor noise (whether from the camera, UAV, or gimbal) after the geolocation process. If the UAV views a target from directly above, $\delta$ produces a corresponding measurement noise in the inertial frame $d_1$. Similarly, if the UAV views a target from an angle, $\delta$ produces the measurement noise $d_2$, where $d_1 < d_2$. Therefore, even with stationary camera platforms as the targets maneuver the actual measurement noise (in the inertial frame) changes. The result of the time-varying measurement noise is morphing or distortion of the generated tracks.

One possible way of dealing with this issue is to incorporate a time-varying measurement noise covariance, which is a function the target position, UAV position/attitude, and gimbal angles. Another possible solution is to include some distortion factors in the bias estimation scheme. However, it should be noted that despite this breakdown in one of the governing assumptions, the cooperative estimation scheme was still able to make the proper associations, leading to the conclusion that it is a robust method.

4.4 Conclusion

In this chapter we presented the main results for the bias estimation, track-to-track association, and track fusion algorithms. Monte Carlo simulations were used to verify the efficacy of the bias estimation scheme in conjunction with the track-to-track association method. Simulation re-
results showed that the calculated test statistic followed the theoretical chi-squared distribution with \( Nn_x \) degrees of freedom, and as a result the desired probability of detection was attained. These results support the conclusion that the covariance matrix \( \hat{P}_0 \) and its inverse were calculated correctly. The methods were applied to actual video obtained from stationary camera and maneuvering UAV platforms. In both scenarios the system was able to make the correct associations with a probability of detection \( P_D = 1 \) and a probability of rejection \( P_R = 1 \). The results from the UAV test indicate that such methods are effective in the presence of actual UAV sensor biases and noise. The track fusion method was also tested using actual video. The results presented showed that the method was able to decrease the RMS error between two tracks by an order of magnitude, bringing the tracks from different vehicles more in consensus with each other.
CHAPTER 5. OCCLUSION ESTIMATION

Once the tracking system has identified and located the targets the next objective is to plan a path and point the gimbal for each vehicle to maximize some performance metric. The metric used may vary depending on the application, however, possible metrics could include the number of targets seen in each camera’s field of view, or the number of targets seen collectively by the team of UAVs. In any case, implementing an optimized path planner/gimbal pointing algorithm requires not only the locations of the vehicles and targets, but also the location and size of occlusions. There has been much work in the field of path planning where the researchers have assumed the use of a geographical map, which includes the location, size, and height of occlusions [5, 41–44].

This section presents a method for cooperatively estimating the location and size of occlusions in real-time, without the use of an *a priori* geographical map. The assumptions made up to this point are maintained with the addition of the assumption that all occlusions are infinitely tall. This implies that any object located between a UAV and target is guaranteed to cause an occlusion, regardless of the UAV’s altitude. This assumption is reasonable in low-altitude flights such as urban environments, where the altitude of the UAV is lower than the buildings, or in environments where the occlusions are much taller than the target height. Note that the occlusions are assumed to be stationary, and do not include other targets.

5.1 Method

Consider the following possible cases for a tracking scenario involving vehicle \( V_i \) and a given target.

1. The target is not within the field of view of the \( i^{th} \) vehicle’s camera

2. The target is within the field of view of the \( i^{th} \) vehicle’s camera and \( V_i \) is detecting the target
3. The target is within the field of view of the \(i^{th}\) vehicle’s camera, however, \(V_i\) is not detecting the target

Case 1 occurs when the camera is not pointing at the target. In this case, the system cannot make any conclusion as to whether there is an occlusion between \(V_i\) and the target. However, in Case 2 the system can conclude with some certainty that there is no occlusion in the line of sight. Similarly, in Case 3 the system can conclude with some certainty that there is an occlusion somewhere along the line of sight, which causes \(V_i\) to not detect the target.

To determine if the target is within the field of view of the camera the inverse of the geolocation process from Section 2.3 is used. Essentially, given the estimated position of the target along with the states of \(V_i\) and its gimbal, the pixel location of the target is calculated. If the calculated pixel location is not within the camera’s field of view then Case 1 has occurred. However, if the calculated pixel location is within the camera’s field of view then it is either Case 2 or 3.

Recall from Section 2.4 that each model maintained by R-RANSAC has an associated consensus set, which stores the measurements that have been used to update that model over the measurement history window. In Section 3.3.1 it was noted that for a cooperative system an \(inf\) is added to the consensus set every time the model is updated using information that is received from another UAV. Note that here, the term measurement is used to describe the measurements obtained from the UAV’s local sensor, and does not refer to measurements from other vehicles. The measurements along with the \(infs\) in the consensus set can be used to indicate if the vehicle is detecting a given target.

For example, if a model from \(V_i\) has a consensus set that contains a significant number of measurements then we know that the model is being updated by the \(i^{th}\) vehicle’s local estimator, and therefore we conclude that \(V_i\) detects the target (Case 2). However, if the model’s consensus set contains a significant number of \(infs\) and little to no measurements, then we know that the model is being updated primarily by the estimator of \(V_j\), and we conclude that \(V_i\) does not detect the target (Case 3). Once it has been determined which case has occurred a likelihood map can be updated accordingly.

The likelihood map segments the tracking area, which is specified in the world frame, into an \(r \times s\) grid. The variable \(\eta_{ij} \in [0, 1]\), where \(i \in [1, r]\) and \(j \in [1, s]\), represents the likelihood of an occlusion being at the location of that \(ij\) element. An update region is specified for each UAV-track.
Figure 5.1: The likelihood update region is specified by projecting a Gaussian distribution from
the position of the target $\hat{x}$ to the position of the UAV. The size of the Gaussian is determined by
the error covariance of the target estimate. In this example $\hat{x} = (30, 30)^T$ and the UAV is at $(0, 0)^T$.
	pair, and is defined as the set of all elements whose likelihood value is to be updated. This region
is formed by the vectors that extend from the north-east position of the UAV to either side of the
$3\sigma$ bound of the target, which is calculated from the error covariance. A Gaussian is projected
from the target to the UAV position (Figure 5.1), and scaled so that each element within the update
region is $\gamma_{ij} \in (0, 1]$. Note that the values outside the update region are $\gamma_{ij} = 0$. In essence, the
projected Gaussian provides a scaling according to the element’s position relative to the direct line
of sight.

This value is multiplied by the parameter $\Lambda_{ij}$, which is determined based on the case num-
ber as in

$$
\Lambda_{ij} = \begin{cases} 
0, & \text{Case 1} \\
\tau_{\text{dec}}, & \text{Case 2} \\
\tau_{\text{inc}}, & \text{Case 3}
\end{cases}
$$

(5.1)

where $\tau_{\text{dec}} < 0$ and $\tau_{\text{inc}} > 0$. These parameters dictate how much the likelihood can change at each
time step and can be tuned. It should be noted that in Case 2 we conclude that there is no occlusion
along the entire line of sight. However, in Case 3 it is unknown where along the line of sight the
occlusion exists. To account for this greater amount of certainty in Case 2 as opposed to Case 3, we set \( |\tau_{\text{dec}}| > |\tau_{\text{inc}}| \).

The product of \( \gamma_{ij} \) and \( \Lambda_{ij} \) is used to update the value of \( c_{ij} \) as

\[
c_{ij}[t] = c_{ij}[t-1] + \gamma_{ij}[t]\Lambda_{ij},
\]  

which in turn is used to calculate the likelihood according to the sigmoid function

\[
\eta_{ij}[t] = \frac{1}{1 + e^{-c_{ij}[t]}},
\]  

A plot of the sigmoid function is found in Figure 5.2. With no prior knowledge of the tracking area, each element is initialized to \( \eta_{ij}[0] = 0.5 \Rightarrow c_{ij}[0] = 0 \), indicating that there is an equal likelihood of there being an occlusion as there is of there not being an occlusion.

After updating the likelihood map each element is compared to a threshold \( \eta_{\text{threshold}} \) to create a binary map. Each element in the new map \( \lambda_{ij} \) is calculated as

\[
\lambda_{ij} = \begin{cases} 
1, & \eta_{ij} \geq \eta_{\text{threshold}} \\
0, & \text{otherwise.}
\end{cases}
\]  

Figure 5.2: Elements in the likelihood map are updated according to the sigmoid function.
The map is treated as a binary image and blob detection techniques, borrowed from the computer vision community, can be used to determine the position and size of the estimated occlusions.

Although not presented in the subsequent results, a natural extension of this method is to estimate the height of each occlusion. This can be done by creating a three-dimensional likelihood map, where each voxel is updated according to Equations 5.2 and 5.3. In such a case, the update region (which is now a volume) is defined as a Gaussian projected from the target to the UAV location and is represented by a 3-dimensional cone along the vector $S$ in Figure 5.3.

### 5.2 Simulation Results

The occlusion estimation technique was tested in a simulation involving a single occlusion, two vehicles, and four targets (Figure 5.4). The targets were evenly spaced around the occlusion, which was centered at the origin of the tracking area, and traveled at constant speeds in a counterclockwise orbit. The two vehicles were also evenly spaced around the occlusion, and traveled at constant speeds in a clockwise orbit. Note that at all times both cameras were pointed at the center of the occlusion, which is not required but is simply an artifact of the simulation environment.

The likelihood map of $V_1$ after approximately 34 seconds of simulation time can be seen in Figure 5.5. Notice the elevated elements in the middle of the tracking area of Figure 5.5(a), which represent areas of high likelihood. The depressed elements, or areas of low likelihood, can also be noted. Using the threshold $\eta_{\text{threshold}} = 0.65$ a binary map was created (see Figure 5.6). Again, the
Figure 5.4: Simulation used to test occlusion estimation. The occlusion (red circle) has a radius of 13 m and is centered at (0.0 m, 0.0 m). The four targets (green circles) are evenly spaced and travel in a counterclockwise direction around the occlusion on the path defined by the dotted box. The UAVs are also evenly spaced around the occlusion and travel in a clockwise direction along the path defined by the dotted orbit. During the simulation the UAVs remained at a constant altitude.

map was treated as a binary image since all of the values were either a 0 or 1. Using a blob detection function from the MATLAB Computer Vision System Toolbox the occlusion estimation technique determined that the occlusion was centered at the north/east position of (0.8 m, 1.5 m) and had a radius of 11.6 m. Compare this to the actual occlusion which was centered at (0.0 m, 0.0 m) and had a radius of 13.0 m.

5.3 Conclusion

Using a vision-based target tracking system to cooperatively estimate the location and size of occlusions in real-time is an unexplored research topic. This capability bridges the gap between the target tracking and optimal path planning/gimbal pointing, which is essential for a robust and autonomous system.
Figure 5.5: Likelihood map of $V_1$ viewed from different angles.

Figure 5.6: A threshold is applied to the likelihood map, creating a binary map, where each element is either a 0 (blue) or 1 (yellow). After approximately 34 seconds of simulation time the estimation scheme reveals a large cluster of high likelihood elements. Compare this to the actual occlusion, which is represented by the red dotted circle.
The preliminary results from Section 5.2 indicate that such a feature is not only feasible, but has the potential to be quite accurate. Therefore, the results warrant additional work in this area. It is recommended that this additional work include extending the method to estimate the height of occlusions.
6.1 Summary of Contributions

The contributions of this thesis involve the development of four cooperative estimation methods: bias estimation, track-to-track association, track fusion, and occlusion estimation.

The state-of-the-art algorithms for bias estimation only account for relative translational biases. However, the bias estimation technique presented in Section 3.1 is able to account for the relative rotational bias as well. This was performed by optimizing the track alignment with respect to the two-dimensional position and the heading angle between two tracks. It was noted that the solution space of the optimization problem may contain local minima, and thus two methods were presented for calculating initial guesses of the biases. Such can be used to seed the optimizer in order to converge on proper solutions.

The track-to-track association method presented in Section 3.2 was based on a sliding window approach and used as its test statistic the square of the Mahalanobis distance. In this thesis the covariance matrix $\bar{P}_0$ necessary for the calculation of the test statistic was derived under a steady-state assumption. It was shown that by making such an assumption, $\bar{P}_0^{-1}$ has a tri-diagonal block structure, where each $n_x \times n_x$ block has a closed-form solution. This finding was presented as a theorem, the proof of which is included in Appendix A. As a result, instead of having to calculate and invert $\bar{P}_0$ (which is of size $Nn_x \times Nn_x$) every time the test statistic is to be calculated, $\bar{P}_0^{-1}$ can be calculated once offline and only requires the inversion of two $n_x \times n_x$ matrices. Results from Monte Carlo simulations showed the calculated test statistic matched the theoretical chi-squared distribution, and yielded the desired probability of detection. This method, along with the bias estimation technique, were also applied to actual video data. The results indicated that such methods are effective in the presence of actual UAV sensor biases and noise. An additional contribution of this thesis is the development of a recursion equation for the test statistic, which can be used to decrease the amount of computation required to recalculate the test statistic as the window is slid.
The equations for track fusion presented in Section 3.3 were based on information filters and were derived to take into account the estimated translational and rotational biases between tracks. These equations were presented in both state and information space, and were shown to decrease the relative RMS error between associated tracks by approximately an order of magnitude, bringing the two tracks from different vehicles more in consensus with each other.

In Chapter 5, equations were developed to update a likelihood map, which is used to indicate areas in which occlusions are likely to exist. The occlusion estimation technique was demonstrated in simulation and was shown to be effective in estimating both the two-dimensional size and location of occlusions. To the best of our knowledge the occlusion estimation method presented in this thesis is the first of its kind; using a cooperative, vision-based target tracking system to estimate the location and size of occlusions in real-time. Having this capability is important for future research, as it bridges the gap between target tracking and optimal path planning/gimbal pointing, without the need of an a priori geographical map.

6.2 Recommendations for Future Work

The cooperative target tracking system developed in this thesis was shown to work well with simulated and actual video data. However, there are several areas which warrant additional consideration and development. These areas are listed below.

- In the current implementation of R-RANSAC the process noise covariance $Q$ is set by the user at the beginning of tracking. The true process noise depends on the actual dynamics of the target, which varies over time. It is recommended that an adaptive Kalman filter be used within R-RANSAC to dynamically estimate $Q$. Such would not only improve the tracking performance, but would also alleviate the burden of tuning R-RANSAC from the user.

- In this thesis it is assumed that the data from each vehicle is synchronized in time. In practical applications this may not be the case, and can greatly affect the results of the bias estimation. Thus, it is recommended that researchers incorporate methods for estimating the time delay between the two sequences of data. A possible solution to this problem could lie in a method known as dynamic time warping.
• In some cases the matrix $\bar{P}_0$ can become singular, and thus noninvertible. This behavior is dependant on the size of the time step and level of process noise. It is recommended that an analysis be performed to determine the following: for a given level of process noise, what is the minimum size of the time step required to ensure invertibility of $\bar{P}_0$?

• In Section 4.3 it was noted that the geolocation process produces time-varying measurement noise, which can cause morphing or distortion of generated tracks. Although the results from Chapter 4 indicate that the cooperative estimation methods can handle this to some degree, it is recommended that the bias estimation technique be reformulated with an additional skewing factor in both the north and east directions. Doing so can help preserve the robustness of the methods when the morphing is severe.

• The occlusion estimation technique presented in Chapter 5 assumes that all occlusions are infinitely tall. With this assumption, the method is able to estimate the position and two-dimensional size of occlusions. It is recommended that this assumption be removed, and a three-dimensional likelihood map be used. Doing so can potentially enable the method to also estimate the height of occlusions.

It is believed that additional research in these specific areas can help to further capitalize on the advances and contributions of this thesis, and will increase the robustness and autonomy of the vision-based, multiple target tracking system.
REFERENCES


[34] Hongyan Zhu and Suying Han. Track-to-Track Association Based on Structural Similarity in the Presence of Sensor Biases. 2014, 2014. 27


APPENDIX A. PROOF OF THEOREM 1

Proof: Recall that the matrices $\bar{P}_0$ and $\bar{P}_0^{-1}$ are $Nn_x \times Nn_x$, where $N$ is the size of the window and $n_x$ is the number of states. From here $\bar{P}_0$ and $\bar{P}_0^{-1}$ will instead be referred to as having $N \times N$ blocks, where each block is of size $n_x \times n_x$ and is defined in Equations 3.46 and 3.51. The proof of Theorem 1 requires that

$$\bar{P}_0 \bar{P}_0^{-1} = \begin{pmatrix}
I & 0 & 0 & \ldots & 0 \\
0 & I & 0 & \ldots & 0 \\
0 & 0 & \ddots & \ddots & \ddots \\
\vdots & \vdots & I & 0 & \ddots \\
0 & 0 & \ldots & 0 & I
\end{pmatrix},$$

where $I$ is the $n_x \times n_x$ identity matrix. To simplify notation $P_\infty$ is replaced with $P$.

Consider the multiplication of the 1st row of $\bar{P}_0$ with the 1st column of $\bar{P}_0^{-1}$:

$$PU + PG^TV^T = P[P^{-1} + G^T(P - GPG^T)^{-1}G] - PG^T(P - GPG^T)^{-1}G$$

$$= I + PG^T(P - GPG^T)^{-1}G - PG^T(P - GPG^T)^{-1}G$$

$$= I.$$

Consider the multiplication of the $i^{th}$ row of $\bar{P}_0$ with the $i^{th}$ column of $\bar{P}_0^{-1}$, where $1 < i < N$:

$$GPV + PW + PG^TV^T = -GPG^TY + P(Y + G^TYG) - PG^TYG$$

$$= (P - GPG^T)Y$$

$$= I.$$
Now consider the multiplication of the $N^{th}$ row of $\tilde{P}_0$ with the $N^{th}$ column of $\tilde{P}_0^{-1}$:

$$GPV + PY = -GPG^T (P - GPG^T)^{-1} + P (P - GPG^T)^{-1}$$

$$= (P - GPG')(P - GPG')^{-1}$$

$$= I.$$

Therefore, all of the diagonal elements of $\tilde{P}_0 \tilde{P}_0^{-1}$ are $I$.

Now consider the multiplication of the 1$^{st}$ row of $\tilde{P}_0$ with the 2$^{nd}$ column of $\tilde{P}_0^{-1}$:

$$PV + PG^TW + P(G^2)V = -PG^T (P - GPG^T)^{-1}$$

$$+ PG^T \left[(P - GPG^T)^{-1} + G^T (P - GPG^T)^{-1} G\right]$$

$$- P(G^2)^T (P - GPG^T)^{-1} G$$

$$= 0.$$

The above equation can be extended to the general case when the $i^{th}$ row of $\tilde{P}_0$ is multiplied by the $j^{th}$ column of $\tilde{P}_0^{-1}$, where $i < j$ and $k = j - i - 1$

$$P(G^i)V + P(G^{i+1})W + P(G^{i+2})V = -P(G^{i+1})^T (P - GPG^T)^{-1}$$

$$+ P(G^{i+1})^T \left[(P - GPG^T)^{-1} + G^T (P - GPG^T)^{-1} G\right]$$

$$- P(G^{i+2})^T (P - GPG^T)^{-1} G$$

$$= 0.$$

This implies that the upper triangle of $\tilde{P}_0 \tilde{P}_0^{-1}$, excluding the diagonal, consists entirely of zeros. Since both $\tilde{P}_0$ and $\tilde{P}_0^{-1}$ are symmetric the product of the two is also symmetric, which implies that the lower triangle of $\tilde{P}_0 \tilde{P}_0^{-1}$, excluding the diagonal, also consists of zeros.
APPENDIX B. PARTIAL DERIVATIVES FOR GRADIENT DESCENT METHOD

In this section the gradients are derived for the optimization problem where the objective $J$ is found using the positional estimates for each track. From Equation 3.6, if $N = 1$ then

$$J = \tilde{p}[t|t]^T \tilde{p}[t|t],$$  \hspace{1cm} (B.1)

where from Equation 3.4

$$
\begin{align*}
\tilde{p}[t|t] &= \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} \hat{p}_{j,n}^i[t|t] \\ \hat{p}_{j,e}^i[t|t] \end{pmatrix} - \begin{pmatrix} \beta_p^n \\ \beta_p^e \end{pmatrix} - \begin{pmatrix} \hat{p}_{i,n}^i[t|t] \\ \hat{p}_{i,e}^i[t|t] \end{pmatrix} \\
&= \begin{pmatrix} \cos \theta \hat{p}_{j,n}^i[t|t] + \sin \theta \hat{p}_{j,e}^i[t|t] - \beta_p^n - \hat{p}_{i,n}^i[t|t] \\ -\sin \theta \hat{p}_{j,n}^i[t|t] + \cos \theta \hat{p}_{j,e}^i[t|t] - \beta_p^e - \hat{p}_{i,e}^i[t|t] \end{pmatrix}.
\end{align*}
$$

This then implies that

$$J = (\cos \theta \hat{p}_{j,n}^i[t|t] + \sin \theta \hat{p}_{j,e}^i[t|t] - \beta_p^n - \hat{p}_{i,n}^i[t|t])^2 + (-\sin \theta \hat{p}_{j,n}^i[t|t] + \cos \theta \hat{p}_{j,e}^i[t|t] - \beta_p^e - \hat{p}_{i,e}^i[t|t])^2.$$

The partial derivative of the objective with respect to $\theta$ is then

$$\frac{\partial J}{\partial \theta} = 2(\cos \theta \hat{p}_{j,n}^i[t|t] + \sin \theta \hat{p}_{j,e}^i[t|t] - \beta_p^n - \hat{p}_{i,n}^i[t|t])(-\sin \theta \hat{p}_{j,n}^i[t|t] + \cos \theta \hat{p}_{j,e}^i[t|t]) + 2(-\sin \theta \hat{p}_{j,n}^i[t|t] + \cos \theta \hat{p}_{j,e}^i[t|t] - \beta_p^e - \hat{p}_{i,e}^i[t|t])(-\cos \theta \hat{p}_{j,n}^i[t|t] - \sin \theta \hat{p}_{j,e}^i[t|t]),$$

which can be expressed as

$$\frac{\partial J}{\partial \theta} = 2\tilde{p}[t|t]^T \frac{\partial \tilde{p}[t|t]}{\partial \theta},$$  \hspace{1cm} (B.2)
where

\[
\frac{\partial \tilde{p}[t|t]}{\partial \theta} = \begin{pmatrix} -\sin \theta & \cos \theta \\ -\cos \theta & -\sin \theta \end{pmatrix} \begin{pmatrix} \tilde{p}_{j,n}[t|t] \\ \tilde{p}_{j,e}[t|t] \end{pmatrix}.
\]

The partial derivative of the objective with respect to \( \beta_{p_n} \) is

\[
\frac{\partial J}{\partial \beta_{p_n}} = 2(\cos \theta \tilde{p}_{j,n}[t|t] + \sin \theta \tilde{p}_{j,e}[t|t] - \beta_{p_n} - \tilde{p}_{i,n}[t|t])(-1),
\]

which can be expressed as

\[
\frac{\partial J}{\partial \beta_{p_n}} = 2 \tilde{p}[t|t]^T \frac{\partial \tilde{p}[t|t]}{\partial \beta_{p_n}},
\]

where

\[
\frac{\partial \tilde{p}[t|t]}{\partial \beta_{p_n}} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}.
\]

The partial derivative with respect to \( \beta_{p_e} \) is similarly defined as

\[
\frac{\partial J}{\partial \beta_{p_e}} = 2 \tilde{p}[t|t]^T \frac{\partial \tilde{p}[t|t]}{\partial \beta_{p_e}},
\]

where

\[
\frac{\partial \tilde{p}[t|t]}{\partial \beta_{p_e}} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.
\]

Equations B.2 - B.4 can also be extended to the case where Equation B.1 is defined for \( N > 1 \) as in

\[
J = \tilde{p}[t]^T \tilde{p}[t],
\]
where

\[ \tilde{p}[t] = \begin{pmatrix} \tilde{p}[t - N + 1|t - N + 1] \\ \tilde{p}[t - N + 2|t - N + 2] \\ \vdots \\ \tilde{p}[t|t] \end{pmatrix}. \]
APPENDIX C. HARDWARE

The hardware used is presented in this section.

C.1 Camera

For all of the tests two GoPro cameras were used (HERO3 for $V_1$, HERO3+ for $V_2$). The resolution for each was set at $1280 \times 960$. Calibration images, similar to Figure C.1, were obtained from each camera and the intrinsic parameters were found using the MATLAB Computer Vision System Toolbox. The intrinsic parameter matrix $\kappa$ is expressed as

$$
\kappa = \begin{bmatrix}
  f_x & s & O_x \\
  0 & f_y & O_y \\
  0 & 0 & 1
\end{bmatrix},
$$

where $f_x$ and $f_y$ are representative of the focal length in pixels, $s$ is a skew factor, and $(O_x, O_y)$ is the pixel location of the camera’s optical center [45]. The camera matrices for $V_1$ and $V_2$ were

![Figure C.1: Calibration images were collected and used to calibrate each camera.](image)
calculated as

\[
\kappa_1 = \begin{bmatrix}
655.95 & 0 & 655.21 \\
0 & 658.35 & 472.48 \\
0 & 0 & 1
\end{bmatrix}
\] \quad \text{(C.1)}

\[
\kappa_2 = \begin{bmatrix}
610.63 & 0 & 654.03 \\
0 & 612.14 & 474.16 \\
0 & 0 & 1
\end{bmatrix}
\] \quad \text{(C.2)}

One thing to note is that for a true pinhole camera \( f_x = f_y \), which is taken as the focal length \( f \). This is nearly true for \( \kappa_1 \) and \( \kappa_2 \). However, to get a single value for \( f \) we simply take the average

\[
f = \frac{f_x + f_y}{2}.
\]

The intrinsic parameters (Equations C.1 and C.2) are used to undistort the video sequences. Results of this undistortion process can be seen in Figure C.2.

![Figure C.2: The original image (left) contains severe image distortion due to a fish-eye type lens. Using the intrinsic camera parameters the distortion can be removed (right).](image-url)
C.2 UAV Platform

For the UAV test (Section 4.2.2) each platform was a 3DR Y6 multicopter (see Figure C.3). The multicopters were flown manually and the onboard PixHawk autopilots recorded all necessary telemetry data. Each camera was mounted to a 3-axis brushless gimbal, which was stabilized by the BaseCam 32-bit Gimbal Controller. The gimbal angles were extracted using the gimbal controller’s serial API, and were recorded to an SD card via an Arduino Uno microcontroller.