2017-05-01

Integration of a Complete Detect and Avoid System for Small Unmanned Aircraft Systems

Jared Kevin Wikle
Brigham Young University

Integration of a Complete Detect and Avoid System for Small Unmanned Aircraft Systems

Jared Kevin Wikle

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

Master of Science

Timothy W. McLain, Chair
Randal W. Beard
Marc Killpack

Department of Mechanical Engineering

Brigham Young University

ABSTRACT

Integration of a Complete Detect and Avoid System
for Small Unmanned Aircraft Systems

Jared Kevin Wikle
Department of Mechanical Engineering, BYU
Master of Science

For unmanned aircraft systems to gain full access to the National Airspace System (NAS), they must have the capability to detect and avoid other aircraft. This research focuses on the development of a detect-and-avoid (DAA) system for small unmanned aircraft systems.

To safely avoid another aircraft, an unmanned aircraft must detect the intruder aircraft with ample time and distance. Two analytical methods for finding the minimum detection range needed are described. The first method, time-based geometric velocity vectors (TGVV), includes the bank-angle dynamics of the ownship while the second, geometric velocity vectors (GVV), assumes an instantaneous bank-angle maneuver. The solution using the first method must be found numerically, while the second has a closed-form analytical solution. These methods are compared to two existing methods. Results show the time-based geometric velocity vectors approach is precise, and the geometric velocity vectors approach is a good approximation under many conditions.

The DAA problem requires the use of a robust target detection and tracking algorithm for tracking multiple maneuvering aircraft in the presence of noisy, cluttered, and missed measurements. Additionally these algorithms needs to be able to detect overtaking intruders, which has been resolved by using multiple radar sensors around the aircraft. To achieve these goals the formulation of a nonlinear extension to R-RANSAC has been performed, known as extended recursive-RANSAC (ER-RANSAC). The primary modifications needed for this ER-RANSAC implementation include the use of an EKF, nonlinear inlier functions, and the Gauss-Newton method for model hypothesis and generation.

A fully functional DAA system includes target detection and tracking, collision detection, and collision avoidance. In this research we demonstrate the integration of each of the DAA-system subcomponents into fully functional simulation and hardware implementations using a ground-based radar setup. This integration resulted in various modifications of the radar DSP, collision detection, and collision avoidance algorithms, to improve the performance of the fully integrated DAA system. Using these subcomponents we present flight results of a complete ground-based radar DAA system, using actual radar hardware.

Keywords: Detect and avoid, sense and avoid, multiple target tracking, recursive-RANSAC, extended recursive-RANSAC, unmanned aircraft system, small UAS, minimum detection range, ground-based radar, radar, detection device, collision detection, collision avoidance, self separation, path planning

ACKNOWLEDGMENTS

First, I want to thank Utah NASA EPSCoR for sponsoring my research. I would also like to thank the Center for Unmanned Aircraft Systems (C-UAS) with support from the National Science Foundation I/UCRC program grant numbers IIP-1161036 and IIP-1171036 and C-UAS Industry Advisory Board members.

I would also like to take a moment to thank all those who have personally aided me in the completion of my thesis. I would first like to thank my advisor Dr. McLain for the direction and patience he has shown me during the numerous stages of my research. I would also like to thank Dr. McLain and my other committee members Dr. Beard and Dr. Killpack, for their valuable technical insights in the research I have performed.

Special thanks are given to Laith Sahawneh, Kaleo Roberts, and Luke Newmeyer for their collaborative efforts they have provided in the development of a complete ground-based radar DAA system using radar hardware. I would also like to thank Jesse Wynne and Brandon Reimschissel for serving as flight test pilots at various stages of my research. I would also like to thank other students who have helped throughout my graduate studies including Peter Niedfeldt, Matt Duffield, Jonathan Spencer, Michael Boren, Gary Ellingson, David Wheeler, and many others.

Finally, I want to express my sincere gratitude to my family and friends who have supported me throughout my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **vi**

LIST OF FIGURES . . . . . **vii**

Chapter 1 Introduction **1**
1.1 Detect-and-Avoid Overview 1
1.2 Summary of Contributions 5
1.3 Thesis Outline 6

Chapter 2 Minimum Detection Range **8**
2.1 Minimum Detection Range Formulation 9
2.1.1 Prior Approaches 12
2.1.2 New Approaches 15
2.2 Results: Method Comparison and Validation 32

Chapter 3 Target Detection and Tracking **44**
3.1 Detection Device 44
3.2 Tracker 46
3.2.1 Desired Tracker Properties 47
3.2.2 Overview of Recursive-RANSAC 54
3.2.3 Extended Recursive-RANSAC (ER-RANSAC) 60
3.2.4 General Improvements Made to R-RANSAC 79
3.3 Implementation of ER-RANSAC Using On-board Radar 82
3.3.1 Nonlinear Constant Acceleration Model 82
3.3.2 On-board Radar Sensors 83
3.3.3 Observability of Nonlinear System 85
3.3.4 Gauss-Newton Results 86
3.3.5 Simulation Results 90

Chapter 4 Complete DAA System Using GBR **101**
4.1 Ground-based Phased-array Radar 104
4.2 Constant Acceleration R-RANSAC Tracker 111
4.3 Collision Detection 115
4.4 A*/Chain-based Collision Avoidance 125
4.5 Simulation Results 128

Chapter 5 DAA Flight Results Using Radar Hardware **143**
5.1 Ground-based Control Station 143
5.2 10 GHz FMCW Radar and BOARAC Processing Board 146
5.3 2D Modifications of DAA Algorithms 158
5.4 Hardware Results 159

Chapter 6 Conclusions and Future Work **169**
6.1 Summary 169
6.2 Recommendations for Future Work 172
6.2.1 Minimum detection range 172
6.2.2 Tracking 173
6.2.3 Complete DAA 174

REFERENCES **176**

Appendix A TGVV Bank-Angle Response **180**
A.1 Case A: $\phi_{\max}$ is reached 180
A.2 Case B: $\phi_{\max}$ is not reached 184

Appendix B Nonlinear Observability **187**
B.1 Matrix A 190
B.2 Matrix C 190
B.3 Matrix F 192

Appendix C General Purpose DAA Simulator **194**
C.1 Desired Capabilities 194
C.2 Previous Simulator 195
C.3 Improved Simulator 196
C.3.1 Capabilities Achieved 196
C.3.2 Organizational Structure 197

LIST OF TABLES

2.1 Bank angle, time intervals, and change in course for each segment of Case A. . . 23
2.2 Bank angle, time intervals, and change in course for each segment of Case B. . . 23
2.3 Nominal parameter values used in the calculation of d_{MDR}. 32
2.4 Computational cost of each method from average runtime and lines of code. . . 43

3.1 Algorithm for RANSAC 57
3.2 Algorithm for R-RANSAC 59
3.3 R-RANSAC parameters. 96
3.4 RANSAC parameters. 97
3.5 Gauss-Newton parameters. 97

4.1 Algorithm for collision detection 118
4.2 Algorithm to find the entrance and exit times horizontally. 121
4.3 Algorithm to find the entrance and exit times vertically. 124
4.4 Collision avoidance parameters. 130
4.5 R-RANSAC parameters. 133
4.6 Length of the avoidance path. 140
4.7 Radar FPGA run time. 140
4.8 Run times for detect and avoid algorithms. 141
4.9 Run times for combined detect and avoid algorithms. 141

5.1 Radar Sensor Parameters 147
5.2 Collision avoidance parameters. 163
5.3 R-RANSAC parameters. 166

LIST OF FIGURES

2.1 Proposed time line for the detect and avoid system [1]. 10
2.2 General volume used to represent the well-clear or NMAC conditions. 11
2.3 The total minimum detection range, d_{MDR}, needed. A representation of the closest point of approach (CPA). 16
2.4 Block diagram of the bank-angle response due to aileron inputs. 20
2.5 Bank-angle response to aileron step inputs. 21
2.6 Geometric diagram for the CPA location resulting from circular turning trajectories of the ownship. 26
2.7 Comparison of d_{MDR} as a function of v_o, v_i, R_s, $\phi_{\max}$, t_c, χ_t, $\dot{\phi}_{\max}$ and τ. 34
2.8 Relative range vs. time and the resulting CPA. 36
2.9 Comparison of the CPA as a function of v_o, v_i, R_s, $\phi_{\max}$, t_c, χ_t, $\dot{\phi}_{\max}$ and τ. . . . 37
2.10 Percent relative error of d_{MDR}, compared to TGVV, as a function of v_o, v_i, R_s, $\phi_{\max}$, t_c, χ_t, $\dot{\phi}_{\max}$ and τ. 39
2.11 Comparison of t_m as a function of v_o, v_i, R_s, $\phi_{\max}$, t_c, χ_t, $\dot{\phi}_{\max}$ and τ. 41
2.12 Self separation results. 42

3.1 Full 360 degree horizontal coverage achieved with three radar units. 49
3.2 High level illustration of nonlinear version of RANSAC using the Gauss-Newton method. 65
3.3 Evolution of states across each iteration of the Gauss-Newton method. 87
3.4 The true trajectory, minimum subset of radar measurements, and resulting Gauss-Newton trajectory. 89
3.5 The true trajectory and Gauss-Newton trajectory shown in 2D. 90
3.6 Flight paths of the ownship and three intruder aircraft. 91
3.7 Flight paths shown in 2D. 92
3.8 Flight paths and radar measurements shown in the inertial reference frame. . . . 94
3.9 Flight paths with radar measurements shown in 2D. 94
3.10 Radar measurements in ownship body frame. 95
3.11 ER-RANSAC position estimates. 98
3.12 ER-RANSAC velocity estimates. 99
3.13 ER-RANSAC acceleration estimates. 100

4.1 Ground-based radar detect-and-avoid system structure diagram. 103
4.2 Surveillance and operating volumes associated with the ground-based radar DAA system. 105
4.3 Radar surveillance volume using planar phased array. 106
4.4 The mean power in each range bin over a discrete time window. 108
4.5 Histogram of range bin 150 across 557 time samples. 108
4.6 Threshold using a constant $P_{FA} = 0.01$. 110
4.7 Final thresholded radar data in each range bin over a discrete time window. . . 111
4.8 Covariance of radar measurements. 114
4.9 Horizontal encounter scenarios. 122
4.10 Vertical encounter scenarios. 125

4.11 Encounter scenario number 1. 129
4.12 The avoidance path of the ownship. 129
4.13 Avoidance path followed by the ownship in encounter scenario number 1. 130
4.14 Radar measurements: range, azimuth and elevation. 131
4.15 Aircraft's paths with radar measurements in encounter scenario number 1. . . . 132
4.16 R-RANSAC tracks: position estimates of aircraft. 133
4.17 R-RANSAC tracks: velocity estimates of aircraft. 134
4.18 Relative range and altitude to intruders. 134
4.19 Encounter scenario number 2. 135
4.20 The avoidance path of the ownship. 136
4.21 Avoidance path followed by the ownship in encounter scenario number 2. 136
4.22 Radar measurements: range, azimuth and elevation. 137
4.23 Aircraft's paths constructed using radar measurements in encounter scenario number 2. 137
4.24 R-RANSAC tracks: position estimates of aircraft. 138
4.25 R-RANSAC tracks: velocity estimates of aircraft. 139
4.26 Relative range and altitude to intruders. 139

5.1 Implementation of ground-based radar control station with hardware. 144
5.2 Portable phased-array radar system designed for use on-board UAS. 148
5.3 Variable probability of false alarm as a function of range bin. 151
5.4 Grouping and peak identification. 152
5.5 Spline fitting. 153
5.6 Histogram of x_{diff}. 154
5.7 Histogram of the absolute value of x_{diff}. 155
5.8 Histogram after converting the exponential distribution into a symmetric beta distribution. 156
5.9 Beta function cdf. 157
5.10 Histogram after converting the beta distribution into a uniform distribution. . . 157
5.11 Final uniformly distributed histogram. 158
5.12 3DR X8 and Y6 aircraft used in flight test with corner reflectors. 159
5.13 Sketch of the encounter geometry of flight test (not to scale). 160
5.14 Ground control station with radar and three aircraft at the start of the test. . . 161
5.15 Encounter geometry of ground-based DAA flight test. 162
5.16 The avoidance path of the ownship and the paths of the intruders in the FR inertial frame centered about the home location. 163
5.17 Radar measurements: range, and azimuth of ownship and intruders. 164
5.18 Aircraft's paths constructed using radar measurements. 165
5.19 R-RANSAC tracks: position estimates of aircraft. 166
5.20 R-RANSAC tracks: velocity estimates of aircraft. 167
5.21 Aircraft's paths constructed using radar measurements and R-RANSAC position estimates. 167
5.22 Relative range to intruders. 168

C.1 Top level of general DAA simulator. 198

C.2 Autopilot structure used for the ownship and intruder aircraft. 199
C.3 Contents of MAV block. 200
C.4 All blocks used for the ownship including the target detection and tracking block and the collision avoidance planner block. 201
C.5 Contents of TargetDetectionAndTracking block. 201
C.6 Location of configurable subsystem within main Simulink block. 202
C.7 Configurable subsystem defined in a Simulink library, used to choose between 4 function types. 203

CHAPTER 1. INTRODUCTION

1.1 Detect-and-Avoid Overview

The use of unmanned aircraft systems (UAS) in commercial and civil applications has been expanding rapidly in recent years. Many UAS missions will require simultaneous operation with existing airspace users. UAS currently have limited access to the National Airspace System (NAS) because they do not have the ability to detect and avoid other air traffic. Among many regulatory and technology issues, safety is the foremost concern and the most significant challenge to overcome before UAS integration into the NAS can be achieved. The Federal Aviation Administration (FAA), the national aviation authority in the United States, calls for a target level of safety comparable to the see and avoid requirement for manned aircraft [2].

Robust and reliable detect-and-avoid (DAA) systems will be necessary for UAS to provide the required target level of safety. Typically, a complete functional detect-and-avoid system is comprised of detection devices and associated trackers, collision detection, risk assessment, collision avoidance, and self-separation algorithms. Historically, a common term used in place of detect-and-avoid is the term sense-and-avoid (SAA). The motivation for using the term detect-and-avoid over sense-and-avoid comes from the fact that any detection device can be used for gathering the state information of another aircraft, and that measurement sensors need not be required. Throughout the remainder of this thesis we will follow the naming convention of DAA, as it pertains to a more general body of work.

The design of a DAA system for UAS should also address regulatory requirements, and performance and reliability standards. Initial efforts to address performance, design, construction, and reliability requirements of DAA systems for UAS are discussed in the white papers produced by RTCA SC-228 [2]. An excellent review of existing regulations, standards, recommended practices along with suggestions and recommendations for DAA requirements

to facilitate the UAS integration into the NAS system are discussed in Refs. [3–5]. Specific design parameters required by a DAA system, such as sensor angular resolution, field of view, and minimum time and detection range needed to prevent a collision assuming a 2D head-on encounter geometry, are addressed in Ref. [1]. Sensor and tracking requirements are derived for a radar-based DAA system considering a 2D flight worst-case encounter scenarios using exhaustive Monte Carlo simulations in Ref. [6]. The authors in Ref. [7] propose a framework that consists of a target level of safety (TLS) approach using an event-tree format to develop specific DAA effectiveness standards linking UAS characteristics and operating environments to midair-collision risk quantified by a fatality rate.

One problem that arises in creating a DAA system for UAS comes from the fact that these aircraft have a much larger range of the sizes and performance capabilities than manned aircraft [8]. These constraints of size, weight and power (SWaP) make DAA extremely challenging using existing sensor technologies [9].

Another, more fundamental problem, is moving any DAA system to the flight testing stage regardless of if it meets the SWaP constraints or not. One system that has reached this flight testing stage has been developed by the Air Force Research Laboratory (AFRL) [10]. To test their DAA system, the AFRL used a Calspan Corporation Learjet in-flight simulator aircraft. The Learjet was equipped with a radar, electro-optical (E/O) camera sensors, and a Traffic Collision Avoidance System (TCAS) unit. Under the Multiple Intruder Autonomous Avoidance (MIAA) program the Learjet was able to avoid multiple intruders in various circumstances. The overall DAA algorithm they used to deal with multiple sensor measurements is known as the Multiple Sensor Integrated Conflict Avoidance (MuSICA) algorithm. There are two main sub-algorithms of MuSICA. The first sub-algorithm is Sensor Data Integration (SDI), and is used for tracking multiple targets [11]. The second sub-algorithm is Jointly Optimal Conflict Avoidance (JOCA), and is used for avoidance of multiple intruders. SDI is comprised of four modules: EKF, data association, FTF, and track management. The data association module uses the Jonker-Volgenant-Castanon (JVC) algorithm. The JVC does not perform well in high-target-density and cluttered environments; therefore, it can only by used high above the ground with relatively few intruder aircraft. While this DAA system developed by the AFRL and other DAA systems exist, they do not meet the SWaP

constraints required by small UAS; therefore, a new DAA system needs to be developed for small UAS.

Within the Center for Unmanned Aircraft Systems (C-UAS), significant research efforts have been put forth in the development of a complete DAA system. Past efforts have included a system that includes both hardware and simulated components, however, the primary detection device has remained as a simulated component. Research efforts have continued to be made to develop a DAA system using actual radar hardware as the primary measurement sensor, providing both range and bearing of the intruder aircraft. This system has been divided among three different groups located at Brigham Young University (BYU). One group is developing a radar unit which will meet the SWaP constraints for a small UAS. The second group is working to support the radar hardware by creating a custom analog-to-digital (A/D) converter board that interfaces with the radar boards and an off-the-shelf development board. This development board contains a field-programmable gate array (FPGA) and a processor which are meant to process the radar returns and eventually run the remaining DAA algorithms. The final group, located within BYU's Multiple Agent Intelligent Coordination and Control (MAGICC) lab, is developing the target detection and tracking, collision detection, and collision avoidance algorithms.

One question that arises in the development of a DAA system is: what is the minimum distance that the detection device needs to detect other aircraft for effective self separation and collision avoidance? In this thesis, we create the time-based geometric velocity vectors (TGVV) and geometric velocity vectors (GVV) methods for estimating the minimum detection range. The TGVV method includes the bank-angle dynamics of the ownship, which results in an exact solution that is found with the aid of numerical techniques. The GVV method assumes an instantaneous bank-angle maneuver of the ownship, allowing us to develop an approximate closed-form analytical solution. These methods are compared to two existing methods, which we refer to in this thesis as the turn time (TT) and geometric tangent (GT) methods. Results show the time-based geometric velocity vectors approach is precise, the geometric velocity vectors approach is a good approximation under many conditions, and the two existing approaches are good approximations at large ownship speeds

relative to the intruder speed, fast ownship bank-angle transients, and small ownship bank angles.

The DAA problem requires the use of a robust target detection and tracking algorithm for tracking multiple maneuvering aircraft in the presence of noisy, cluttered, and missed measurements. Additionally this algorithms needs to be able to detect overtaking intruders. To achieve these goals, we have developed a a target detection and tracking system that uses radar as the primary sensor, where multiple radar sensors are mounted around the ownship aircraft to achieve 360 degrees of horizontal surveillance coverage. These radar sensor measurements are then used within a multiple target tracking algorithm to estimate the states of the intruder aircraft. The tracking algorithms that we use include, recursive-RANSAC (R-RANSAC) and extended recursive-RANSAC (ER-RANSAC). The R-RANSAC algorithm is developed in Ref. [12], while the general formulation of the ER-RANSAC algorithm is shown in this thesis, which extends the general linear R-RANSAC algorithm to nonlinear systems. In both the R-RANSAC and ER-RANSAC algorithms, we use a constant acceleration model to track the maneuvering behavior of the intruder aircraft. The model used within R-RANSAC is a translational point mass model, while the model used within ER-RANSAC is a nonlinear model that includes the flight characteristics of a turning, accelerating, and climbing aircraft. The nonlinear model more accurately captures the turning and climbing behavior of aircraft, and is better suited to predict where a maneuvering aircraft will be at some future point in time. While the nonlinear model has these added benefits, there are other factors that make the linear model an attractive candidate that will be discussed later. As such, both the linear and nonlinear models will be used in various results presented in this thesis. To aid in the generation of these and other results, we use an improved detect-and-avoid simulator that is capable of modeling multiple intruder aircraft specified by the user. Additionally, this simulator allows the user to easily switch between different algorithms, and allows the user to add new algorithms needed for development.

A complete DAA system includes target detection and tracking, collision detection, and collision avoidance. In this research we demonstrate the integration of each of the DAA-system subcomponents into fully functional simulation and hardware implementations using a ground-based radar (GBR) setup. This integration resulted in various modifications

of the radar DSP, collision detection, and collision avoidance algorithms, to improve the performance of the fully integrated DAA system. The ultimate goal is to create a complete DAA system that is fully autonomous and which uses an air-based detection device, however, to achieve this goal, we have first chosen to utilize the radar hardware in a ground-based radar DAA setup. This is an important step as it allows for greater debugging capabilities when trying to develop a completely new sensor to detect small UAS with a small radar cross section (RCS). Using this radar hardware and the other integrated DAA subcomponents, we present flight results of a complete ground-based radar DAA system. By using the radar in a ground-based setup, we demonstrate the feasibility of the hardware and algorithms developed in this thesis and by members of the research team.

1.2 Summary of Contributions

The main contributions of this thesis relate to target detection and tracking elements of the DAA problem, and the integration of subcomponents of the DAA system into fully functional simulation and hardware implementations. The specific contributions of this thesis include:

- The derivation of the TGVV and GVV methods, which are an exact numerical solution and a closed-form analytical approximation to the minimum detection range, respectively.
- An extensive comparison of the TGVV and GVV methods with two existing methods: the TT and GT methods. We find that the TGVV and GVV methods create more accurate estimates of the minimum detection range over a wide range of encounter scenarios.
- A general formulation of the ER-RANSAC algorithm, which extends the general linear R-RANSAC algorithm to nonlinear systems.
- The implementation of a target detection and tracking system for robustly tracking intruder aircraft in the varied encounters a UAS might possibly experience. This system

uses multiple radar units for increased sensor coverage, and either the R-RANSAC or ER-RANSAC algorithms implemented with constant-acceleration dynamic models.

- An improved detect-and-avoid simulator that is capable of modeling a user specified number of intruder aircraft, provides an elegant way for the user to choose between alternate function implementations, and allows the user to easily add new function implementations.

- The integration of the DAA-system subcomponents into fully functional simulation and hardware implementations. This includes modifications to the radar DSP, collision detection, and collision avoidance algorithms, to improve the performance of the fully integrated DAA system.

- Flight testing of a complete ground-based radar DAA system, using actual radar hardware that demonstrates the feasibility of the hardware and algorithms developed in this thesis and by members of the research team.

1.3 Thesis Outline

The outline of the remainder of this thesis is organized as follows. In Chapter 2, we create the TGVV and GVV methods for estimating the minimum detection range. In this chapter, we also compare the TGVV and GVV methods with two existing methods, including the TT and GT methods. In Chapter 3, we describe a novel method for robustly detecting and tracking intruder aircraft in the varied encounters a UAS might possibly experience. Chapter 3 is divided into three main sections. First, we describe the selection of radar as the primary detection device. Second, we describe the desired properties of a tracking system and how we have achieved these properties through the use of R-RANSAC with a constant-acceleration model. Also in this section, we present the general formulation of the ER-RANSAC algorithm, which extends the general linear R-RANSAC algorithm to nonlinear systems. Third, we demonstrate the performance of this tracking system using the ER-RANSAC algorithm and three on-board radar sensors. In Chapters 4 and 5, we describe the integration of the DAA-system subcomponents into fully functional simulation

and hardware implementations, respectively. For the simulation implementation we model a 3D ground-based radar and 3D aircraft flight paths. For the hardware implementation we describe modifications that needed to be made in the radar DSP, R-RANSAC, collision detection, and collision avoidance algorithms, in order to be able to use actual radar hardware that only provides measurements of range and azimuth angle. We also describe the ground-based control station and all other hardware which were necessary to create flight test results. Chapter 6 presents our conclusions and possibilities for future work.

CHAPTER 2. MINIMUM DETECTION RANGE

Robust and reliable DAA systems will be necessary for UAS to provide the required target level of safety called for by the FAA. These DAA systems typically include sensors and associated trackers, collision detection, risk assessment, collision avoidance, and self-separation algorithms.

The main role of the sensor and associated tracker is to detect any of the various types of hazards, such as traffic or terrain, and track the motion of the detected object to gain sufficient confidence that the detection is valid. Electro-optical (E/O) and infrared (IR) cameras, light detection and ranging (LiDAR), and radar are examples of sensors employed to detect non-cooperative traffic [13]. Non-cooperative traffic means that data about potential conflicts is not communicated or transmitted to the ownship UAS from the conflicting intruders. The traffic alert and collision avoidance system (TCAS) and automatic dependent surveillance-broadcast (ADS-B) are examples of systems for detecting cooperative intruders. Both cooperative and non-cooperative sensors can be used with DAA systems and both can be used in conjunction with the results of this chapter.

Not every aircraft that is observed by the detection system presents a risk of collision or violation of the well-clear boundary. The self-separation algorithm must, therefore, identify potential well-clear violations and plan new paths that remain well clear of intruder aircraft while optimizing an objective function or performance metric. Self separation is designed to prevent future collision-avoidance maneuvers and is achieved using less-aggressive maneuvers. If the well-clear boundary is penetrated, the collision detection and collision avoidance system must be used to detect potential collisions and compute collision-free paths. A collision avoidance maneuver is considered as the last resort effort to steer the UAS onto a safe course to prevent an imminent collision and may require an aggressive change in flight path.

The design of a DAA system for UAS should also address regulatory requirements, and performance and reliability standards. An overview of these requirements and standards was initially shown in Chapter 1. Among these requirements and design specifications, developing sensors that achieve sufficiently large target detection ranges for effective self separation and collision avoidance is a crucial aspect of a viable DAA solution. This chapter provides an exact numerical solution and a closed-form analytical approximation to the minimum detection range. An outline of this chapter is as follows. In Section 2.1, derivation of the minimum detection range d_{MDR} is given along with the appropriate definitions and assumptions. An overview of two existing methods is given in Section 2.1.1 followed by the new methods in Section 2.1.2. In Section 2.2 results are presented providing a comparison of the four methods: the two presented in this chapter, and those presented in Ref. [1] and [14].

2.1 Minimum Detection Range Formulation

The minimum required detection range arises from the time required to complete the detection and avoidance of an intruder. The minimum time for the DAA system to be able to track the intruder, detect a collision or well-clear violation, plan an avoidance maneuver, wait for human review/approval, and fly the maneuver determines the distance at which the UAS must detect the intruder. The detection of a well-clear violation or collision threat must be accomplished at no less than the minimum detection range to allow the ownship to execute the maneuver with sufficient time so that the closest point of approach is greater than or equal to the separation requirement. A time sequence for the DAA system, similar to the proposed sequence in Ref. [1], is shown in Fig. 2.1. According to the time sequence, the time required for DAA operations, t_{DAA}, is defined as the sum of the computation time, t_c, and the time that is required to maneuver, t_m.

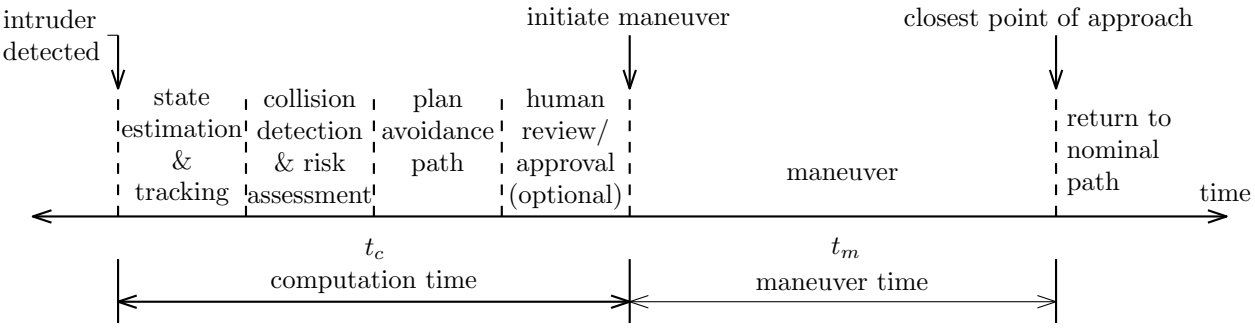


Figure 2.1: Proposed time line for the detect and avoid system [1].

Current manned aviation regulations have no explicit values for separation requirements, however, various attempts have been made to define them. For collision avoidance of manned aviation, a common requirement is the near mid-air collision (NMAC) volume, which is a disk-shaped volume with a horizontal radius of 500 ft and a vertical height of 200 ft [15–17]. No such requirements exist for UAS, however, the NMAC volume could be used as a conservative requirement. For self separation, the well-clear boundary is more ambiguous and depends on the specific aircraft involved and their associated speeds and altitudes. Recent efforts to define this boundary for UAS have resulted in values in the range of 0.5 to 1 nmi [18]. For UAS the potential ownship and intruder aircraft can vary widely in vehicle size, weight, and airspeed, and the separation requirements could be scaled accordingly. For this work, we will assume a purely geometric safety volume centered around the aircraft as shown in Fig. 2.2. The general choice for this volume is a cylinder of radius R_s and height h_s centered at the current location of the aircraft. This volume will be used to represent the well-clear or NMAC volume. From this volume, a well-clear violation or an NMAC is defined as an incident that occurs when two aircraft pass with a distance less than R_s horizontally and $\frac{h_s}{2}$ vertically.

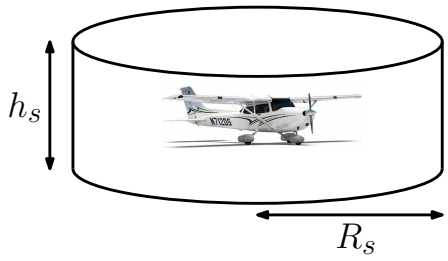


Figure 2.2: General volume used to represent the well-clear or NMAC conditions.

Various types of sensors exist for detecting intruder aircraft and they can be located at different locations, such as on the ownship aircraft, a stationary ground-based platform, or a moving ground based platform. If the sensor is fixed to the ownship, the minimum detection range calculated in this paper can be directly used as the minimum sensing range requirement of the sensor. If the sensor is used with a ground-based system, the minimum detection range determines how close an aircraft can fly to the edge of maximum surveillance range and still be guaranteed to avoid intruders. The equations developed in this paper for the minimum detection range are independent of the specific sensor type chosen, however, because of field of view limits, multiple sensors may be required to achieve the desired field of view.

The analysis in this chapter assumes that there is only one intruder aircraft. If there is more than one intruder then the avoidance maneuver will likely be more complex and will require a greater detection range. This paper considers the longest-detection-range encounter that occurs when both the ownship and intruder are flying at a constant altitude, course, and airspeed in a direct head-on approach. Various types of avoidance maneuvers can be taken by the ownship to avoid the intruder's safety volume. First, the ownship could perform a turning maneuver at a constant altitude to stay outside the horizontal safety radius R_s. Second, the ownship could perform a climbing/descending maneuver without turning to achieve a relative altitude equal to or greater than $h_s/2$ while inside the safety radius. Third, the ownship could both turn and climb/descend simultaneously to avoid either the safety radius or the safety altitude, whichever comes first. Finally, the ownship could adjust its speed in addition to one of the previous maneuvers. Since R_s is generally much larger

than h_s, a constant-altitude turning maneuver will require the largest distance to avoid the safety volume and will be the focus of our avoidance-maneuver analysis. For simplicity, the ownship velocity is assumed to be constant and above the stall speed of the ownship. The speed of the ownship and intruder are defined as v_o and v_i respectively.

In this analysis, the ownship's turning dynamics follow the coordinated-turn relationships [19]

$$\dot{\chi} = \frac{g}{v_o} \tan \phi, \tag{2.1}$$

$$R_{\min} = \frac{v_o}{\dot{\chi}_{\max}} = \frac{v_o^2}{g \tan \phi_{\max}}, \tag{2.2}$$

where $\dot{\chi}$ is the course rate, ϕ is the bank angle, $R_{\min}$ is the minimum turning radius, $\dot{\chi}_{\max}$ is the maximum course rate, $\phi_{\max}$ is the maximum bank angle, and g is the gravitational constant. This analysis is intended for use with fixed-wing aircraft with turning dynamics that are well modeled by the coordinated-turn relationships. The analysis is also valid for other types of aircraft, such as rotorcraft, provided their turn dynamics are approximated by the coordinated turn.

The assumptions that have been made and other real-world issues may limit the validity of the analysis and as a result additional range may be required in practice. Ownship and intruder states are not known perfectly due to state estimation errors resulting from imperfect IMU, GPS, and detection sensors. The intruder may maneuver during the encounter by turning, changing altitude, or changing speed. Finally, the ownship may not follow the ideal coordinated turn flight trajectory. Each of these issues will need to be considered before final sensor requirements can be made, however, the results of this paper provide a solid foundation to build upon.

2.1.1 Prior Approaches

One method, known as the tactical separation assisted flight environment (TSAFE) resolution algorithm developed by Erzberger [20], deserves mention. The TSAFE resolution algorithm does not solve for the minimum detection range specifically, however, various supplemental equations are common between our method and the TSAFE resolution algorithm

including the position and course dynamics of the ownship. In the TSAFE resolution algorithm, two aircraft are given arbitrary positions, headings, and speeds relative to each other. Using these initial conditions, resolution maneuvers are calculated based on one or both of the aircraft performing a horizontal turning maneuver followed by straight-line flight. The turning maneuvers are performed by right or left turn maneuvers with the bank angle and turn time as the control variables. By varying each of the variables, equations are developed to calculate the closest point of approach (CPA) between the two aircraft during the turning maneuver and during the straight-line flight. The final CPA between the two aircraft is then found find taking the minimum of the CPA during the turning maneuver and the straight-line portion. The last step of the TSAFE resolution algorithm is to choose the appropriate bank angle, turn direction, and turn time which result in the CPA being greater than the required safe distance. Although the TSAFE resolution algorithm was not formulated to find the minimum detection range, the CPA equations utilized by TSAFE resolution could be used in an iterative manner to solve for the minimum detection range by adjusting the starting distance between the two aircraft until the CPA equals the required safe distance.

Two approaches developed specifically for calculating minimum detection range are found in the literature. One approach, developed by Geyer et al. [1], is referred to in this paper as the turn-time (TT) approach. The second approach, developed by Sahawneh et al. [14], is referred to as the geometric-tangent (GT) approach. Both approaches assume an instantaneous bank-angle maneuver and a head-on, constant-altitude, constant-velocity encounter. A brief description of these two methods is given in the following subsections.

Turn Time Approach

Using the TT method proposed in Ref. [1], the minimum detection range is calculated using the expression

$$d_{\mathrm{MDR}} = (v_o + v_i)(t_c + t_t), \tag{2.3}$$

where t_t is the time when the ownship and intruder are closest as the ownship is executing a turning maneuver. The turn time is found using equations for the north and east positions of both the ownship and intruder as functions of time, and the assumption that the turn

time is approximately equal to the time to collision in the absence of a maneuver. Using this time-based approach the minimum detection range is expressed as

$$d_{\text{MDR}} \approx (v_o + v_i)\left(t_c + \sqrt{\frac{2R_s \cot \phi_{\max}}{g}}\right). \tag{2.4}$$

In Ref. [1], the authors acknowledge that their solution is an approximation to the true minimum detection range, and state that it is meant to be used as a heuristic for choosing the right sensor and its resolution. They further state that it is not suitable for small distances and velocities, but did not specifically define limiting values.

Geometric Tangent Approach

The GT method proposed in Ref. [14] approximates the minimum detection range as

$$d_{\text{MDR}} = (v_o + v_i)t_c + d_m,$$

where d_m is the distance between the ownship and the intruder when the ownship starts maneuvering. In this method the ownship executes a turning maneuver with a constant turning radius. The closest point of approach is then assumed to occur when the ownship is located on the edge of the safety circle around the intruder and the turning radius of the ownship is tangent to the safety circle around the intruder. Using this geometric relationship the minimum detection range is expressed as

$$d_{\text{MDR}} \approx (v_o + v_i)t_c + \sqrt{R_s^2 + 2R_s\frac{v_o^2}{g\tan\phi_{\max}}} + \frac{v_i v_o}{g\tan\phi_{\max}}\cos^{-1}\left(\frac{v_o^2}{v_o^2 + R_s g\tan\phi_{\max}}\right). \tag{2.5}$$

In Ref. [14], the authors acknowledge that their solution is an under-approximation and state that compensation can be made by selecting a positive slack parameter, δ_r, to obtain $\bar{d}_{\text{MDR}} = (1 + \delta_r)d_{\text{MDR}}$. Slack parameters must be found using experimental results, such as those from Monte Carlo simulations, but insights into appropriate slack parameter values were not given.

2.1.2 New Approaches

The TT and GT approaches both make the assumption that the ownship executes a turning maneuver by instantaneously banking to a specified angle. This assumption simplifies the derivation of the minimum detection range by confining the trajectory of the ownship to a circular arc. In the turning maneuver of real aircraft, an instantaneous bank-angle maneuver is not physically possible. Instead, the bank angle has a transient response resulting from the deflection of the ailerons. The response of the bank angle determines the course rate of a coordinated turn as shown in Eq. (2.1). This non-constant course rate results in a trajectory that is not circular and must be determined by numerical integration of complex time-based functions. Considering the bank-angle dynamics results in a more accurate prediction of the minimum detection range that is larger than the predictions of the TT and GT methods. The proposed method that takes into account the bank-angle dynamics is called the time-based geometric velocity vectors (TGVV) approach and is described fully in this paper.

The second method presented in this paper, known as the geometric velocity vectors (GVV) approach, is a special case of the TGVV approach that maintains the assumption of an instantaneous bank-angle maneuver. By utilizing this assumption, the GVV method allows flight trajectories to be represented geometrically as circular paths instead as functions of time, which allows a closed-form analytical solution for the minimum detection range to be derived. We will show under what conditions, the instantaneous bank-angle assumption is valid, allowing the GVV method to be used with confidence. We will further show that the GVV method produces more accurate approximations to the TGVV solution over a wider range of conditions than the solutions offered by the TT and GT approaches. The TGVV and GVV methods also allow the turning angle for the avoidance maneuver to be defined by the analyst instead of prescribing a 90-degree turn. While this may result in a slightly larger d_{MDR}, it also allows deviations from the nominal flight path to be reduced at the discretion of the analyst.

Problem Formulation

A diagram for the total minimum detection range, d_{MDR}, is shown in Fig. 2.3, and the resulting general equation for d_{MDR} is represented as

$$d_{\text{MDR}} = d_o + d_i + d_{\text{CPA}},$$

where d_o is the total head-on distance traveled by the ownship, and d_i is the total head-on distance traveled by the intruder. The final term d_{CPA} is the remaining head-on distance between the ownship and intruder when the closest point of approach has been reached.

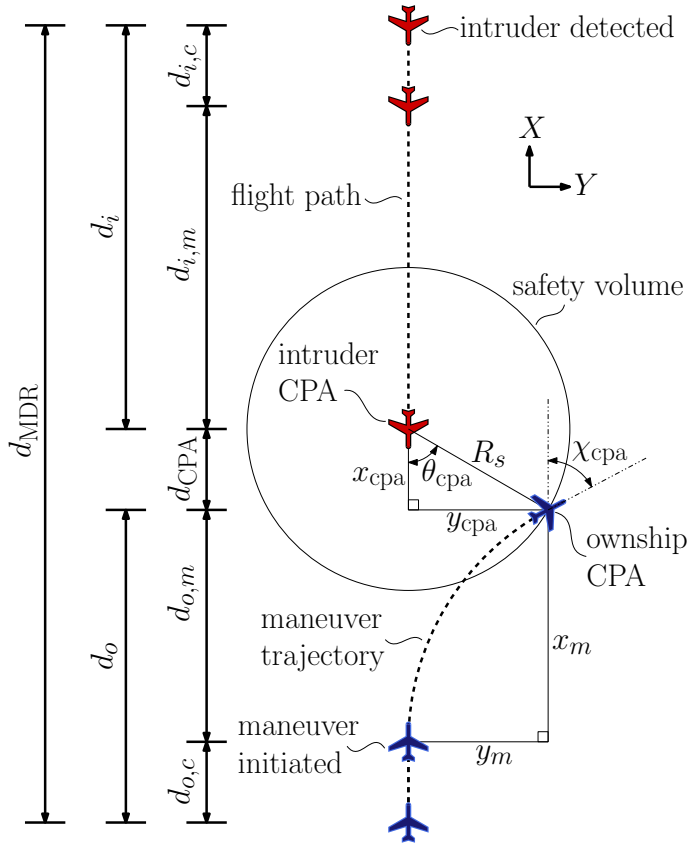


Figure 2.3: The total minimum detection range, d_{MDR}, needed. A representation of the closest point of approach (CPA).

In Fig. 2.3 we can also see that the variables d_o and d_i can both be broken down into two subsegments each. The total head-on distance traveled by the ownship can be defined as $d_o = d_{o,c} + d_{o,m}$, where $d_{o,c}$ is the head-on distance traveled by the ownship during computation time, and $d_{o,m}$ is the head-on distance traveled by the ownship while it is executing its maneuver. Similarly the total head-on distance traveled by the intruder can be defined as $d_i = d_{i,c} + d_{i,m}$, where $d_{i,c}$ is the head-on distance traveled by the intruder during computation time, and $d_{i,m}$ is the head-on distance traveled by the intruder while the ownship is executing its maneuver. Using these definitions, the minimum detection range becomes

$$d_{\text{MDR}} = d_{o,c} + d_{o,m} + d_{i,c} + d_{i,m} + d_{\text{CPA}}.$$

When analyzing the interaction between two aircraft during an avoidance maneuver, the CPA is a significant point of interest and a detailed derivation of this location is the basis upon which the new methods are built. The previous GT approach also uses the CPA as the basis of their derivation of the minimum detection range, however, the GT approach differs in where the CPA is located. In the GT method the CPA is assumed to be located where the circular arc transversed by the ownship during its avoidance maneuver becomes tangent to the safety circle drawn around the intruder. In the TGVV and GVV methods the CPA is also located on the edge of the safety circle around the intruder, however, the tangency assumption is removed. Instead, the ownship is assigned an arbitrary course angle χ_{cpa} when it is located on the edge of the safety circle as seen in Fig. 2.3. This illustration is represented in a right-handed X-Y-Z inertial reference frame, and the variable χ_{cpa} is defined relative to the X axis. In addition to χ_{cpa}, an additional angle is needed to define the location of the ownship on the edge of the safety circle when the CPA is reached which is shown by the variable θ_{cpa} in Fig. 2.3. The variable θ_{cpa} is measured relative to the negative X axis and represents the angle between the forward flight path of the intruder and the line connecting the CPA of the intruder and ownship. Using the definition of θ_{cpa}, a right triangle is formed with its hypotenuse equal to the safety radius R_s and the two sides equal to x_{cpa}, and y_{cpa}.

The CPA can be identified by taking the derivative of the range with respect to time, known as the range rate, and setting it to zero. The range rate is calculated as

$$\dot{r} = \frac{\mathbf{p}_o - \mathbf{p}_i}{\|\mathbf{p}_o - \mathbf{p}_i\|} \cdot (\mathbf{v}_o - \mathbf{v}_i), \tag{2.6}$$

where $\mathbf{p}_o$, $\mathbf{p}_i$, $\mathbf{v}_o$, and $\mathbf{v}_i$ are vectors defined in the inertial reference frame. The vector $\mathbf{p}_o$ is the position vector of the ownship, $\mathbf{p}_i$ is the position vector of the intruder, $\mathbf{v}_o$ is the velocity vector of the ownship, and $\mathbf{v}_i$ is the velocity vector of the intruder. Using Fig. 2.3, the position and velocity vectors can be defined in terms of the safety radius R_s, the ownship and intruder velocities v_o, and v_i, and the unknown variables x_{cpa}, y_{cpa}, and χ_{cpa}. Since the aircraft are assumed to be flying at constant altitude, the Z component of the position and velocity vectors will be neglected. The origin is defined as the intruder position when the CPA is reached so that $\mathbf{p}_i = (0,0)$. Using the intruder's position as the origin, the ownship's position is then defined as $\mathbf{p}_o = (-x_{\text{cpa}}, y_{\text{cpa}})$. The intruder's velocity vector is defined to be in the -X direction, therefore, the intruder's velocity vector is defined as $\mathbf{v}_i = (-v_i,0)$. The ownship's velocity vector points in the direction of χ_{cpa} and is defined by $\mathbf{v}_o = (v_o \cos\chi_{\text{cpa}}, v_o \sin\chi_{\text{cpa}})$. Substituting these position and velocity vectors into Eq. (2.6) and equating it to zero results in

$$\begin{aligned} 0 &= \frac{(-x_{\text{cpa}}, y_{\text{cpa}}) - (0,0)}{\|(-x_{\text{cpa}}, y_{\text{cpa}}) - (0,0)\|} \cdot ((v_o \cos\chi_{\text{cpa}}, v_o \sin\chi_{\text{cpa}}) - (-v_i, 0)), \\ &= \frac{1}{R_s} \left[v_o \sin\chi_{\text{cpa}} y_{\text{cpa}} - (v_i + v_o \cos\chi_{\text{cpa}}) x_{\text{cpa}} \right], \end{aligned} \tag{2.7}$$

where we now have a single equation with unknown variables x_{cpa}, y_{cpa}, and χ_{cpa}.

If an instantaneous bank-angle maneuver is assumed, as in the GVV method, the three unknown variables can be expressed in terms of the single unknown variable θ_{cpa}. The resulting equation can then be solved for θ_{cpa} explicitly, from which the specific values for x_{cpa}, y_{cpa}, and χ_{cpa} are found.

If on the other hand, a non-instantaneous bank-angle maneuver is assumed, as in the TGVV method, the variable χ_{cpa} cannot be expressed in terms of θ_{cpa}. For this case χ_{cpa} is found by integration of the turning dynamics of the aircraft over time. Since the variable

χ_{cpa} is now based on time, the three unknown variables are expressed in terms of the time to maneuver to the CPA, t_m, instead of the unknown variable θ_{cpa}. After substituting these expressions back into Eq. (2.7), the resulting equation cannot be solved for t_m explicitly and must instead be found through numerical methods. Once t_m has been determined, the specific values for x_{cpa}, y_{cpa}, and χ_{cpa} can be found.

Using Fig. 2.3 and the definitions of the CPA as described above, d_{CPA} is equal to x_{cpa}. From this figure we can also see that $d_{o,m}$ is equal to the X component of the maneuvering ownship when it reaches the CPA, x_m. The head-on distance traveled by the intruder while the ownship is executing its maneuver is a linear function of the time it takes the ownship to maneuver to the CPA as $d_{i,m} = v_i t_m$. Finally, since the ownship and intruder are assumed to be traveling at constant velocity, $d_{o,c}$ and $d_{i,c}$ are linear function of the computation time and are expressed as $d_{o,c} = v_o t_c$ and $d_{i,c} = v_i t_c$ respectively. With each of these definitions, the general equation for the minimum detection range finally becomes

$$d_{\text{MDR}} = (v_o + v_i)t_c + x_m + v_i t_m + x_{\text{cpa}}. \tag{2.8}$$

The remaining variables x_m, t_m, and x_{cpa} are dependent on the specific maneuver taken by the ownship and the location on the edge of the safety circle where the CPA occurs. The specific maneuver taken by the ownship and the resulting location of the CPA differ between the TGVV and GVV methods and will be defined in the following two sections along with the resulting minimum detection range.

Time-Based Geometric Velocity Vectors Approach

As stated above the TGVV approach assumes that the turning maneuver executed by the ownship is driven by a non-instantaneous bank angle change and the resulting trajectory must be characterized by numerical integration of time-based turn dynamics. The first step in characterizing the trajectory is to define the X and Y positions of the ownship during its maneuver as functions of time as

$$p_x(t) = \int_{t_0}^{t} v_x(\sigma)d\sigma = \int_{t_0}^{t} v_o \cos\chi(\sigma)d\sigma, \tag{2.9}$$

$$p_y(t) = \int_{t_0}^{t} v_y(\sigma)d\sigma = \int_{t_0}^{t} v_o \sin\chi(\sigma)d\sigma. \tag{2.10}$$

These positions are measured relative to the location where the ownship initiates its avoidance maneuver and the time is measured relative to the time when the ownship initiates its avoidance maneuver, t_0. The variables v_x and v_y are the X and Y velocity components of the ownship as functions of time while it performs its turning maneuver. The variable χ is the course of the ownship as a function of time while it performs its turning maneuver, and σ is the independent variable of integration.

Assuming the ownship performs a coordinated turn maneuver, the course of the aircraft can be found by integrating the course rate from Eq. (2.1) as

$$\chi(t) = \int_{t_0}^{t} \dot{\chi}(\sigma)d\sigma = \int_{t_0}^{t} \frac{g}{v_o}\tan\phi(\sigma)d\sigma. \tag{2.11}$$

For the GVV, method which will be described later, ϕ is assumed to be a step function with a magnitude of $\phi_{\max}$, however, the TGVV approach assumes that the turning maneuver executed by the ownship is influenced by the bank-angle dynamics. At this point any desired banking dynamics could be used to define the course as a function of time, however, we have chosen to use a first-order transfer function that describes the roll rate of the aircraft, p, in response to the deflection of the ailerons δ_a. Roll rate is integrated to get the bank angle. A block diagram of this system is shown in Fig. 2.4, where K and τ are general first-order system parameters, and s is the Laplace variable.

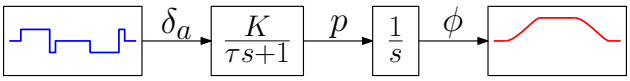
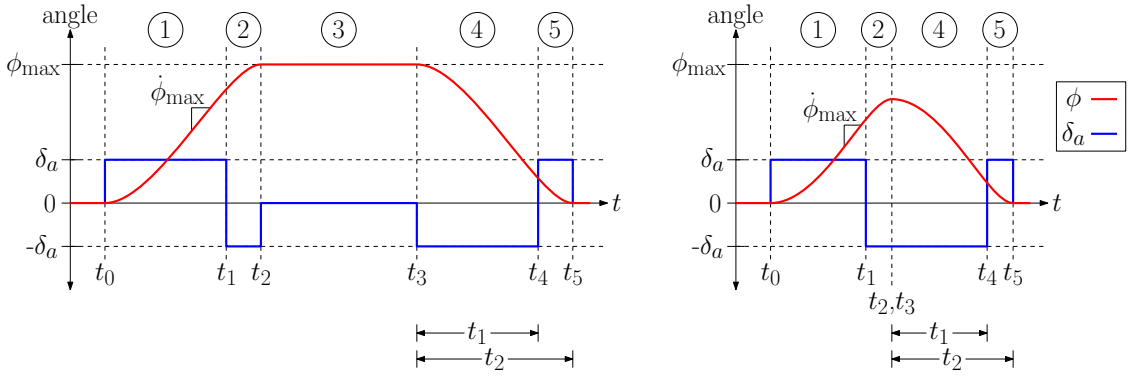


Figure 2.4: Block diagram of the bank-angle response due to aileron inputs.

Using this dynamic model we design a set of aileron commands to achieve a desired bank-angle response as shown in Fig. 2.5(a). In interval 1, a positive aileron step is used at t_0 to achieve a constant bank-angle rate, $\dot{\phi}_{\max}$. In interval 2, a negative aileron step is used at t_1 to stop the rolling motion at the maximum bank angle, $\phi_{\max}$. In interval 3, the aircraft

holds this constant bank angle, beginning at t_2, until it is time to return to level flight. The time at which the aircraft needs to begin returning to level flight is shown by t_3 and is chosen so that the total change in course resulting from the turning maneuver is equal to χ_t. In interval 4, a negative aileron step is used at t_3 to achieve a constant bank-angle rate, $-\dot{\phi}_{\max}$. In interval 5, a positive aileron step is used at t_4 to stop the rolling motion with the bank angle returns at zero. Finally, at t_5 the aircraft remains in level flight at the desired course, χ_t, for the remainder of the avoidance maneuver. In some cases the aircraft may not reach the maximum bank angle before it is time to return to level flight to ensure the proper χ_t is achieved. In these cases the bank angle has the response shown in Fig. 2.5(b) from which we notice t_2 and t_3 occur at the same time instant and there is no longer an interval 3. We also notice that intervals 2 and 4 are combined in terms of the negative step input to the system from the ailerons.



(a) Case A: $\phi_{\max}$ is reached.

(b) Case B: $\phi_{\max}$ is not reached.

Figure 2.5: Bank-angle response to aileron step inputs.

From Fig. 2.5, the segments of the bank-angle response where ϕ is changing as a result of the aileron input are intervals 1, 2, 4, and 5. The time response of ϕ to step inputs in the aileron command can be derived from the ordinary differential equation describing the bank-angle dynamics

$$\tau\ddot{\phi} + \dot{\phi} = K\delta_a(t).$$

The bank-angle response due to an aileron step input of magnitude δ_a occurring at time t_0 is given by

$$\phi(t) = K\delta_a \left[\tau e^{-(t-t_0)/\tau} - \tau + (t - t_0)\right] + \tau\dot{\phi}(t_0)\left(1 - e^{-(t-t_0)/\tau}\right) + \phi(t_0),$$

while the bank-rate response is given by

$$\dot{\phi}(t) = K\delta_a \left(1 - e^{-(t-t_0)/\tau}\right) + \dot{\phi}(t_0)e^{-(t-t_0)/\tau}.$$

The magnitude of the aileron step input is chosen so that the steady-state bank-angle rate is the prescribed maximum $\dot{\phi}_{\max}$, which gives

$$\delta_a = \frac{\dot{\phi}_{\max}}{K}. \tag{2.12}$$

Substituting this value of δ_a back into the equations for $\phi(t)$ and $\dot{\phi}(t)$ results in equations in terms of τ and $\dot{\phi}_{\max}$ as

$$\phi(t) = \tau\dot{\phi}_{\max}\left(e^{-(t-t_0)/\tau} - 1\right) + \dot{\phi}_{\max}(t - t_0) + \tau\dot{\phi}(t_0)\left(1 - e^{-(t-t_0)/\tau}\right) + \phi(t_0), \tag{2.13}$$

$$\dot{\phi}(t) = \dot{\phi}_{\max}\left(1 - e^{-(t-t_0)/\tau}\right) + \dot{\phi}(t_0)e^{-(t-t_0)/\tau}. \tag{2.14}$$

These equations are the general equations used to define the response of the bank angle to a positive or negative step input from the ailerons for Cases A and B. The specific bank-angle responses for each interval are derived for both cases in Appendix A.

Tables 2.1 and 2.2 provide a summary of the specific bank-angle response for each interval, shown by ϕ_i, for Case A and Case B, respectively. Additionally, the transition times for each segment are shown by t_i, and the change in course during each segment is shown by $\delta\chi_i$. To determine if Case A or Case B is required for the bank-angle dynamics of the ownship we use the value calculated for $\delta\chi_3$. If $\delta\chi_3 \geq 0$ then Case A is used, and if $\delta\chi_3 < 0$ then Case B is used.

Table 2.1: Bank angle, time intervals, and change in course for each segment of Case A.

Case A		
Interval	$\phi_i(t)$	
1	$\phi_1(t) = \tau\dot{\phi}_{\max}\left(e^{-t/\tau} - 1\right) + \dot{\phi}_{\max}t$	
2	$\phi_2(t) = \tau\dot{\phi}_{\max}\left(1 - 2e^{-(t-t_1)/\tau} + e^{-t/\tau}\right) - \dot{\phi}_{\max}(t - 2t_1)$	
3	$\phi_3(t) = \phi_{\max}$	
4	$\phi_4(t) = \tau\dot{\phi}_{\max}\left(1 - e^{-(t-t_3)/\tau}\right) - \dot{\phi}_{\max}(t - t_3) + \phi_{\max}$	
5	$\phi_5(t) = \tau\dot{\phi}_{\max}\left(2e^{-(t-t_4)/\tau} - 1 - e^{-(t-t_3)/\tau}\right) + \dot{\phi}_{\max}(t - 2t_4 + t_3) + \phi_{\max}$	
Interval	t_i	$\delta\chi_i$
1	$t_1 = -\tau \ln\left[\frac{e^{-(\phi_{\max}/\dot{\phi}_{\max})/\tau}}{1+\sqrt{1-e^{-(\phi_{\max}/\dot{\phi}_{\max})/\tau}}}\right]$	$\delta\chi_1 = \int_{t_0}^{t_1} \frac{g}{v_o}\tan(\phi_1(t))dt$
2	$t_2 = 2t_1 - \frac{\phi_{\max}}{\dot{\phi}_{\max}}$	$\delta\chi_2 = \int_{t_1}^{t_2} \frac{g}{v_o}\tan(\phi_2(t))dt$
3	$t_3 = t_2 + \frac{R_{\min}\delta\chi_3}{v_o}$	$\delta\chi_3 = \chi_t - (\delta\chi_1 + \delta\chi_2 + \delta\chi_4\vert_{t_3=0,t_4=t_1} + \delta\chi_5\vert_{t_3=0,t_4=t_1,t_5=t_2})$
4	$t_4 = t_3 + t_1$	$\delta\chi_4 = \int_{t_3}^{t_4} \frac{g}{v_o}\tan(\phi_4(t))dt$
5	$t_5 = t_3 + t_2$	$\delta\chi_5 = \int_{t_4}^{t_5} \frac{g}{v_o}\tan(\phi_5(t))dt$

Table 2.2: Bank angle, time intervals, and change in course for each segment of Case B.

Case B		
Interval	$\phi_i(t)$	
1	$\phi_1(t) = \tau\dot{\phi}_{\max}\left(e^{-t/\tau} - 1\right) + \dot{\phi}_{\max}t$	
2,4	$\phi_{2,4}(t) = \tau\dot{\phi}_{\max}\left(1 - 2e^{-(t-t_1)/\tau} + e^{-t/\tau}\right) - \dot{\phi}_{\max}(t - 2t_1)$	
5	$\phi_5(t) = \tau\dot{\phi}_{\max}\left(e^{-t/\tau}(2e^{t_1/\tau} - 1)^2 - 1 - 2\ln\left[2e^{t_1/\tau} - 1\right]\right) + \dot{\phi}_{\max}t$	
Interval	t_i	$\delta\chi_i$
1	numerical method in Appendix A.2	$\delta\chi_1 = \int_{t_0}^{t_1} \frac{g}{v_o}\tan(\phi_1(t))dt$
2,4	$t_2 = \tau\ln\left[2e^{t_1/\tau} - 1\right]$, $t_4 = t_2 + t_1$	$\delta\chi_{2,4} = \int_{t_1}^{t_4} \frac{g}{v_o}\tan(\phi_{2,4}(t))dt$
5	$t_5 = 2t_2$	$\delta\chi_5 = \int_{t_4}^{t_5} \frac{g}{v_o}\tan(\phi_5(t))dt$

Using the information from Table 2.1 we create an expression for the course of the ownship after it initiates the avoidance maneuver for Case A as

$$\chi(t) = \begin{cases} 0 & \text{if } t \leq t_0, \\ \int_{t_0}^{t} \frac{g}{v_o} \tan(\phi_1(\sigma)) d\sigma & \text{if } t_0 < t \leq t_1, \\ \delta\chi_1 + \int_{t_1}^{t} \frac{g}{v_o} \tan(\phi_2(\sigma)) d\sigma & \text{if } t_1 < t \leq t_2, \\ \delta\chi_1 + \delta\chi_2 + \int_{t_2}^{t} \frac{g}{v_o} \tan(\phi_3(\sigma)) d\sigma & \text{if } t_2 < t \leq t_3, \\ \delta\chi_1 + \delta\chi_2 + \delta\chi_3 + \int_{t_3}^{t} \frac{g}{v_o} \tan(\phi_4(\sigma)) d\sigma & \text{if } t_3 < t \leq t_4, \\ \delta\chi_1 + \delta\chi_2 + \delta\chi_3 + \delta\chi_4 + \int_{t_4}^{t} \frac{g}{v_o} \tan(\phi_5(\sigma)) d\sigma & \text{if } t_4 < t \leq t_5, \\ \chi_t & \text{if } t_5 < t, \end{cases} \tag{2.15}$$

Similarly, using the information from Table 2.2 we create an expression for the course of the ownship after it initiates the avoidance maneuver for Case B as

$$\chi(t) = \begin{cases} 0, & \text{if } t \leq t_0 \\ \int_{t_0}^{t} \frac{g}{v_o} \tan(\phi_1(\sigma)) d\sigma, & \text{if } t_0 < t \leq t_1 \\ \delta\chi_1 + \int_{t_1}^{t} \frac{g}{v_o} \tan(\phi_{2,4}(\sigma)) d\sigma, & \text{if } t_1 < t \leq t_4 \\ \delta\chi_1 + \delta\chi_{2,4} + \int_{t_4}^{t} \frac{g}{v_o} \tan(\phi_5(\sigma)) d\sigma, & \text{if } t_4 < t \leq t_5 \\ \chi_t, & \text{if } t_5 < t \end{cases} \tag{2.16}$$

Using Eqs. (2.15) and (2.16) for the course of the ownship and Eqs. (2.9) and (2.10) for the position of the ownship, the trajectory of the ownship is fully defined. Using this trajectory, we return to the analysis of the CPA location. The time to maneuver is the time it takes the ownship to initiate a turning maneuver until it reaches the CPA location on the edge of the safety circle around the intruder. This segment of the flight path of the ownship is shown in Fig. 2.3 and is composed of X and Y components, x_m and y_m, respectively. These X and Y components can be defined in terms of the equations for $p_x(t)$ and $p_y(t)$,

Eqs. (2.9) and (2.10), and the time to maneuver as

$$x_m = p_x(t_m), \tag{2.17}$$

$$y_m = p_y(t_m), \tag{2.18}$$

From Eq. (2.7) we have three unknown variables x_{cpa}, y_{cpa}, and χ_{cpa}. Using Fig. 2.3 we see that the unknown variable y_{cpa}, which defines the Y component of the CPA, has the same value as the Y component of the maneuvering ownship y_m at the CPA. Accordingly, x_{cpa} and y_{cpa} can be expressed as

$$y_{\text{cpa}} = y_m = p_y(t_m), \tag{2.19}$$

$$x_{\text{cpa}} = \sqrt{R_s^2 - y_{\text{cpa}}^2} = \sqrt{R_s^2 - p_y(t_m)^2}, \tag{2.20}$$

Similarly χ_{cpa} can be expressed as

$$\chi_{\text{cpa}} = \chi(t_m). \tag{2.21}$$

After substituting these expressions for x_{cpa}, y_{cpa}, and χ_{cpa} into Eq. (2.7) and simplifying we get

$$0 = p_y(t_m)v_o \sin\chi(t_m) - \sqrt{R_s^2 - p_y(t_m)^2}(v_i + v_o \cos\chi(t_m)). \tag{2.22}$$

This equation is now a function of a single variable t_m, which can be solved for using the Newton-Raphson method. Once the value of t_m has been found the three variables x_{cpa}, y_{cpa}, and χ_{cpa} can then be calculated, along with the variables x_m and y_m.

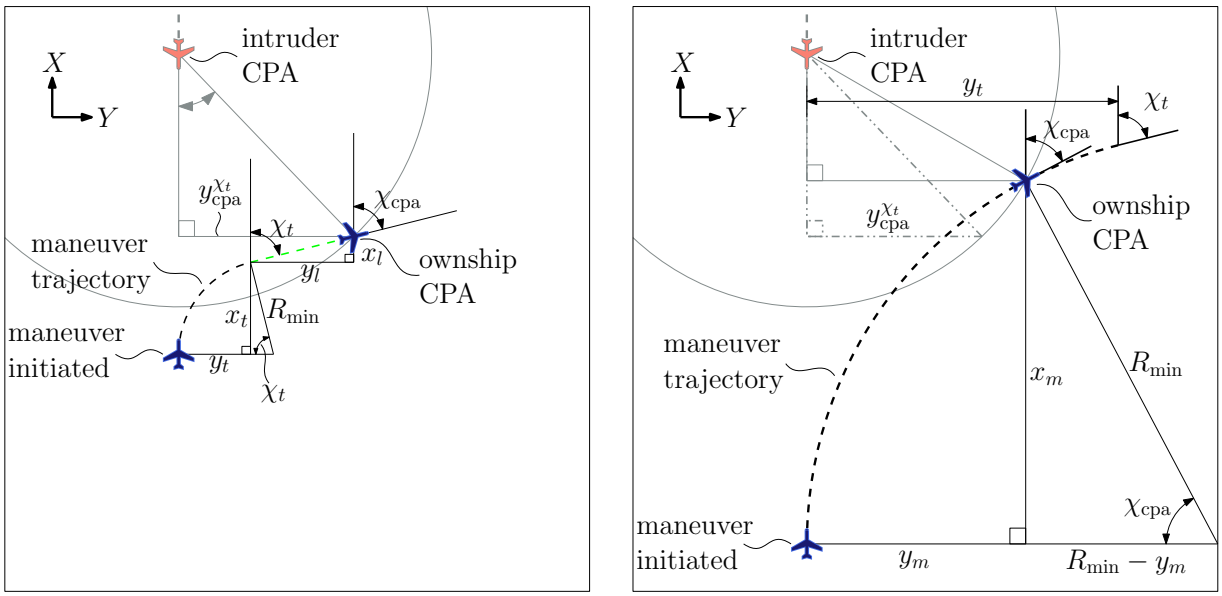
Now that the CPA location has been defined we return to the calculation of the minimum detection range shown in Eq. (2.8). With the values for x_m, t_m, and x_{cpa} just derived, the minimum detection range can be defined as

$$d_{\text{MDR}} = (v_o + v_i)t_c + p_x(t_m) + v_i t_m + \sqrt{R_s^2 - p_y(t_m)^2}, \tag{2.23}$$

where t_m is found from Eq. (2.22) using the Newton-Raphson method, $p_x(t_m)$ and $p_y(t_m)$ are found from Eqs. (2.9) and (2.10), and $\chi(t_m)$ is generally defined in Eq. (2.11), but is specifically defined for both Case A and Case B in Eqs. (2.15) and (2.16), respectively.

Geometric Velocity Vectors Approach

The TGVV method described above incorporates the bank-angle dynamics of the ownship into the avoidance path. If the bank-angle dynamics of the ownship are fast relative to the maneuver time, then the assumption of an instantaneous bank angle becomes more realistic. For the GVV method we make this assumption, and in the results section we show under what conditions this assumption is valid by comparing it to the TGVV method.



(a) Case 1: $y_t \leq y_{\text{cpa}}^{\chi_t}$. Ownship completes turning maneuver before reaching the CPA location.

(b) Case 2: $y_t > y_{\text{cpa}}^{\chi_t}$. Ownship reaches CPA location before completing turning maneuver.

Figure 2.6: Geometric diagram for the CPA location resulting from circular turning trajectories of the ownship.

Instead of solving the bank-angle dynamics we begin by defining the geometry that can be used to calculate the minimum detection range as shown in Fig. 2.6. From this figure we see that there are two different geometrical cases that can occur while the ownship is maneuvering. For the first case, shown in Fig. 2.6(a), the ownship turns with a small turning radius compared to the safety radius and is able to complete its turning maneuver before reaching the CPA location. This means that after the ownship completes its turning maneuver it will fly straight until the CPA location is reached. The X and Y components

of the turning segment of the maneuver are represented by the variables x_t and y_t, and the X and Y components of the linear segment of the maneuver are represented by x_l and y_l. In the second case, shown in Fig. 2.6(b), the ownship turns gradually and reaches the CPA location before completing its turning maneuver. This means that the ownship will still be in a banked turn as it passes the CPA location. The X and Y components of the turning segment of the maneuver for this case are similarly represented by x_t and y_t.

These cases can be distinguished mathematically in the following manner. First we assume the ownship executes a turning maneuver until its course is equal to the prescribed value for χ_t. The Y component of this turning maneuver for both cases is calculated from y_t as

$$y_t = R_{\min}(1 - \cos\chi_t). \tag{2.24}$$

Second we calculate y_{cpa} with the course set equal to the desired turn angle χ_t and define this as $y_{\text{cpa}}^{\chi_t}$. The exact derivation of this parameter will be shown in the subsection for Case 1. This Y coordinate denotes the transition between Case 1 and Case 2 where the turn maneuver is completed at the exact moment the CPA is reached. The final step is to compare the calculated y_t and $y_{\text{cpa}}^{\chi_t}$, which allows us to determine which geometry should be used to calculate the minimum detection range. For Case 1, $y_t \leq y_{\text{cpa}}^{\chi_t}$, while for Case 2 $y_t > y_{\text{cpa}}^{\chi_t}$.

As stated previously, for the GVV method Eq. (2.7) can be expressed in terms of the variable θ_{cpa}. Two of the variables x_{cpa} and y_{cpa} can be immediately expressed in terms of θ_{cpa} as

$$x_{\text{cpa}} = R_s \cos\theta_{\text{cpa}}, \tag{2.25}$$

$$y_{\text{cpa}} = R_s \sin\theta_{\text{cpa}}. \tag{2.26}$$

Substituting these into Eq. (2.7) and simplifying results in

$$0 = v_o \sin\chi_{\text{cpa}} \sin\theta_{\text{cpa}} - (v_i + v_o \cos\chi_{\text{cpa}}) \cos\theta_{\text{cpa}}. \tag{2.27}$$

The variable χ_{cpa} is different for Cases 1 and 2 and will be derived in the following two subsections along with the resulting minimum detection range.

Case 1: $y_t \leq y_{\text{cpa}}^{\chi_t}$

In Case 1, the ownship completes its turning maneuver before reaching the CPA, therefore, the course of the ownship when it reaches the CPA, χ_{cpa}, will be equal to the prescribed value χ_t. Substituting this into Eq. (2.27) results in

$$0 = v_o \sin\chi_t \sin\theta_{\text{cpa}} - (v_i + v_o \cos\chi_t)\cos\theta_{\text{cpa}}. \tag{2.28}$$

This equation is then used to solve for θ_{cpa} as

$$\theta_{\text{cpa}} = \tan^{-1}\left(\frac{v_i + v_o\cos\chi_t}{v_o \sin\chi_t}\right),$$

where θ_{cpa} is now used in the expressions for x_{cpa}, and y_{cpa} to produce

$$x_{\text{cpa}} = R_s \frac{v_o \sin\chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_o v_i \cos\chi_t}}, \tag{2.29}$$

$$y_{\text{cpa}} = R_s \frac{v_i + v_o\cos\chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_o v_i \cos\chi_t}}. \tag{2.30}$$

Since the ownship reaches a course of χ_t before reaching the CPA in Case 1, then the variable $y_{\text{cpa}}^{\chi_t}$ is equivalent to y_{cpa} defined by Eq. (2.30)

$$y_{\text{cpa}}^{\chi_t} = R_s \frac{v_i + v_o\cos\chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_o v_i \cos\chi_t}}.$$

This value of $y_{\text{cpa}}^{\chi_t}$ is used to determine whether a specific encounter scenario is of Case 1 or Case 2 geometry.

Next we define the variables x_t, y_t, x_l, y_l from the geometry of Fig. 2.6(a) as

$$x_t = R_{\min} \sin \chi_t, \tag{2.31}$$

$$y_t = R_{\min}(1 - \cos \chi_t), \tag{2.32}$$

$$y_l = y_{\text{cpa}} - y_t = R_s \frac{v_i + v_o \cos \chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_o v_i \cos \chi_t}} - R_{\min}(1 - \cos \chi_t), \tag{2.33}$$

$$x_l = y_l \cot \chi_t = \left[R_s \frac{v_i + v_o \cos \chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_o v_i \cos \chi_t}} - R_{\min}(1 - \cos \chi_t) \right] \cot \chi_t, \tag{2.34}$$

Using these definitions we can define the remaining variables needed for the minimum detection range x_m and t_m as

$$\begin{aligned} x_m &= x_t + x_l, \\ &= R_{\min} \sin \chi_t + \left[R_s \frac{v_i + v_o \cos \chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_o v_i \cos \chi_t}} - R_{\min}(1 - \cos \chi_t) \right] \cot \chi_t, \end{aligned} \tag{2.35}$$

$$\begin{aligned} t_m = \frac{L}{v_o} &= \frac{R_{\min}\chi_t + \sqrt{x_l^2 + y_l^2}}{v_o} \\ &= \frac{R_{\min}\chi_t + \left[R_s \frac{v_i + v_o \cos \chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_o v_i \cos \chi_t}} - R_{\min}(1 - \cos \chi_t) \right] \sqrt{1 + \cot^2 \chi_t}}{v_o}, \end{aligned} \tag{2.36}$$

where L is the length of the avoidance path of the ownship during the turning and straight segments of the maneuver.

Substituting Eqs. (2.29), (2.36), and (2.35) into Eq. (2.8), and using the relationship defined in Eq. (2.2) for $R_{\min}$, produces the final minimum detection range equation for Case 1

as

$$d_{\text{MDR}} = (v_o + v_i)t_c + R_{\min}\sin\chi_t + \left[R_s\frac{v_i + v_o\cos\chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_ov_i\cos\chi_t}} - R_{\min}(1-\cos\chi_t)\right]\cot\chi_t$$
$$+ \frac{v_i}{v_o}\left[R_{\min}\chi_t + \left[R_s\frac{v_i + v_o\cos\chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_ov_i\cos\chi_t}} - R_{\min}(1-\cos\chi_t)\right]\sqrt{1+\cot^2\chi_t}\right]$$
$$+ R_s\frac{v_o\sin\chi_t}{\sqrt{v_o^2 + v_i^2 + 2v_ov_i\cos\chi_t}}. \tag{2.37}$$

Case 2: $y_t > y_{\text{cpa}}^{\chi_t}$

For this case we begin with Eq. (2.27), however, we must define expressions for $\cos\chi_{\text{cpa}}$ and $\sin\chi_{\text{cpa}}$ as functions of θ_{cpa}. We first define expressions for y_m and x_m as

$$y_m = y_{\text{cpa}} = R_s\sin\theta_{\text{cpa}}, \tag{2.38}$$
$$x_m = \sqrt{R_{\min}^2 - (R_{\min} - y_m)^2} = \sqrt{R_s\sin\theta_{\text{cpa}}(2R_{\min} - R_s\sin\theta_{\text{cpa}})}. \tag{2.39}$$

These values for y_m and x_m can now be used to define $\cos\chi_{\text{cpa}}$ and $\sin\chi_{\text{cpa}}$ as

$$\cos(\chi_{\text{cpa}}) = \frac{R_{\min} - y_m}{R_{\min}} = \frac{R_{\min} - R_s\sin\theta_{\text{cpa}}}{R_{\min}}, \tag{2.40}$$
$$\sin(\chi_{\text{cpa}}) = \frac{x_m}{R_{\min}} = \frac{\sqrt{R_s\sin\theta_{\text{cpa}}(2R_{\min} - R_s\sin\theta_{\text{cpa}})}}{R_{\min}}. \tag{2.41}$$

Similarly an expression for χ_{cpa} can be defined as

$$\chi_{\text{cpa}} = \tan^{-1}\left(\frac{x_m}{R_{\min} - y_m}\right) = \tan^{-1}\left(\frac{\sqrt{R_s\sin\theta_{\text{cpa}}(2R_{\min} - R_s\sin\theta_{\text{cpa}})}}{R_{\min} - R_s\sin\theta_{\text{cpa}}}\right). \tag{2.42}$$

Substituting these expressions from Eqs. (2.40) and (2.41) into Eq. (2.27) and manipulating produces an equation in terms of a single parameter θ_{cpa} in the following form

$$0 = a\sin^3\theta_{\text{cpa}} + b\sin^2\theta_{\text{cpa}} + c\sin\theta_{\text{cpa}} + d, \tag{2.43}$$

where a, b, c, and d are defined as

$$\begin{aligned} a =& 2v_i v_o R_{\min} R_s, \\ b =& v_o^2 R_s^2 - (v_i + v_o)^2 R_{\min}^2, \\ c =& -2v_o(v_i + v_o) R_{\min} R_s, \\ d =& (v_i + v_o)^2 R_{\min}^2. \end{aligned}$$

Equation (2.43) has been formulated to be cubic in $\sin\theta_{\text{cpa}}$. Applying Cardano's formulas [21] and expressing $\sin\theta_{\text{cpa}}$ as z produces three roots for $\sin\theta_{\text{cpa}}$ as

$$\begin{aligned} z_1 =& -\frac{b}{3a} + (S+T), \\ z_2 =& -\frac{b}{3a} - \frac{1}{2}(S+T) + \frac{1}{2} i\sqrt{3}(S-T), \\ z_3 =& -\frac{b}{3a} - \frac{1}{2}(S+T) - \frac{1}{2} i\sqrt{3}(S-T), \end{aligned}$$

where S and T are defined as $S = \sqrt[3]{R + \sqrt{D}}$, $T = \sqrt[3]{R - \sqrt{D}}$ respectively, D is defined as $D = Q^3 + R^2$, and Q and R are defined as $Q = \frac{c}{3a} - \left(\frac{b}{3a}\right)^2$, $R = \frac{bc}{6a^2} - \frac{d}{2a} - \left(\frac{b}{3a}\right)^3$ respectively. Since we are trying to find a root for the expression $\sin\theta_{\text{cpa}}$, the root must first lie within the bounds $[-1, 1]$. Second, the solution for θ_{cpa} must lie within the bounds of $[0, 90]$ degrees due to the head-on approach of the ownship and intruder and the right turning maneuver of the ownship as seen in Fig. 2.6, which means the root must further be restricted to $[0, 1]$. Finally, there may still exist multiple roots within the bounds $[0, 1]$, therefore, a third constraint must be satisfied. The chosen root must produce a value of θ_{cpa} that when used to calculate χ_{cpa} in Eq. (2.42) produces a value within the bounds of $[0, 90]$ degrees. Once these constraints are satisfied, the true root is identified and used to find θ_{cpa} as

$$\theta_{\text{cpa}} = \sin^{-1} z. \tag{2.44}$$

This value of θ_{cpa} is now used to define values for x_{cpa} and y_{cpa} shown in Eqs. (2.25) and (2.26), and y_m and x_m shown in Eqs. (2.38) and (2.39). The time to maneuver is found

as

$$t_m = \frac{L}{v_o} = \frac{R_{\min}\chi_{\text{cpa}}}{v_o} = \frac{1}{v_o} R_{\min} \tan^{-1}\left(\frac{\sqrt{R_s \sin\theta_{\text{cpa}}(2R_{\min} - R_s \sin\theta_{\text{cpa}})}}{R_{\min} - R_s \sin\theta_{\text{cpa}}}\right). \tag{2.45}$$

Substituting Eqs. (2.25), (2.39), and (2.45) into Eq. (2.8) produces the final equation for the Case 2 minimum detection range as

$$\begin{aligned} d_{\text{MDR}} = {} & (v_o + v_i)t_c + \sqrt{R_s \sin\theta_{\text{cpa}}(2R_{\min} - R_s \sin\theta_{\text{cpa}})} \\ & + \frac{v_i}{v_o} R_{\min} \tan^{-1}\left(\frac{\sqrt{R_s \sin\theta_{\text{cpa}}(2R_{\min} - R_s \sin\theta_{\text{cpa}})}}{R_{\min} - R_s \sin\theta_{\text{cpa}}}\right) + R_s \cos\theta_{\text{cpa}}, \end{aligned} \tag{2.46}$$

where θ_{cpa} is defined in Eq. (2.44) and $R_{\min}$ is defined in Eq. (2.2).

2.2 Results: Method Comparison and Validation

With the equations for the minimum detection range derived, we now present results showing calculated values for the minimum detection range as a function of each of the parameters used in the equations. We present these results for the two methods developed in this paper, the TGVV and GVV methods. We also present results for the two prior methods, TT and GT, and provide a detailed comparison of all four methods. The parameters that are used by all four methods include v_o, v_i, R_s, $\phi_{\max}$, and t_c. For the TGVV and GVV methods we have the additional parameter χ_t, and for the TGVV method we have two parameters used to describe the bank-angle dynamics, $\dot{\phi}_{\max}$ and τ. In creating the results, a nominal set of values are chosen for each of these parameters except the ownship speed and are listed in Table 2.3.

Table 2.3: Nominal parameter values used in the calculation of d_{MDR}.

Parameter	v_i	R_s	$\phi_{\max}$	t_c	χ_t	$\dot{\phi}_{\max}$	τ
Value	150 kts	500 ft	30 deg	5.0 s	90 deg	30 deg	0.5 s

The minimum detection range is calculated for all four methods using the parameter values from Table 2.3 and is shown in Fig. 2.7. In this figure d_{MDR} is plotted versus the ownship speed, with level curves used to vary one or two additional parameters.

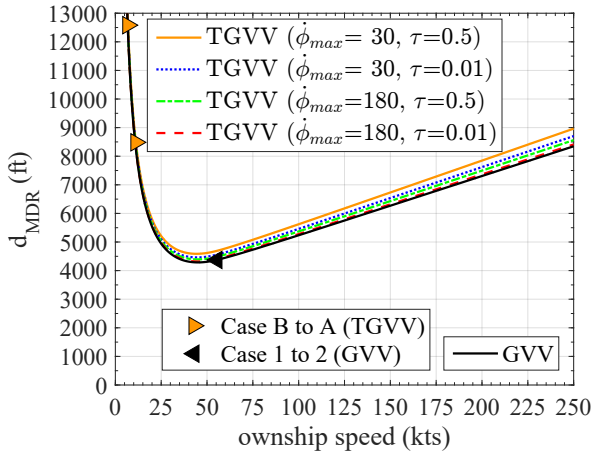
In Fig. 2.7(a) we first compare various values for $\dot{\phi}_{\max}$ and τ, which only affect the calculation of the TGVV method. The GVV method is also plotted to demonstrate what values of $\dot{\phi}_{\max}$ and τ result in the two methods producing similar values.

For Figs. 2.7(b) through 2.7(f) a single pair of values are used for $\dot{\phi}_{\max}$ and τ, which results in a single set of level curves for the TGVV method. Upon careful inspection, it can be seen that each subplot contains one level that corresponds to the core set of parameters in Table 2.3, which results in one common level among each of the subplots. This allows us to see how the minimum detection range deviates from a common level as each parameter is changed. Figure 2.7(b) is used to vary v_i, Fig. 2.7(c) varies R_s, Fig. 2.7(d) varies $\phi_{\max}$, Fig. 2.7(e) varies t_c, and Fig. 2.7(f) varies χ_t.

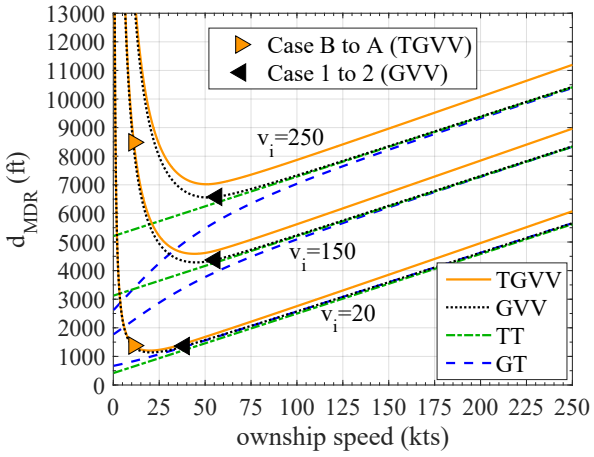
In Fig. 2.7 the right-pointing orange triangles are used to identify transition points from Case B to Case A for the TGVV method when viewed from left to right. Similarly, left-pointing black triangles are used to identify transition points from Case 1 to Case 2 for the GVV method when viewed from left to right.

From Fig. 2.7(a) a general trend between the TGVV and GVV methods can be seen. We see that as $\dot{\phi}_{\max}$ increases and τ decreases, the TGVV method d_{MDR} values approach the same values as those from the GVV method, shown by the black line. It can be seen that the TGVV method always predicts a minimum detection range slightly larger than the GVV method, which results from including the bank-angle dynamics in the turning maneuver.

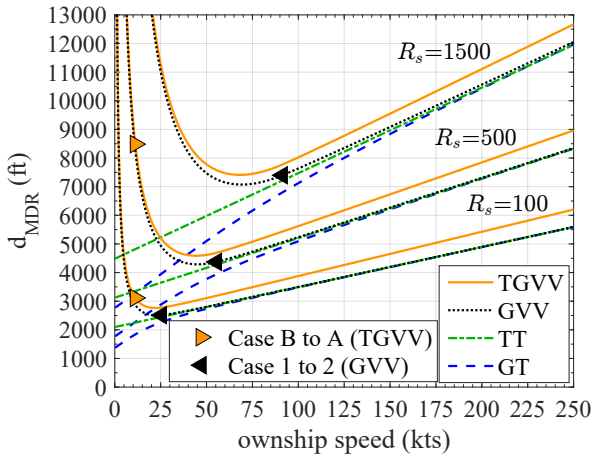
It is evident from Figs. 2.7(b) through 2.7(f) that the TT and GT methods approach the GVV method for ownship speeds in the Case 2 region of the GVV method. This is because Case 2 of the GVV method uses an avoidance maneuver that is performed solely by turning, which is an assumption made in both the TT and GT methods. For ownship speeds in the Case 1 region of the GVV method, the TT and GT methods predict values for the minimum detection range significantly smaller than the GVV and TGVV methods.



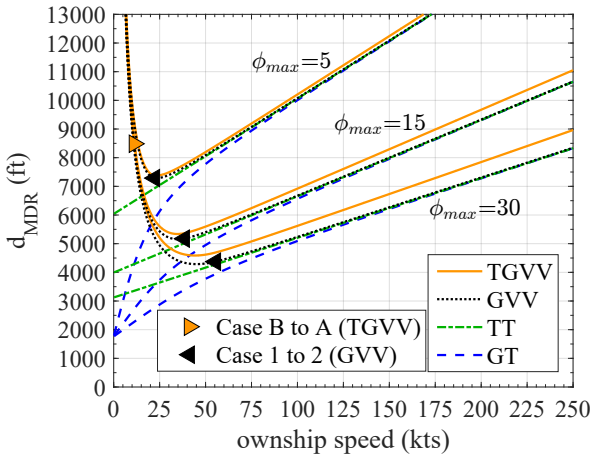
(a) Comparison as a function of $\dot{\phi}_{\max}$ and τ.



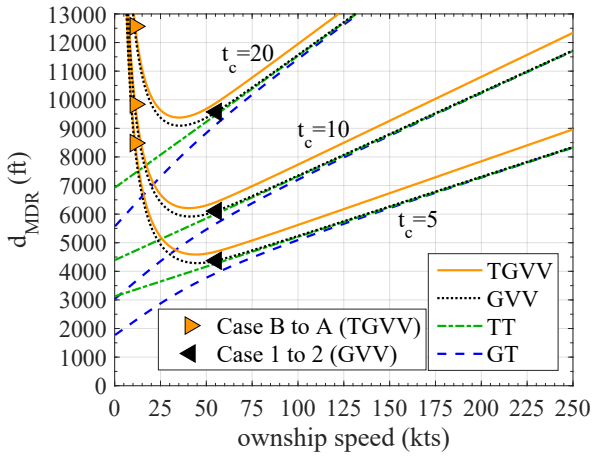
(b) Comparison as a function of v_i.



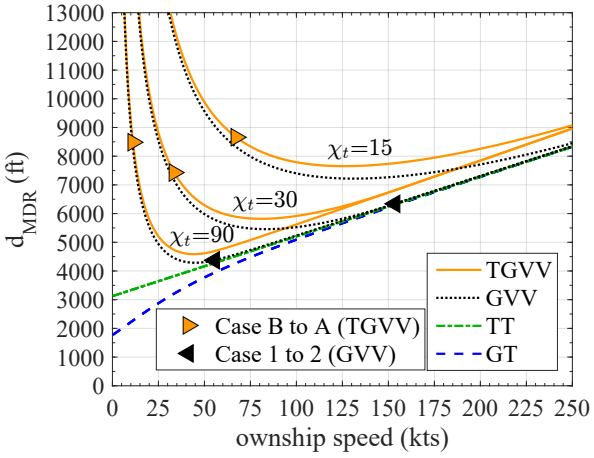
(c) Comparison as a function of R_s.



(d) Comparison as a function of $\phi_{\max}$.



(e) Comparison as a function of t_c.



(f) Comparison as a function of χ_t.

Figure 2.7: Comparison of d_{MDR} as a function of v_o, v_i, R_s, $\phi_{\max}$, t_c, χ_t, $\dot{\phi}_{\max}$ and τ.

Finally, from each subplot of Fig. 2.7 it can be seen that as the ownship speed increases, each of the methods models d_{MDR} as a linear function of the ownship speed. For large values of v_o, the nonlinear geometric methods (GVV and GT), converge to the linear model predicted by the TT method. Using the equation for the minimum detection range from the TT method, the slope of these lines can be inferred from of Eq. 2.4 to be a function of R_s, $\phi_{\max}$, and t_c. As R_s and t_c increase, the slopes of the resulting lines increase, as can be seen in Figs. 2.7(c) and 2.7(e); however, as $\phi_{\max}$ increases, the slope of the resulting lines decrease, as can be seen in Fig. 2.7(d). As the remaining variables, v_i and χ_t, are increased, the slopes resulting from the geometric methods stay constant as shown in Figs. 2.7(b) and 2.7(f).

Having calculated the minimum detection range, we must now determine the accuracy of the results. If the true minimum detection range is used to initialize the distance between two aircraft, the resulting CPA between the two aircraft will be exactly equal to the safety radius, R_s. This means we can check the accuracy of the minimum detection range calculated from each of the four methods by comparing the resulting CPA to the safety radius. To find the resulting CPA for each method, a simulation is performed. The ownship and intruder are initialized in a head-on configuration at a distance equal to d_{MDR}. Both aircraft fly towards each other without maneuvering during the computation time, after which the ownship begins turning using the bank-angle dynamics described by the TGVV method with the parameters $\dot{\phi}_{\max}$, τ, $\phi_{\max}$, and χ_t. Once the ownship has turned to the predefined χ_t, it flies straight until it is far from the intruder. During the simulation the relative range and CPA between the ownship and the intruder are calculated.

An example of the relative range is shown in Fig. 2.8. The parameters that are used come from Table 2.3, and the value of the ownship speed is $v_o = 25$ kts. The TGVV method is shown by the solid orange line, and at time zero the relative range between the two aircraft is approximately 5209 feet. As the two aircraft continue their flight paths, the closest point of approach is exactly equal to 500 feet at approximately 18.9 seconds. Since the TGVV method uses the same dynamic model as the ownship in the CPA simulation, the predicted relative range from the TGVV method is identical to the simulation truth model. The GVV method is shown by the dotted black line and the relative range predicted at time zero is approximately 4942 feet. In this case the two aircraft reach a CPA of 456 feet at

approximately 17.9 seconds, which is a violation of the safety volume. Similarly the ownship flight paths from the TT and GT methods result in a penetration of the safety volume with a CPA of 243 and 116 feet at approximately 12.9 and 9.9 seconds respectively.

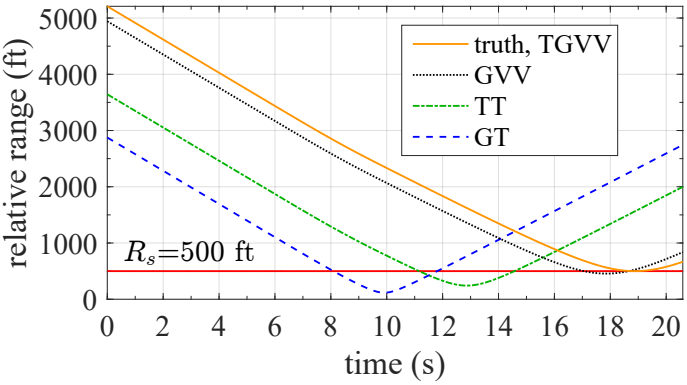
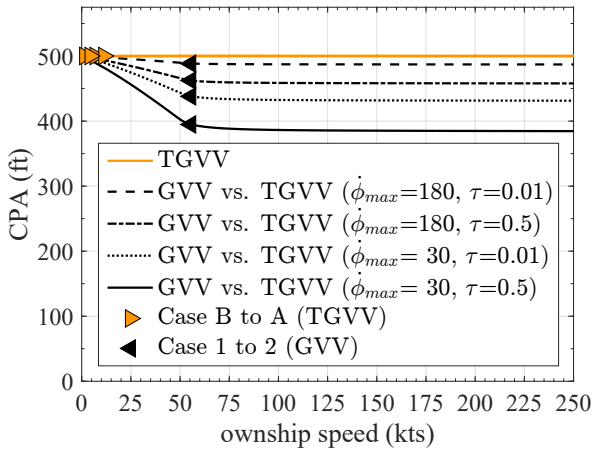
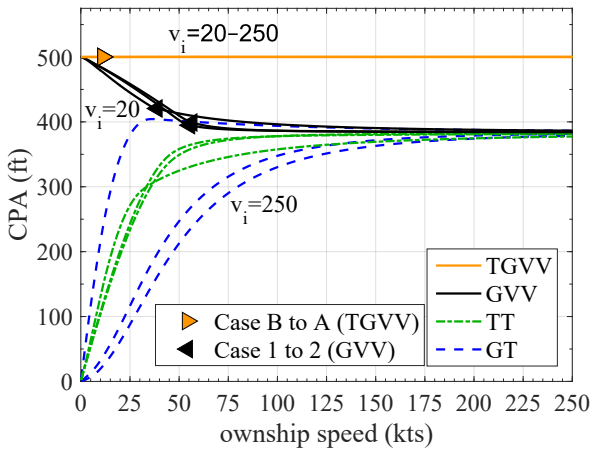


Figure 2.8: Relative range vs. time and the resulting CPA.

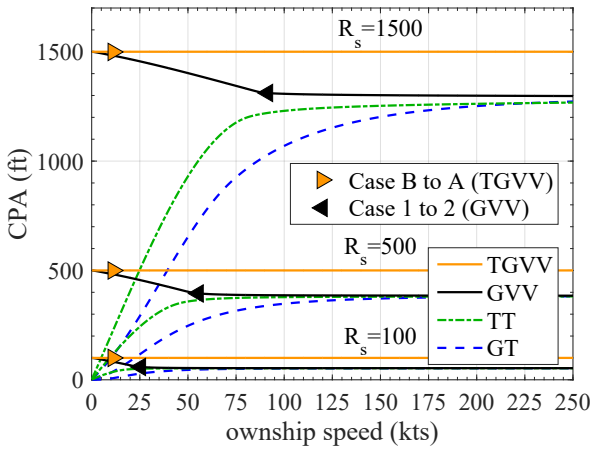
The CPA is used to compare the accuracy of each of the four methods. In Fig. 2.9 the CPA is plotted versus the speed of the ownship. In Fig. 2.9(a), four different sets of parameters are used for the bank-angle dynamics of the ownship, whereas, in the other subplots only the nominal set of parameters is used for the bank-angle dynamics of the ownship, while other critical parameters are varied. The first observation to make from each of the subplots is that the TGVV method always produces a CPA exactly equal to the value chosen for R_s which means that the safety volume has not been penetrated and the true minimum detection range has been found. Additionally, the GVV, TT, and GT methods all produce a CPA less than the chosen value for R_s which means the safety volume has been penetrated and the calculated minimum detection range is an under-approximation. Fig. 2.9(a), however, shows that the CPA of the GVV method approaches the safety radius of 500 ft as $\dot{\phi}_{\max}$ increases, and τ decreases, implying that the GVV method becomes a good approximation of the TGVV method as the speed of the bank-angle response increases.



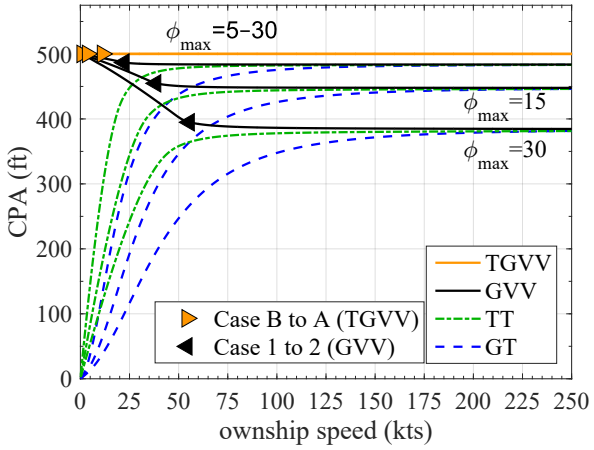
(a) Comparison as a function of $\dot{\phi}_{\max}$ and τ.



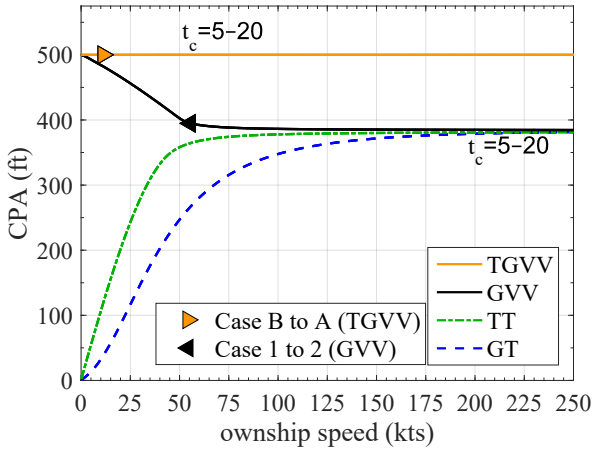
(b) Comparison as a function of v_i.



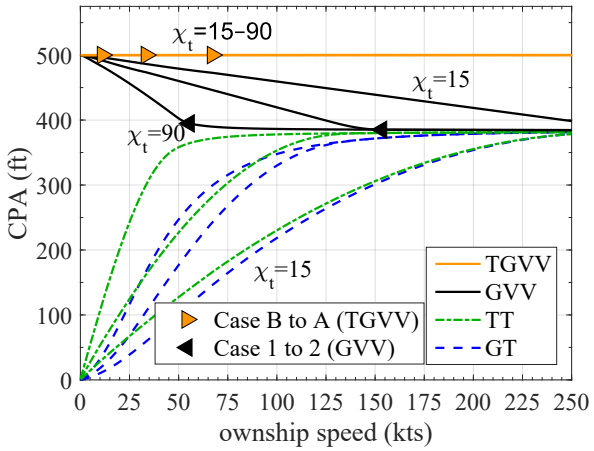
(c) Comparison as a function of R_s.



(d) Comparison as a function of $\phi_{\max}$.



(e) Comparison as a function of t_c.



(f) Comparison as a function of χ_t.

Figure 2.9: Comparison of the CPA as a function of v_o, v_i, R_s, $\phi_{\max}$, t_c, χ_t, $\dot{\phi}_{\max}$ and τ.

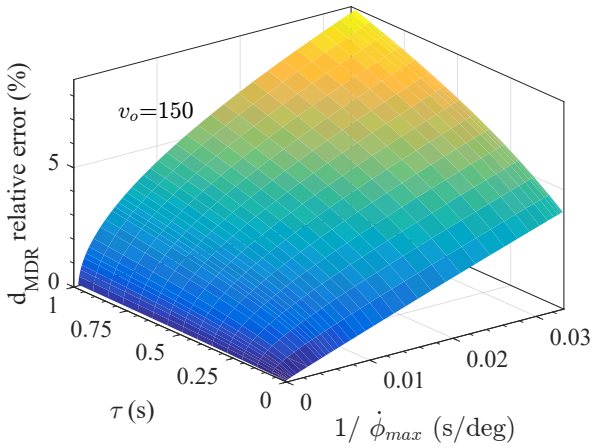
The next critical observation to make is seen in Fig. 2.9(b), which shows that the CPA of the TT and GT methods converge to the CPA of the GVV method as v_o increases. Additionally, it can be seen the GT method approaches the GVV method as v_i decreases. This is because the underlying assumption of tangent turning circles is more reasonable when the intruder speed is small relative to the ownship speed.

Figure 2.9(d) shows that the CPA of the GVV, TT, and GT methods approach the desired value chosen for R_s as $\phi_{\max}$ decreases. This is because smaller values of $\phi_{\max}$ result in shorter bank-angle transients and produce more circular turns that more closely match the circular-turn assumptions of the GVV, TT, and GT methods.

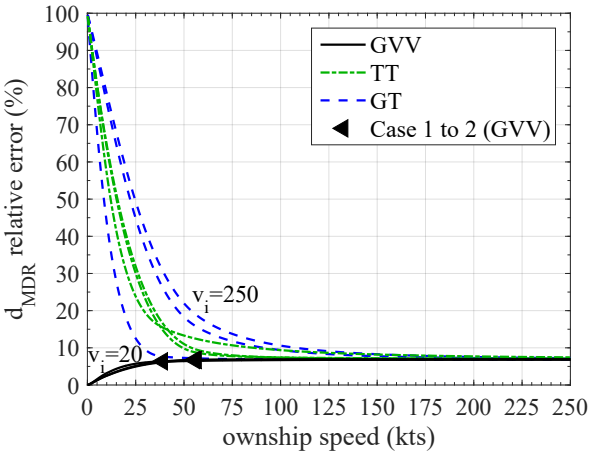
Finally, from Fig. 2.9(f) the TT and GT methods depart from the GVV method as χ_t decreases at low ownship speeds. This is because the TT and GT methods do not provide any compensation for limiting the turn angle, χ_t. Instead, these methods assume the ownship is always in a banked turning maneuver as it passes the intruder. This assumption matches the avoidance trajectory of the ownship for Case 2 of the GVV method and is a valid assumption for large ownship speeds. This assumption causes issues at low ownship speeds because the ownship completes its turning maneuver before reaching the CPA which corresponds to Case 1 of the GVV method.

The CPA plots show that the TGVV method always results in a CPA equal to the safety radius, which means the true minimum detection range has been found. The geometric methods (GVV, TT, and GT), however, are computationally simpler and it is of interest to know how well they approximate the solution produced by the TGVV method. Figure 2.10 provides this information by showing the percent relative error in the minimum detection range between the other methods and the TGVV method.

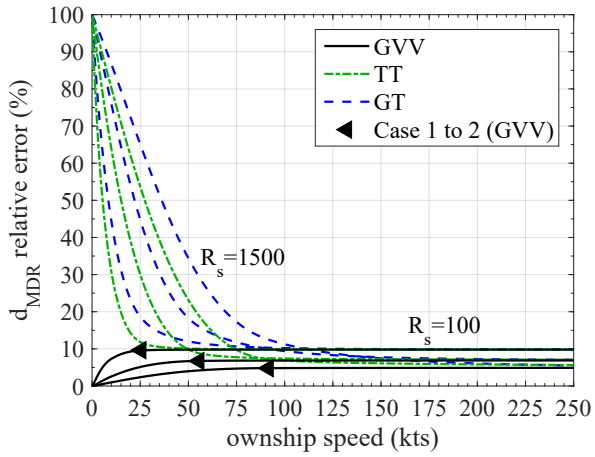
From each subplot in Fig. 2.10 we see that the GVV method produces the smallest relative error. We also see that the TT method generally produces a relative error less than the GT method except when v_i approaches zero as seen in Fig. 2.10(b). In Fig. 2.10(a) the relative error in d_{MDR} is plotted versus $1/\dot{\phi}_{\max}$ and τ with the remaining parameters fixed at their nominal values and with the ownship speed equal to 150 kts. The value of τ is varied between 0 and 1 s, while $\dot{\phi}_{\max}$ is varied from 30 deg/s to infinity. Plotting against the inverse



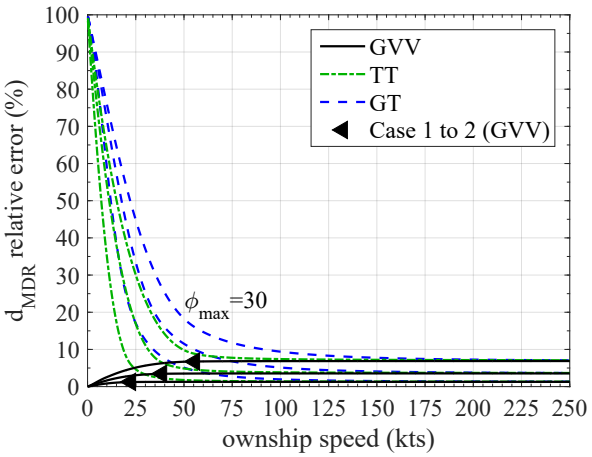
(a) GVV relative to TGVV as a function of $\dot{\phi}_{\max}$ and τ.



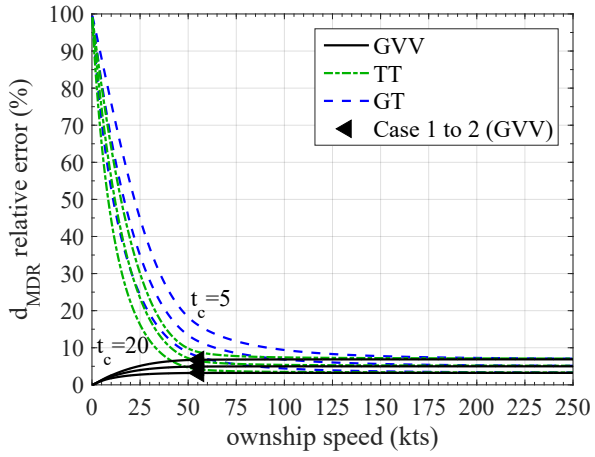
(b) Relative to TGVV as a function of v_i.



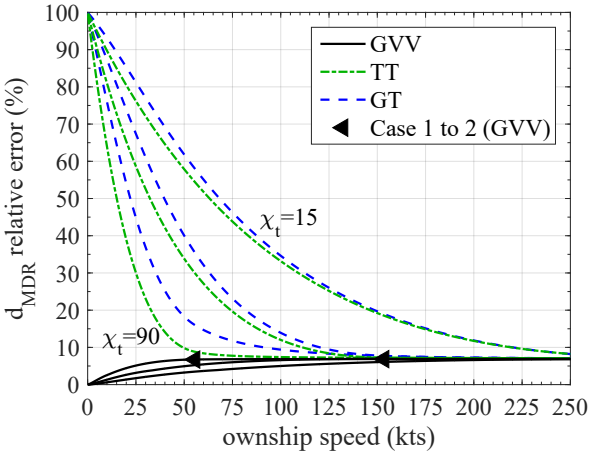
(c) Relative to TGVV as a function of R_s.



(d) Relative to TGVV as a function of $\phi_{\max}$.



(e) Relative to TGVV as a function of t_c.



(f) Relative to TGVV as a function of χ_t.

Figure 2.10: Percent relative error of d_{MDR}, compared to TGVV, as a function of v_o, v_i, R_s, $\phi_{\max}$, t_c, χ_t, $\dot{\phi}_{\max}$ and τ.

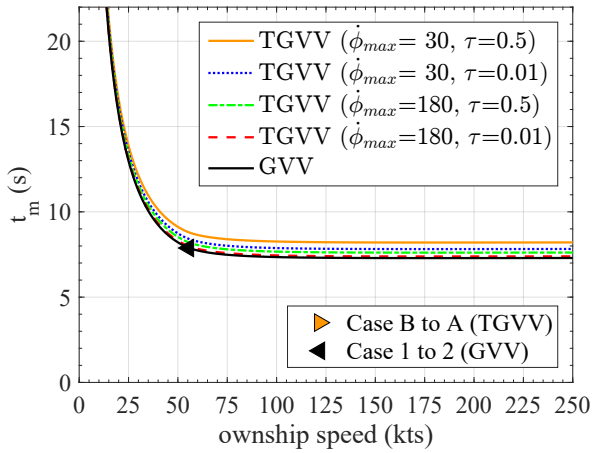
of $\dot{\phi}_{\max}$ improves the presentation of the data and facilitates interpretation of the result. By continuously varying $\dot{\phi}_{\max}$ and τ, the effects of each variable on the accuracy of the GVV method can be more easily seen. From the subplot it can be seen that the relative error does decrease as τ decreases, however, it does not decrease to zero. As the inverse of $\dot{\phi}_{\max}$ decreases, however, the relative error does go to zero for all values of τ. This shows that as the bank-angle maneuver of the ownship becomes more instantaneous, the results of the geometric GVV method converge to those of the TGVV method.

The results presented thus far show that the TT and GT methods converge to the GVV method for large ownship speeds relative to the intruder speed, and the GVV method converges to the TGVV method for fast bank-angle transients and small bank angles. We can thus conclude that the TT and GT methods provide good approximations for the minimum detection range at large ownship speeds relative to the intruder speed, fast bank-angle transients, and small bank angles.

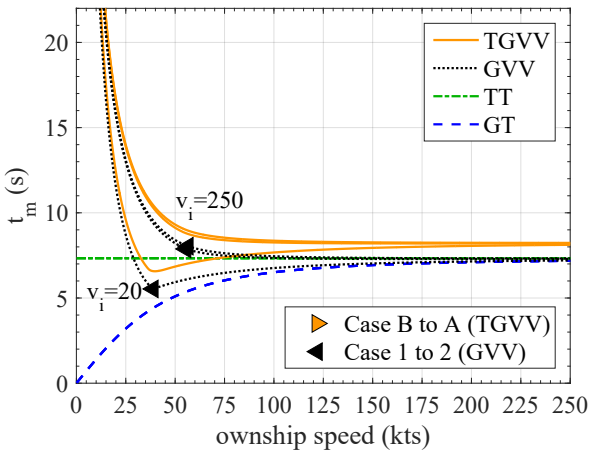
Although the main focus of this paper is on methods for the calculation of minimum detection range, the time to maneuver is a quantity of significant importance and accompanying results for this parameter are presented in Fig. 2.11.

As stated previously, the minimum detection range of the geometric methods (GVV, TT, and GT) approach linear functions for large ownship speeds. From Fig. 2.11, a similar observation can be made for the time to maneuver. The TT method predicts a maneuver time that is constant with ownship speed. Comparing Eqs. 2.3 and 2.4 and noting that the maneuver time t_m is the same as the turn time t_t for the TT method, shows that the predicted TT maneuver time is $\sqrt{\frac{2R_s \cot \phi_{\max}}{g}}$, which is independent of the ownship speed. For large ownship speeds the GT and GVV methods converge to the maneuver time value predicted by the TT method. In Fig. 2.11(c), we see that as R_s increases the predicted maneuver times also increase. From Fig. 2.11(d), we see that as $\phi_{\max}$ increases the maneuver times decrease.

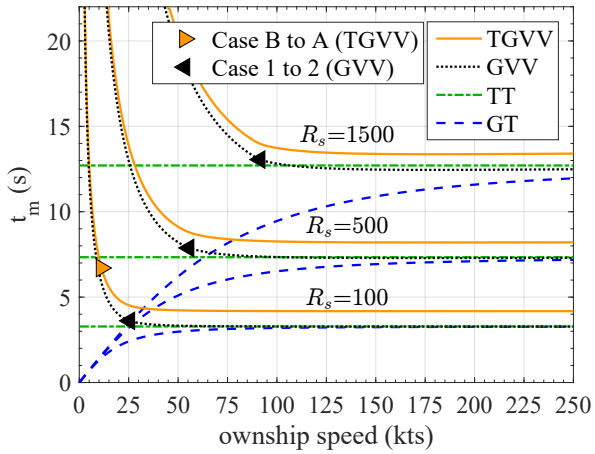
From Figs. 2.11(b) and 2.11(f) we see that the TGVV and GVV methods account for variations v_i and χ_t, as seen by three sets of lines, whereas, the TT and GT methods do not account for them. This agrees with the expressions for the predicted time to maneuver in the TT and GT methods where the predicted t_m for the GT method can be derived from Eq. (2.5) as $\frac{v_o}{g \tan \phi_{\max}} \cos^{-1}\left(\frac{v_o^2}{v_o^2 + R_s g \tan \phi_{\max}}\right)$.



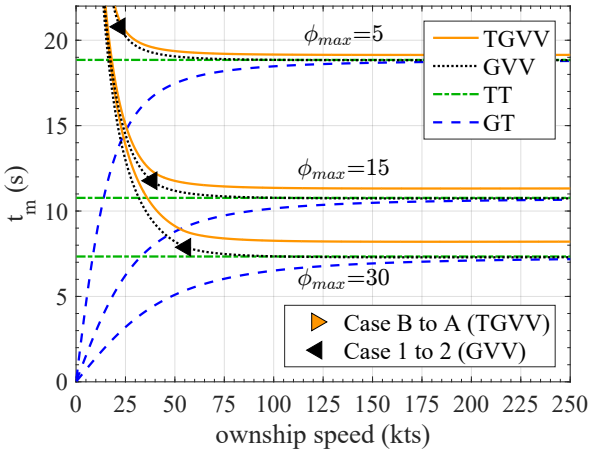
(a) Comparison as a function of $\dot{\phi}_{\max}$ and τ.



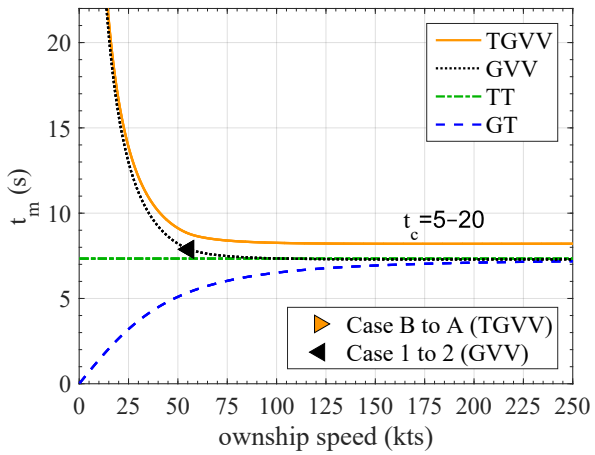
(b) Comparison as a function of v_i.



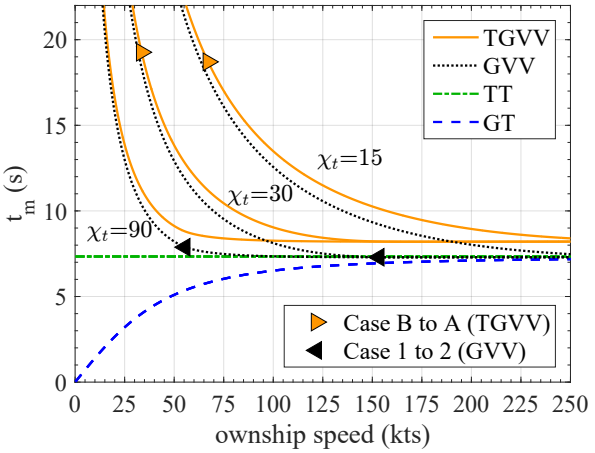
(c) Comparison as a function of R_s.



(d) Comparison as a function of $\phi_{\max}$.



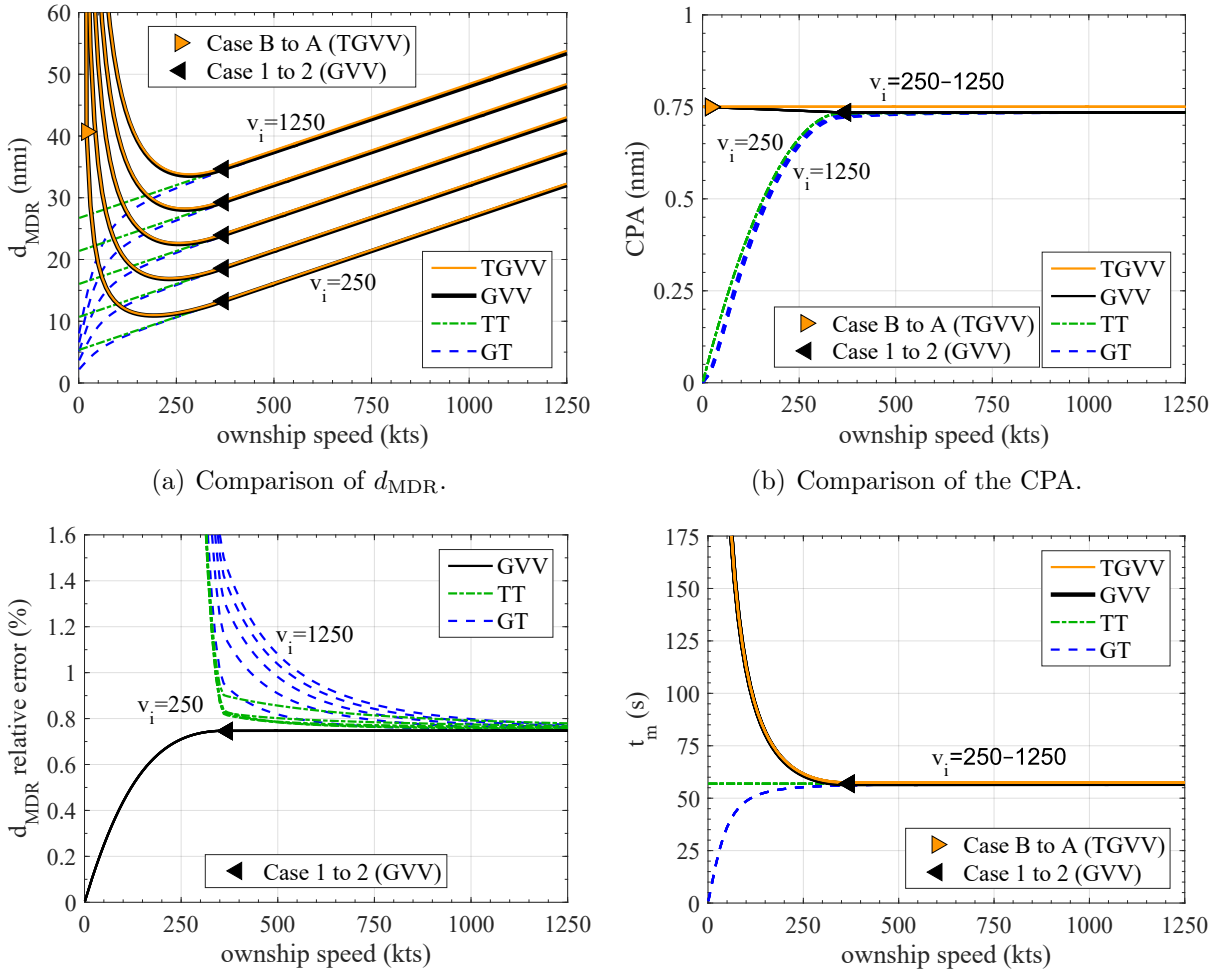
(e) Comparison as a function of t_c.



(f) Comparison as a function of χ_t.

Figure 2.11: Comparison of t_m as a function of v_o, v_i, R_s, $\phi_{\max}$, t_c, χ_t, $\dot{\phi}_{\max}$ and τ.

The results presented thus far have used parameter values consistent with a collision-avoidance encounter. Results are now presented for parameter values more consistent with a self-separation scenario. These results use the following parameters: $R_s = 0.75$ nmi, which is the lateral UAS well-clear requirement, $\phi_{\max} = 5$ deg, $\chi_t = 15$ deg, $t_c = 20$ s, which includes tracking and typical pilot response delay with air traffic control (ATC) interaction, $\dot{\phi}_{\max} = 10$ deg/s, $\tau = 0.5$ s, $v_o =$ 0–1250 kts, and $v_i = 250, 500, 750, 1000, 1250$ kts. The results for these parameters are shown in Fig. 2.12.



(a) Comparison of d_{MDR}.

(b) Comparison of the CPA.

(c) Percent relative error of d_{MDR} compared to TGVV.

(d) Comparison of t_m.

Figure 2.12: Self separation results.

From Fig. 2.12(a) it can be seen that the the minimum detection range is now on the order of 10 to 50 nmi instead of 2000 to 10000 ft. Figure 2.12(b) shows that the CPA of the TGVV method is equal to the value chosen for R_s of 0.75 nmi. We also see that the GVV method produces a CPA close to the desired value for R_s. In Fig. 2.12(c) we see that the relative error of the GVV method stays well under 1 percent. Finally in Fig. 2.12(d), we see that for large ownship speeds the time to maneuver has increased to about 57 s as would be expected for the self separation simulation parameters. Each of the subplots in Fig. 2.12 demonstrate that the GVV method can be used as an accurate approximation to the TGVV method when parameter values are aligned with those commonly found in self-separation encounters. Similarly, the TT and GT methods provide good approximations to the TGVV method, but only for large ownship speeds.

Finally, to complete our comparison of the four methods considered in this paper we present results that characterize the computational cost of each of the methods. These results are shown in Table 2.4 and include the average runtime of each method and the number of lines of code needed to implement each method. The average runtime for each method was based on 30,000 samples and was executed in Matlab on a 64-bit, 2.70 GHz, four-core, Intel I-7 laptop with 16 GB of RAM. For the TGVV method, numerical methods were used with a time step of 0.001 s. The number of lines of code required by each method was determined for a MATLAB implementation. The number of lines of code for the TGVV and GVV methods are only approximations as multiple lines of code could be combined, however, this does provide some insight into how much effort would be required to program each method. While the run time of each method is of interest, none of the implementations are expensive in terms of required computation time when compared to the time scales of the corresponding DAA maneuvers.

Table 2.4: Computational cost of each method from average runtime and lines of code.

Computational Cost				
	TGVV	GVV	TT	GT
Runtime (s)	1.77e-2	2.01e-5	1.19e-6	2.27e-6
Lines of code	≈200	≈50	1	1

CHAPTER 3. TARGET DETECTION AND TRACKING

Target detection and tracking is a vital component of a detect-and-avoid system. Target detection requires a detection device which is capable of receiving simultaneous measurements from multiple aircraft. Target tracking involves two steps: first, a method must be put into place that is capable of using imperfect measurements from the detection device to identify the total number of intruder aircraft at each time step. Second, the states of each of the identified intruder aircraft must then be estimated. The resulting target tracks will then be used for subsequent steps of the DAA system. Specifically, they are used for predicting the relative positions between the ownship and intruder aircraft in the future, where knowing these future positions allows us to determine if a collision is expected to occur. If a collision is predicted to occur, then a new collision avoidance path will be calculated and flown by the ownship aircraft.

An overview of the remainder of this chapter is as follows: In Section 3.1, the selection of radar as the detection device is explained. In Section 3.2, the selection of recursive-RANSAC as the tracker is motivated along with original research done to extend recursive-RANSAC to nonlinear models. Finally in Section 3.3, the implementation of an air-based radar detection device with extended recursive-RANSAC as the tracker is described with accompanying simulation results.

3.1 Detection Device

The FAA released a literature review of detect, sense, and avoid technologies in 2009 [22]. In this review two overarching technologies are discussed, cooperative SAA and non-cooperative SAA. While cooperative technologies would be preferable, it cannot be assumed that all aircraft will be equipped with such a system, therefore, this research focuses

on the use of non-cooperative technologies. Non-cooperative technologies can be broken into two categories, active systems and passive systems.

Collectively, passive sensors are ideal because they do not require the generation of power consuming signals, but instead rely on external methods to provide the detectable signal. Examples of passive systems include motion detection, electro-optical (E/O), infrared, and acoustic sensors. Passive E/O cameras are an attractive solution because of the vast availability of these sensors and the simplicity in interpreting the data from these sensors using the human eye. Additionally, camera technology is ideal because of the low cost, small size, and low power requirements; however, traditional cameras are not able to detect intruders far away, so high definition cameras are required. The main problem with these sensing devices is that they do not work in poor weather, at night, and they do not provide a direct measurement of range. These limitations are significant enough that an alternate solution is desired.

Active systems have the ability to overcome the problems mentioned for passive systems, and some examples of active systems include radar, laser, and sonar. Active systems have one main drawback when compared to passive systems. Active systems do not rely on external sources such as reflections from the sun or heat sources to provide a detectable signal of the target, instead, they utilize reflected signals that originate from the active sensor itself. The drawbacks of this difference are that active systems must create a signal to be transmitted that may require large amounts of power to create, and the generation of this signal makes the ownship highly visible to others where they may desire to remain unseen. This last issue is not as big of a concern for general civilian use, however, it would be a concern for military surveillance and reconnaissance type missions. The focus of this thesis is in using a detection device for the DAA problem among the general public, and not necessarily to be used by the military. As a result, active sensors appear to be the sensor of choice due to their ability to work in poor weather and at night, while also providing a direct measurement of range.

Radar has been under development for many years and has many attractive features that make it one of the best solutions for the DAA problem. Radar is ideal for situations where normal vision is impaired, such as poor weather or night time conditions. This is

because radar is an active sensor that generates a signal that is able to cut through poor weather. Furthermore, it uses its own reflected signal to detect an object as opposed to the reflection of light originating from the sun, which means it works irrespective of the amount of sunlight. Normal radar units tend to be large and heavy, but current work is being done at BYU to create a small, low-weight, low-power radar suitable for small UAS. Although the research in this thesis does not include the development of this radar, the tracker and DAA system described in this thesis have revolved around its use as the primary detection device.

3.2 Tracker

With radar selected as the primary detection device, a method must be put into place to use these radar measurements to detect the number of intruder aircraft and to estimate the states of each of the intruder aircraft within the field of view of the radar. For this thesis, we use a multiple target tracking algorithm that was originally developed here at Brigham Young University by Dr. Randy Beard and Peter Niedfeldt, known as recursive-RANSAC (R-RANSAC) [12, 23]. A couple of early applications of this algorithm include tracking multiple ground point scatters from an airborne synthetic aperture radar [24], and tracking multiple dynamic targets in clutter [25].

The continued use of R-RANSAC for multiple target tracking in various applications has been an active area of research for the past several years. Dr. Beard has led these efforts in conjunction with various graduate students. A brief description of these various implementations of R-RANSAC are given as follows. In Ref. [26], multiple targets are tracked using video taken from static platforms. In Ref. [27], multiple targets are tracked using video taken from an airborne camera. In Ref. [28], multiple landing sites for rotorcraft in urban environments, are quickly found from large terrain maps. In Ref. [29], cooperative target tracking is performed using moving camera platforms to geolocate the targets in an inertial frame.

There are two main contributions that we have made to the R-RANSAC algorithm, as a result of the research shown in this chapter. First, we extend the use of R-RANSAC to track multiple dynamic aircraft in a three dimensional inertial frame from either airborne or ground-based radar sensors, to be used in the DAA solution. Second, we perform a general

formulation of the extended recursive-RANSAC (ER-RANSAC) algorithm, which extends the general linear R-RANSAC algorithm to nonlinear systems. To explain the contributions of this thesis to the recursive-RANSAC algorithm in greater detail, the remainder of the section is organized as follows. In Section 3.2.1 the desired properties of a tracker are discussed along with an explanation of why recursive-RANSAC was chosen as the tracker for this research. In Section 3.2.2 an overview of the original recursive-RANSAC algorithm is given. Lastly, In Section 3.2.3 a description is given on how recursive-RANSAC has been extended to nonlinear models.

3.2.1 Desired Tracker Properties

Past research has been performed at Brigham Young University's MAGICC lab on the development of a DAA system. Specifically, research conducted by Klaus [30] involved creating a Matlab and Simulink simulation environment that uses an extended Kalman filter (EKF) to track a single intruder that flies at constant altitude, velocity, and heading. This simulation environment also included a model of a radar unit that provides a 120 deg field of regard (FOR) in azimuth and a 30 deg FOR in elevation.

This target detection and tracking system worked well under certain conditions, however, there are certain limitations that require further development. Specifically this system was only set up to track a single intruder aircraft that is not maneuvering and which is assumed to be approaching in front of the ownship. Additionally, this system does not consider measurement errors resulting from spurious measurements, missed measurements, and multiple measurements. Each of these limitations are addressed in the development of a revised target detection and tracking system which is described next.

Overtaking Intruders

The radar model previously implemented in Klaus's simulation included three forward-pointing radar units. These radar units were oriented in three distinct directions so that the return signal strength from each radar could be used to determine the azimuth angle to the target. The resulting surveillance area from this radar setup included 120 degrees of visibility

in the azimuth direction and 30 degrees in elevation centered about the nose of the aircraft. This surveillance area leaves a majority of the airspace around the ownship unmonitored. As long as the ownship maintains a large forward velocity relative to the other aircraft in the airspace, then all possible collision scenarios will occur in the forward direction, and the radar model implemented in Ref. [30] will be sufficient. All possible collision scenarios will only occur in the forward direction because the ownship will be traveling too fast for aircraft to overtake it from above, below or behind.

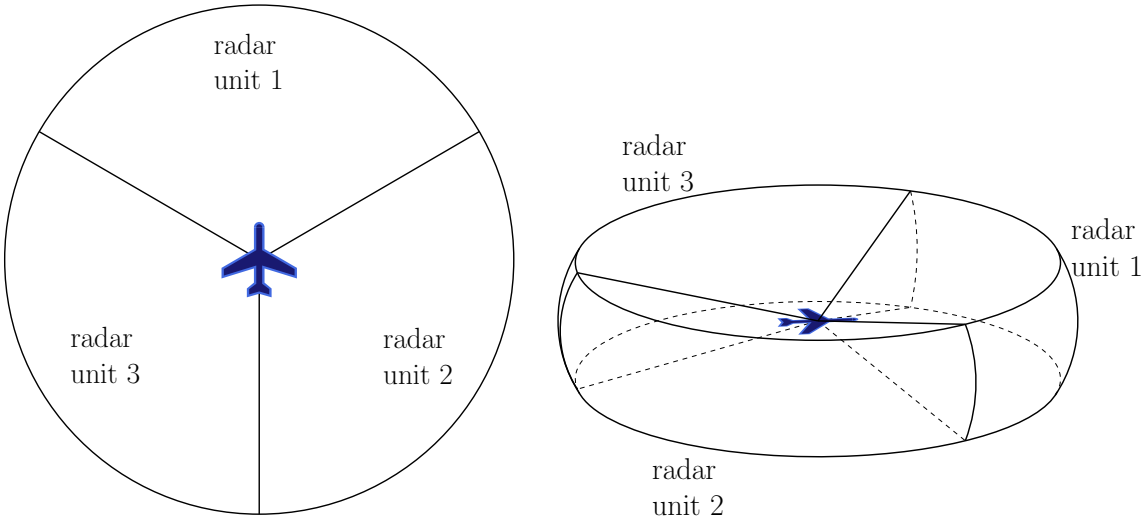
As the speed of the ownship decreases in comparison to the other airspace users, the radar model used in Ref. [30] begins to decrease in effectiveness. This is because the ownship will begin to appear stationary compared to the other aircraft and can therefore be easily overtaken from any direction including behind, below, or above. Small UAS will generally be traveling much slower than the associated aircraft they will be sharing the airspace with, therefore, these small UAS and other slow moving aircraft must ideally expand their monitored airspace to include full 360 degree coverage horizontally and vertically. In order to get full 360 degree coverage horizontally and vertically, the field of view of the detection devices must be known. One possible solution would be to use a single omni-directional detection device. This solution would be ideal because it would reduce the number of detection devices down to the minimum value of one. In practice omni-directional devices are difficult if not impossible to find. Most detection devices will be pointing in a certain direction with a specific field-of-view capability.

Another possible solution would be to use multiple single-direction detection devices. The idea would be to orient each of the devices in different directions until the entire airspace is being monitored around the ownship.

As radar has been selected as the detection device for this research, and since there is a group at Brigham Young University currently developing radar hardware to be used in our DAA system, we realize that a more feasible step in attaining full 360 degree coverage horizontally and vertically would be to first achieve full 360 degree coverage horizontally utilizing the radar hardware being developed at Brigham Young University. Although the airspace above and below the ownship will be unmonitored with this type of detection device,

this is a reasonable step to take because encounters in the horizontal direction have a larger probability of occurring than encounters in the vertical direction.

The main design constraints for the radar unit being developed here at BYU were to minimize SWaP and to provide a reasonably large field of view of approximately 120 degrees horizontally and 30 degrees vertically. The design constraint of a 120-degree horizontal FOR was used because for planar-array radar, the received power decreases as the angle increases off bore-sight, and are generally only useful up to 60 degrees off bore-sight. Using these design parameters, 360 degree coverage horizontally could be achieved by mounting three of these radar units around the ownship oriented at azimuth angles spaced 120 degrees apart from each other. An illustration of this setup can be seen in Figure 3.1.



(a) Top view of three radar setup.

(b) Isometric view of three radar setup.

Figure 3.1: Full 360 degree horizontal coverage achieved with three radar units.

The three radar setup is shown from a top down view in Figure 3.1(a) and a 3D view in Figure 3.1(b). From these figures we see that the first radar unit is pointing out the nose of the aircraft, the second radar unit is pointing backwards and to the right, and the third radar unit is pointing backwards and to the left. The specific diagram shown in Figure 3.1 only uses a vertical field of view of approximately 30 degrees because that was the original design constraint used for the radar hardware, however, each of the three radar units could

theoretically be modified to increase the vertical field of view to plus or minus 60 degrees for a total of 120 degrees, which would have a significant impact on the total surveillance area of the radar detection system.

Multiple Intruders and Measurement Errors

The target detection and tracking algorithm previously implemented in Klaus's simulation utilized an EKF to track a single intruder without clutter measurements and without missed measurements. While this was a necessary first step, we realize that one of the requirements needed to make a tracking algorithm satisfactory is the ability to detect and track multiple targets in the presence of clutter and missed measurements. This requirement is necessary because the airspace is shared by many aircraft and almost all detection devices contain these types of measurement errors. There are numerous multiple target tracking (MTT) algorithms available in the scientific community and a few of them will be discussed below. Each of these MTT algorithms has its own strengths and weaknesses, however, by comparing each of these algorithms we will show that recursive-RANSAC is well suited for use with radar as the detection device, and for use on-board a small UAS.

Global Nearest Neighbor (GNN) [31]. Global nearest neighbor is the simplest MTT algorithm where each track is updated using the nearest-neighbor measurement. Some limitations of GNN are that it requires prior knowledge of the number of targets, it does not include track management, and it is not robust to clutter.

Probabilistic Data Association (PDA) [32]/ *Joint Probabilistic Data Association (JPDA)* [33]. Probabilistic data association weights measurements within a gate based on their statistical distance, the probability of false alarms, and the probability of new targets. The weights of all the measurements within the gate sum to 1 and a combined single measurement is used to update the track. PDA does well in tracking single targets in clutter and can also track multiple targets in clutter as long as the targets are well-separated; however, if the targets are not well separated then the JPDA algorithm can be used to weight measurements that are shared between multiple tracks. As with GNN, PDA requires that the number of targets be known and does not include track management.

Multiple Hypothesis Tracking (MHT) [34]. Multiple hypothesis tracking calculates probabilities for three types of hypotheses for new measurements. The new measurements can either come from previously known targets, new targets, or the measurements may be false measurements. All combinations of hypotheses are compared, and the combination with the highest probability is selected. Some advantages of MHT are that track management is built right into the algorithm and it is robust to clutter. Two main drawbacks of MHT are that it is computationally complex and creating the code is known to be difficult.

Probabilistic Hypothesis Density (PHD) [35, 36]. Probabilistic hypothesis density filters avoid explicitly associating measurements with individual tracks; instead, PHD updates every track with every measurement, which results in the formation of many new tracks at each time step. Each of these tracks are assigned a probability and only the tracks with a high probability are retained for the next time step. PHD does well in high-clutter, high-target-density environments, however, it struggles to retain tracks when measurements are missed. Other advantages of the PHD filter are that it can track targets with nonlinear dynamics and track management is built directly into the algorithm.

Particle Filter (PF) [37]. Particle filter tracking requires a sample size N_s of the number of particles that are randomly assigned values from the state space. These particles are propagated forward in time with random process noise and then assigned a likelihood based on new measurements. Finally a re-sample is done based on the likelihood of the current particles. The re-sample causes the particles to converge to locations where targets exist. Some advantages of this algorithm are that it can track nonlinear, non-Gaussian systems in cluttered environments. For multiple target tracking the number of targets must be known. Another drawback is that the number of particles needed increases with the size of the state space and with the number of targets; large amounts of particles can dramatically increase the computational cost.

Recursive-RANSAC (R-RANSAC) [12]. Recursive-RANSAC updates each track with all measurements within a specified gate. Any measurements not associated with a track are used to create a new track using the RANSAC algorithm; therefore, track management is built into the algorithm. The inlier ratio is used to identify good tracks and the top $\mathcal{M}$ tracks are kept even if they have bad inlier ratios. This prevents tracks from being lost

across multiple time steps even when there are missed measurements. R-RANSAC is robust to clutter and missed measurements, and is computationally efficient.

Many detection devices, including radar, have two main sources of error that must be addressed when implementing a MTT algorithm. These sensors often produce measurements that include clutter/spurious measurements, and missed measurements at random time steps; therefore, it is desired to use an MTT algorithm which is robust to clutter and missed measurements. In addition to measurement errors, the MTT algorithm must also address the specific requirements of a DAA system being run on board a small UAS. Two of these requirements include: the MTT needs to include track management to be fully autonomous, and it needs to be computationally efficient to be run by a small lightweight processor on board a small UAS. With each of these requirements we notice that GNN, PDA, and PF do not include track management, PHD is not robust to missed measurements, and MHT is computationally complex. The R-RANSAC algorithm, on the other hand, includes track management, is robust to clutter and missed measurements, and is computationally efficient. Since Recursive-RANSAC meets each of the desired properties, it has been selected as the MTT algorithm for the DAA system described in this thesis.

Maneuvering Intruders

The final improvement made to the previous target detection and tracking system is the ability to track maneuvering intruder aircraft. In the previous tracking system, various encounter scenarios were created with each aircraft flying at a constant altitude, velocity, and heading. The resulting dynamic model used in the EKF was a constant-velocity model in the horizontal plane with a known altitude, where the estimated states included the intruders' north position, east position, velocity, and heading. The simulated radar model only provided range and azimuth measurements, and did not provide elevation measurements, therefore, the altitude of the intruder aircraft was unobservable and it had to be assumed that this quantity was known.

While it may be true that most aircraft detected will be flying at a nearly constant altitude, velocity, and heading, this research has expanded the tracker capabilities to be more robust to maneuvering intruder aircraft. Specifically we consider aircraft that may be

turning, climbing or descending, and accelerating or decelerating. This improved capability requires the use of an expanded set of states that are capable of capturing each of these maneuvers. One version of this expanded set of states is based on a linear constant-acceleration model. In this model the states are p_n, p_e, h, $\dot{p_n}$, $\dot{p_e}$, $\dot{h}$, $\ddot{p_n}$, $\ddot{p_e}$, and $\ddot{h}$. A second version of this expanded set of states is based on a nonlinear constant-acceleration model. In this model the states are p_n, p_e, h, V_g, χ, γ, $\dot{V}_g$, $\dot{\chi}$, and $\dot{\gamma}$. In both of these state descriptions, we notice that the altitude of the aircraft is now being estimated. To make the altitude state observable, we have assumed that the radar measurements have also been expanded to include elevation angle measurements in addition to range and azimuth angle measurements.

While both versions of the expanded sets of states are valid for capturing the dynamic maneuvers of an aircraft and for propagating these dynamic states into the immediate future, only the nonlinear version is successfully able to propagate accurate state estimates into the distant future. This is because the flight characteristics of an aircraft are more accurately described using the nonlinear constant-acceleration model. Specifically these nonlinear states are able to accurately model an aircraft which is 1) turning at a constant rate to change its heading or loitering about a specific location, 2) accelerating or decelerating at a fixed value resulting from a take-off or landing type approach or from a desire to reach an optimal airspeed based on current wind conditions and desired flight time, 3) changing its climb or descent angle at a constant angular rate resulting from an aircraft initiating a transition from one altitude level to another or as a result of an aircraft deciding to change from climbing to descending or descending to climbing. Using constant-acceleration values with the linear version of the states will result in a steady-state heading and flight path angle with the acceleration along this directional vector, thus neglecting the continuous turning of the aircraft and the continuous changing of flight path angle of the aircraft. On the other hand, using constant-acceleration values with the nonlinear states will result in a future trajectory that does not approach a single directional vector, but instead is able to maintain a continuous arching trajectory expected from a constant turning maneuver or climbing or descending at a constant angular rate.

Since the nonlinear model more accurately captures the turning and climbing behavior of aircraft, it is better suited to predict where the aircraft will be at some future point in

time. While the nonlinear model has these added benefits, there are other factors that make the linear model an attractive candidate that will be discussed later. As such, both the linear and nonlinear models will be used to obtain tracking results for this thesis.

3.2.2 Overview of Recursive-RANSAC

As discussed in Section 3.2.1, R-RANSAC has been selected as the MTT algorithm for this research. Also in that section we noted that to track maneuvering targets, an expanded set of states based on a constant acceleration model is desired. This model and subsequent states can either be based on linear or nonlinear system dynamics. In addition to the state equation being linear or nonlinear, the output equation can be linear or nonlinear depending on what type of detection device we use. Based on the fact that we are using radar as the detection device to provide measurements of range, azimuth, and elevation, and using either set of states we have defined previously, nonlinear output equations will result. There are however ways of transforming these nonlinear output equations resulting from the radar into a linear set of output equations, which means it is possible to create a fully linear set of state-space equations. Since the state-space equations for this problem can be expressed in either linear or nonlinear form, we therefore desired to use the R-RANSAC framework to track multiple maneuvering targets that are modeled by either linear or nonlinear state space equations.

Prior to this work, only linear methods have been used for the dynamic models and measurement equations of the targets being tracked with R-RANSAC; however, there is nothing in the general R-RANSAC framework that prevents its use with nonlinear methods. The ability to extend R-RANSAC to systems with nonlinear dynamics was first observed by Niedfeldt as found in his discussion of future work in his dissertation. In this discussion Niedfeldt states that any nonlinear filtering technique can be used within the R-RANSAC framework such as the extended Kalman filter, unscented Kalman filter, or others, and suggests that the only unresolved issue is how to appropriately initialize a new trajectory using RANSAC [12].

In the remainder of this section we give a brief overview of the generic RANSAC and Recursive-RANSAC frameworks. In Section 3.2.3 we then describe how we have been

able to successfully create an implementation of R-RANSAC that extends its use to include nonlinear state-space models.

RANSAC

In the world today, there are many types of sensors readily available to measure elements of the space around it. Often these measurements are sampled at discrete time intervals and the general trends of these data sets fit within the framework of known models. For example, it may be desired to estimate the parameters of a line or curve resulting from a set of measurements, or it may be desired to estimate the parameters of any other model. The process of estimation of a known model from a set of noisy data is referred to as regression, and many regression techniques have been created such as least squares, maximum likelihood, and many others. These techniques are batch processes that use an overdetermined set of measurements to estimate a single set of parameters. They do so by creating a simple model that follows the general trends of the data while minimizing some error metric. Among these regression techniques, it is often required that the received measurements originate from a single signal and that gross errors are not present. If these requirements are not met, the parameter estimation scheme will likely diverge from the true parameters resulting in a failure of the regression method. The limitations of these traditional regression techniques have resulted in the formation of many heuristics that attempt to resolve these known issues.

Due to the potential failure of traditional regression techniques to converge to the true parameters of a model in the presence of multiple signals and gross errors, and the various attempts to solve these issues with heuristic methods, a novel method has been developed that estimates the parameters of a single signal in the presence of other signals and gross errors and is referred to as Random Sample Consensus (RANSAC) [38]. The novelty of the RANSAC method stems from the fact that instead of using large amounts of data to estimate the parameters of a single signal, a minimal subset of measurements are selected randomly from the batch of data and are used to initialize a model hypothesis. Additional measurements are then added to this data set that fit within the current model hypothesis. The size of the minimal subset is based on observability principles, and the inclusion of additional measurements is determined by the parameters of the current model

hypothesis and some error tolerance specification. This newly increased data set is referred to as the consensus set hypothesis and is then used to create an updated model hypothesis. This process is then repeated a predetermined number of times, and the estimated model parameters from the iteration with the largest consensus set are used as the final estimated model parameters.

Niedfeldt does a great job outlining the mathematical representation of RANSAC and it has been repeated here in Algorithm 1. For a complete overview of this algorithm the interested reader is referred to Ref. [12]. Variables used within this algorithm include discrete time steps k, the measurement window length N, the time step of the first measurement in the current measurement window is defined as $k_N \triangleq k - N + 1$, the past N-length window of time samples $k_N : k$, the complete measurement set across the entire measurement window $\mathcal{Z}_{k_N:k}$, a minimum subset of s measurements as $\mathcal{S}_{\mathbf{q}} \in \mathbb{S}^s_{k_N:k}$ where $\mathbf{q}$ represents the indices of the randomly selected measurements, the model hypothesis estimate $\hat{\mathbf{x}}'$, a user defined function $g()$, the hypothesis consensus set χ', a user specified function Inlier(), error threshold parameter for RANSAC τ_{R}, an optional early termination parameter γ, and the total number of iterations ℓ.

Within this algorithm there are three parts which are of particular interest. First, we see that each model hypothesis $\hat{\mathbf{x}}'$ is formed according to a general function $g()$ using the minimum subset $\mathcal{S}_{\mathbf{q}}$ (Line 3). Second, the consensus set hypothesis χ' is found through some user defined function Inlier() using the measurement set $\mathcal{Z}_{k_N:k}$, the model hypothesis $\hat{\mathbf{x}}'$, and some user specified error tolerance parameter τ_{R} (Line 4). Third, we see that the final estimation of the model parameters is obtained by smoothing the best estimate using the consensus set (Line 12). The key observation from these three parts of the algorithm is that specific methods have not been defined; instead, the user is free to use any methods they desire.

Table 3.1: Algorithm for RANSAC

Algorithm 1 Random Sample Consensus (RANSAC) Algorithm

Input: Measurement set $\mathcal{Z}_{k_N:k}$, number of iterations ℓ,
error threshold τ_{R}, stopping criteria γ.

for ℓ iterations **do**
  Randomly select a minimum subset $\mathcal{S}_{\mathbf{q}} \in \mathbb{S}^{s}_{k_N:k}$
  Generate a hypothesis $\hat{\mathbf{x}}' = g(\mathcal{S}_{\mathbf{q}})$
  Compute the hypothesis consensus set, $\chi' = \mathrm{Inlier}(\mathcal{Z}_{k_N:k}, \hat{\mathbf{x}}', \tau_{\mathrm{R}})$
  if new hypothesis has larger consensus set than previous hypothesis **then**
    Store current hypothesis.
  end if
  if $|\chi'| \geq \gamma N$ **then**
    break
  end if
end for
Smooth best estimate using consensus set.

Recursive-RANSAC

The major limitations of the RANSAC algorithm described above are that it can only estimate the parameters of a single signal and it does not have the ability to update previously found models when subsequent measurements are received. As such, a new algorithm known as recursive-RANSAC (R-RANSAC) has been developed which extends the traditional RANSAC algorithm to overcome these issues. The basic idea of R-RANSAC is that previously found model estimates from RANSAC are preserved between time steps and these model estimates are recursively updated as new measurements are received at each time step. The measurements received at each time step are only used to update existing model estimates if they are an inlier to these models. If the current measurements are not an inlier to any of the existing models, then they are each used within the RANSAC algorithm as one of the minimum subset measurements to form a new model estimate. Only

model hypotheses with a certain level of support are deemed as valid. As multiple model hypotheses may end up tracking the same signal, these models are merged together. Finally, as the number of model estimates continues to grow, these models are pruned to keep a predetermined maximum number of models selected from the best models.

Similar to the RANSAC algorithm described above, Niedfeldt provides a good mathematical representation of R-RANSAC and it has been repeated here in Algorithm 2. The details of this algorithm can similarly be found in Ref. [12]. As R-RANSAC is largely used for state estimation of multiple targets, the term "model estimates" originating from RANSAC is interchangeably referred to as "state estimates," or "tracks."

Variables used within this algorithm, not previously defined, include the previous measurement window $\mathcal{Z}_{k_N:k-1}$, current measurement scan $\mathcal{Z}_k$, the association matrix J, an indexing variable for the current measurement scan i, an indexing variable for each of the tracks j, the i^{th} row and j^{th} column of the association matrix J_{ij}, the i^{th} measurement from the current time step $\mathbf{z}_k^i$, the j^{th} track estimate at the current time step $\hat{\mathbf{x}}_k^j$, error threshold parameter for R-RANSAC τ_{RR}, the empty set $\emptyset$, the measurement weighting for the i^{th} measurement and the j^{th} track w_{ij}, a minimum subset that includes the i^{th} current measurement $\mathcal{S}_{\bar{\mathbf{q}}} \in \{\mathbb{S}_{k_N:k-1}^{s-1} \times \{\mathbf{z}_k^i\}\}$, the consensus set of the j^{th} track at the current time step χ_k^j, the inlier ratio of the j^{th} track at the current time step ρ_k^j, survival probability p_s, the number of consecutive missed detections MD, number of stored tracks $\mathcal{M}$, good track threshold τ_ρ, the lifetime of the j^{th} track at the current time step t_k^j, and the minimum lifetime threshold τ_{T}. Although not seen in this algorithm, the j^{th} R-RANSAC track is stored as a six-tuple $\mathcal{M}^j = (\hat{\mathbf{x}}^j, P^j, \chi^j, \rho^j, t^j, \mathcal{L}^j)$, where P^j is the state covariance, and $\mathcal{L}^j$ is the track label.

Within the mathematical representation of the R-RANSAC algorithm there are four parts of interest that each occur within the current time step, k. First, we see that the state estimates from each track are propagated forward in time (Line 2). Second, the association matrix J is computed by determining if each of the i measurements $\mathbf{z}_k^i$ from the current measurement scan $\mathcal{Z}_k$ is an inlier to each of the j stored models $\hat{\mathbf{x}}_k^j$ using some user specified error tolerance parameter τ_{RR} (Line 3). This is done through some user defined function if Inlier(). Third, new tracks are formed from RANSAC using each of the current

measurements that are not an inlier to any of the existing tracks (Line 7). Fourth, current measurements that are an inlier to existing tracks are used to update each state estimate $\hat{\mathbf{x}}_k^j$ using the measurement weighting w_{ij} (Line 9). The same key observation is made from these four parts of the R-RANSAC algorithm as was made from the highlighted parts of the RANSAC algorithm; specific methods for each of these four parts have not been defined, but are instead free to be used with any methods the user desires.

Table 3.2: Algorithm for R-RANSAC

Algorithm 2 Recursive-RANSAC(R-RANSAC) Algorithm

Input: Previous measurement window $\mathcal{Z}_{k_N:k-1}$, current measurement scan $\mathcal{Z}_k$, measurement window length N, number of stored tracks $\mathcal{M}$, error threshold τ_{RR}, good track threshold τ_ρ, minimum lifetime threshold τ_{T}.

1: **for** each time step k **do**

2: Propagate state estimate of all active tracks.

3: Compute the association matrix J, where $J_{ij} = \begin{cases} 0 & \text{if Inlier}(\mathbf{z}_k^i, \hat{\mathbf{x}}_k^j, \tau_{\mathrm{RR}}) = \emptyset, \\ 1 & \text{otherwise.} \end{cases}$

4: Compute measurement weighting w_{ij} using the association matrix J.

5: **for** each $\mathbf{z}_k^i \in \mathcal{Z}_k$ **do**

6: **if** the measurement is an outlier to all tracks, $\sum_{\forall j}(J_{ij}) = 0$ **then**

7: Create new track using RANSAC, where the current measurement is in each randomly selected minimum subset, $\mathcal{S}_{\overline{\mathbf{q}}} \in \{\mathbb{S}_{k_N:k-1}^{s-1} \times \{\mathbf{z}_k^i\}\}$.

8: **else**

9: Update $\hat{\mathbf{x}}_k^j$ using w_{ij} and $\mathbf{z}_k^i$.

10: **end if**

11: Update the consensus set χ_k^j and the inlier ratio $\rho_k^j = \frac{|\chi_k^j|}{N}$.

12: **end for**

13: Kill tracks with probability $(1 - p_S)^{\mathrm{MD}}$.

14: Merge and prune to keep best $\mathcal{M}$ tracks.

15: Identify good tracks, according to $\rho_k^j \geq \tau_\rho$ and $t_k^j > \tau_{\mathrm{T}}$.

16: **end for**

3.2.3 Extended Recursive-RANSAC (ER-RANSAC)

As explained earlier, in all previous implementations of R-RANSAC only linear methods have been used for the the dynamic models and measurement equations of the targets being tracked; however, from our overview of the general RANSAC and R-RANSAC algorithms, we have seen that both of these algorithms have been designed to be very modular and there is nothing which restricts their use to linear methods. In this section we describe how we have successfully created an implementation of R-RANSAC that extends its use to include nonlinear state-space models, which is called extended recursive-RANSAC (ER-RANSAC). First, we will define the implementation details required to extend the R-RANSAC algorithm to the nonlinear case, followed by the details required to extend the RANSAC algorithm.

Nonlinear Extension of R-RANSAC

As was originally stated by Niedfeldt in his dissertation, the R-RANSAC algorithm can be used with nonlinear filtering techniques such as the extended Kalman filter, unscented Kalman filter, or others. He also explained how the R-RANSAC framework can be extended to nonlinear systems by replacing Lines 2, 7, and 9 in Algorithm 2 with the appropriate nonlinear propagation, hypothesis generation, and update equations, respectively [12]. We add that Line 3 must also be replaced by the appropriate nonlinear inlier detection function.

In our implementation of a nonlinear version of R-RANSAC we have chosen to use an extended Kalman filter (EKF). For our system dynamics and measurements we assume time-invariance. The general state space model for a nonlinear time-invariant system is shown as

$$\mathbf{x}_k = \mathbf{x}_{k-1} + dt \cdot f(\mathbf{x}_{k-1}) + \mathbf{w}, \tag{3.1}$$

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}, \tag{3.2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{y} \in \mathbb{R}^m$ is the measurement vector, dt is the time step, and $\mathbf{w}$ and $\mathbf{v}$ are zero-mean Gaussian process and measurement noise with covariance $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$, respectively.

Assuming the states and covariance matrix P have been previously initialized, the EKF propagation of the states and covariance are given by

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + dt \cdot f(\hat{\mathbf{x}}_{k-1}), \tag{3.3}$$

$$A = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k}, \tag{3.4}$$

$$P = P + dt(AP + PA^\top + Q), \tag{3.5}$$

where $A \in \mathbb{R}^{n \times n}$ is the Jacobian matrix resulting from taking the partial derivatives of the nonlinear function $f()$ with respect to each of the states $\mathbf{x}$ and evaluating these partial derivatives using the updated state estimate $\hat{\mathbf{x}}_k$.

The nonlinear measurement update equations for the states and covariance are given by

$$C = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k}, \tag{3.6}$$

$$L = PC^\top (R + CPC^\top)^{-1}, \tag{3.7}$$

$$P = (I - LC)P, \tag{3.8}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k + L(\mathbf{y}_k - h(\hat{\mathbf{x}}_k)), \tag{3.9}$$

where $C \in \mathbb{R}^{m \times n}$ is the Jacobian matrix resulting from taking the partial derivatives of the nonlinear function $h()$ with respect to each of the states $\mathbf{x}$ and evaluating these partial derivatives using the propagated state estimate $\hat{\mathbf{x}}_k$, $L \in \mathbb{R}^{n \times m}$ is the Kalman gain, and $I \in \mathbb{R}^{n \times n}$ is the identity matrix.

Having defined the nonlinear propagation and update steps we now define the nonlinear inlier detection function $\text{Inlier}(\mathbf{y}_k^i, \hat{\mathbf{x}}_k^j, \tau_{\text{RR}})$ as

$$\mathcal{I} = \{j : |\mathbf{y}_k^i - h(\hat{\mathbf{x}}_k^j)| < \tau_{RR}\}, \forall j = \{1, \ldots, \mathcal{M}\}, \tag{3.10}$$

where the error threshold parameter has been expanded as $\tau_{\text{RR}} \in \mathbb{R}^m$. The general choice of this parameter is now shown as

$$\tau_{\text{RR}} = \begin{bmatrix} s_1\sigma_{y_1} \\ s_2\sigma_{y_2} \\ \vdots \\ s_m\sigma_{y_m} \end{bmatrix}, \tag{3.11}$$

where σ_{y_i} is the standard deviation of each of the m elements returned from the detection device, and s_i is a tunable coefficient used to scale each of the elements of the error threshold parameter. From Equation (3.10) we see that the inlier function is calculating the difference between the current measurement and the predicted measurement based on the nonlinear output equation evaluated using the estimated states, and checking if this measurement error calculation is less than the error threshold parameter. The result of this inlier function will be a vector of elements containing logical true or false. If each element of this vector is true then that means the current measurement is an inlier to the model. If any of the elements in this vector are false, then the current measurement is not an inlier to the model.

The EKF propagation steps defined in Equations (3.3), (3.4), and (3.5) are now used in Line 2 of Algorithm 2, the EKF update steps defined in Equations (3.6), (3.7), (3.8), and 3.9 are used in Line 9, and the nonlinear inlier function defined in Equation (3.10) is used in Line 3. It is also required to use a nonlinear hypothesis generation step in Line 7 of Algorithm 2, however, the details of this step will be given in our description of the nonlinear extension of the RANSAC algorithm shown next.

Nonlinear Extension of RANSAC

From Niedfeldt's discussion of how to extend R-RANSAC to systems with nonlinear dynamics, he has stated that the only unresolved issue is how to appropriately initialize a new trajectory using RANSAC [12]. From the research performed in this thesis, we now show that this issue can be resolved by replacing Lines 3, 4, and 12 in Algorithm 1 with appropriate nonlinear model hypothesis generation, inlier detection, and smoothing functions, respectively.

From the three parts of the RANSAC algorithm that must be replaced by their appropriate nonlinear counterpart, the model hypothesis generation step requires the most detailed explanation. In the linear implementations of RANSAC this step has traditionally either used a least-squares or maximum likelihood estimator method to solve for the state estimate at the beginning of the measurement window $\mathbf{x}_{k_N}$. The nonlinear method we propose be used instead is the Gauss-Newton method [39]. The Gauss-Newton method is a nonlinear regression technique, which similar to linear techniques, aims to determine the parameters of a model which minimizes the sum of the squared error. The main caveat with these nonlinear regression techniques is that the solution must instead be found through an iterative approach. This means it takes longer to come to a solution and there are greater computational requirements. In addition to the increase in time and greater computational requirements, the iterative approach taken by the Gauss-Newton method also has a few other limitations that are worth noting. The solution may converge slowly, it may have oscillations, and it may never converge. Although these issues exist, we expect that their impact will be negligible because the Gauss-Newton method will be called many times within the RANSAC algorithm. If the Gauss-Newton method fails to converge with one set of data, it will be executed many additional times with various sets of new data and will likely converge to a solution on one of these iterations. The genius of the RANSAC algorithm is that it only selects the results from the iteration with the best results, and as such is already configured to reject solutions from the Gauss-Newton method which fail to converge.

Before we proceed to the derivation of the nonlinear version of RANSAC, we first provide a high level overview of the steps performed within this new implementation of RANSAC, as illustrated in Figure 3.2. The first two frames in this illustration help define the general problem we are confronted with. The remaining four frames show how the nonlinear Gauss-Newton solution is used within each of the major steps of the RANSAC algorithm.

In the first frame, we see the trajectory taken by a nonlinear dynamic target and a moving platform that is carrying a detection device with nonlinear output equations. The second frame show the measurements that result from the nonlinear output equation, with the final measurement labeled as $\mathbf{y}_k$. For the third frame, we show the selection of the

minimum subset using three measurements, where the final measurement is always included in this subset. For the fourth frame, the three measurements selected as the minimum subset are used within the Gauss-Newton method to iteratively solve for a model hypothesis, shown by the state estimate at the final time step $\hat{\mathbf{x}}'_k$. In all previous implementation of RANSAC, the model hypothesis is found at the initial time step of the measurement window. This nonlinear version of RANSAC could have followed a similar process, however, later in this chapter we show how there are benefits in solving for the state estimate at the final time step. In the fifth frame, we propagate the model hypothesis state estimate backwards in time to evaluate the nonlinear inlier function at each time step. The nonlinear inlier function results in a thresholding boundary in the inertial frame that will not necessarily be elliptical, as would be the case with radar measurements of range and azimuth angle. By evaluating this nonlinear inlier function at each time step, we can build the consensus set for the current model hypothesis χ'. This backwards propagation also results in an expression for the state estimate at the beginning of the time window $\hat{\mathbf{x}}_{k_N}$. The steps performed in frames three through five, are repeated multiple times to find a model hypothesis with the largest consensus set. After the model with the largest consensus set has been found, we proceed to the steps shown in the sixth frame.

In the sixth frame, we perform the smoothing step of RANSAC. For this specific implementation, this is done using an EKF, which propagates the state estimate and covariance matrix from the beginning of the measurement time window forward in time to the final time step, where we update along the way using the measurements in the consensus set. The evolution of the covariance matrix is also seen in the sixth frame, where the shape of this covariance is seen to be elliptical in nature. The final state estimate after the smoothing step $\hat{\mathbf{x}}_k$ and the covariance matrix P, are identified as the solution to the RANSAC algorithm, which are then returned to the ER-RANSAC algorithm.

Now that we have illustrated the steps performed within this new implementation of RANSAC, we are ready to begin our derivation of the Gauss-Newton solution needed within the nonlinear extended version of RANSAC. After the derivation of the Gauss-Newton solution, we mention some modifications we have added to minimize oscillations, prevent divergence, and help the solution converge to the correct states.

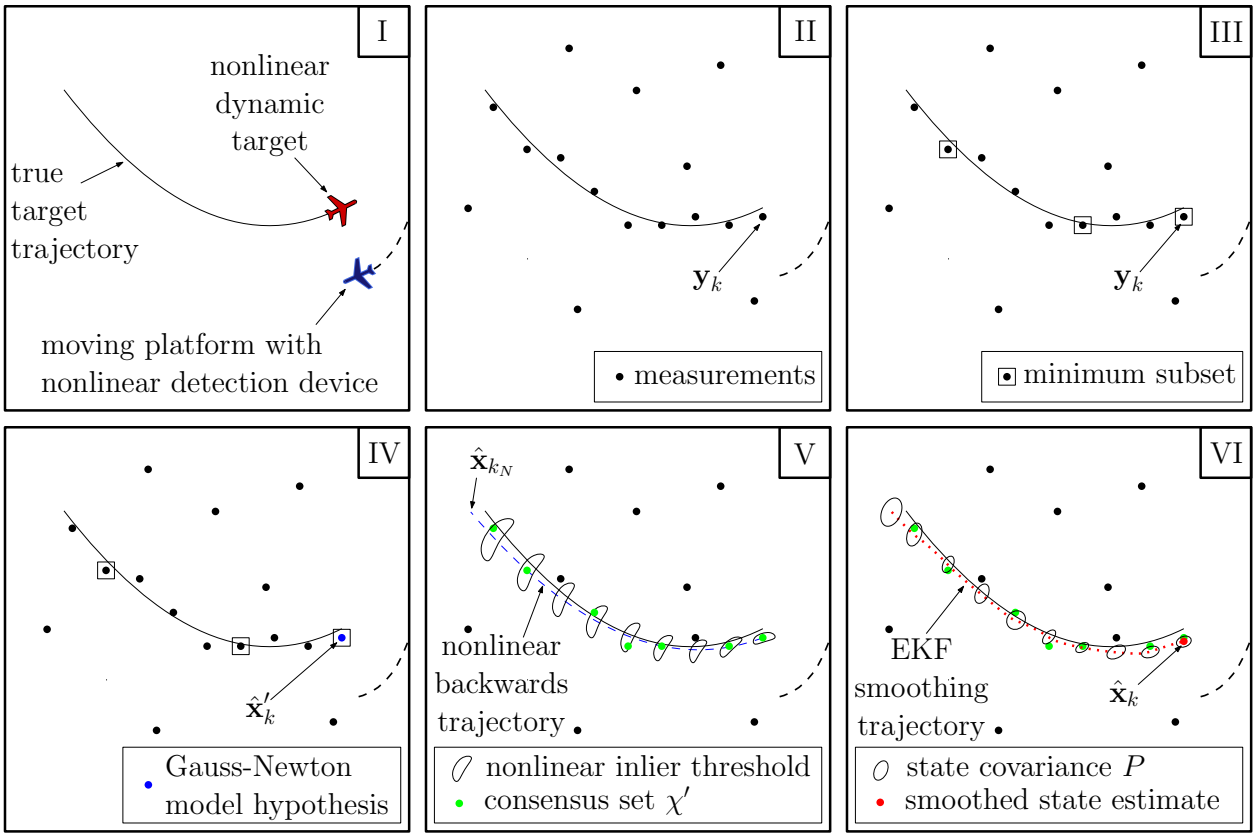


Figure 3.2: High level illustration of nonlinear version of RANSAC using the Gauss-Newton method.

While the linear implementations of RANSAC solve for the state estimate at the beginning of the measurement window, for the nonlinear version of RANSAC we have found it to be advantageous to solve for the state estimate at the current time step, $\mathbf{x}_k$. The primary reasoning for this change is to reduce the computational overhead, however, we discuss the advantages in detail in Section 3.2.4. Due to this change it becomes necessary to represent the state space equations by their evolution backwards in time as

$$\mathbf{x}_{k-1} = \mathbf{x}_k - dt \cdot f(\mathbf{x}_k) + \mathbf{w}_{k-1}, \tag{3.12}$$

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k. \tag{3.13}$$

Before we proceed we now make a definition that will greatly simplify the notation as

$$f'(\mathbf{x}_k) \triangleq \mathbf{x}_k - dt \cdot f(\mathbf{x}_k). \tag{3.14}$$

The idea behind this notation is that the multiple parts of Equation (3.12) that contain the state vector are now combined into a single nonlinear equation as

$$\mathbf{x}_{k-1} = f'(\mathbf{x}_k) + \mathbf{w}_{k-1}. \tag{3.15}$$

Now consider the sequence of N measurements characterized by

$$\begin{bmatrix} \mathbf{y}_k \\ \mathbf{y}_{k-1} \\ \vdots \\ \mathbf{y}_{k_N} \end{bmatrix} = \begin{bmatrix} h(\mathbf{x}_k) \\ h(\mathbf{x}_{k-1}) \\ \vdots \\ h(\mathbf{x}_{k_N}) \end{bmatrix} + \begin{bmatrix} \mathbf{v}_k \\ \mathbf{v}_{k-1} \\ \vdots \\ \mathbf{v}_{k_N} \end{bmatrix}. \tag{3.16}$$

Using Equation (3.15), we now write Equation (3.16) in terms of the states at the current time step $\mathbf{x}_k$ as

$$\begin{bmatrix} \mathbf{y}_k \\ \mathbf{y}_{k-1} \\ \mathbf{y}_{k-2} \\ \vdots \end{bmatrix} = \begin{bmatrix} h(\mathbf{x}_k) \\ h(f'(\mathbf{x}_k) + \mathbf{w}_{k-1}) \\ h(f'(f'(\mathbf{x}_k) + \mathbf{w}_{k-1}) + \mathbf{w}_{k-2}) \\ \vdots \end{bmatrix} + \begin{bmatrix} \mathbf{v}_k \\ \mathbf{v}_{k-1} \\ \mathbf{v}_{k-2} \\ \vdots \end{bmatrix}. \tag{3.17}$$

The goal is to solve Equation (3.17) for the states $\mathbf{x}_k$, however, the nonlinearities of these equations prevent us from doing so. As a result, we have selected the Gauss-Newton method to help us solve for these states. The key idea of the Gauss-Newton method is to express the original nonlinear equations in a linearized form using a first-order Taylor-series approximation from which we can solve for the desired variables using linear least-squares theory.

One use of a Taylor-series approximation is to provide an estimate of a nonlinear function close to a known point, however, when performing the Gauss-Newton method the Taylor-series approximation is used in a different way. In the Gauss-Newton method the estimated value of the nonlinear function is treated as a known quantity which we get in the form of measurements, and the point about which the function is linearized is treated as the unknown quantity. By providing an initial guess of this unknown point, we can then use an appropriate least-squares method on the Taylor-series expansion to solve for the residual. This residual is then used to update our guess of the unknown point. This process is repeated

in an iterative manner until the residual falls below some threshold τ_{GN} or the maximum number of iterations is reached ℓ_{GN}.

In our implementation, the initial guess of the state $\mathbf{x}_k$ will be represented by $\mathbf{x}_{k,j}$, and the updated guess of the state is represented by $\mathbf{x}_{k,j+1}$. From these definitions we therefore linearize the original equation using $\mathbf{x}_k = \mathbf{x}_{k,j+1}$ about the point $\mathbf{x}_{k,0} = \mathbf{x}_{k,j}$. Additionally since the process noise $\mathbf{w}$ is zero-mean Gaussian, we linearize the original equation about $\mathbf{w}_{i,0} = 0$, $\forall i = (k-1, k-2, \cdots, k_N)$. A useful definition that we will use in the derivation of the Taylor-series expansion is shown as

$$\begin{aligned} f_1'(\cdot) &\triangleq f'(\cdot), \\ f_2'(\cdot) &\triangleq f'(f'(\cdot)), \\ f_3'(\cdot) &\triangleq f'(f'(f'(\cdot))), \\ &\vdots \end{aligned}$$

where the numbered subscript represents the number of nested functions.

Starting with the first line of Equation (3.17), the Taylor-series expansion of the nonlinear term $h(\mathbf{x}_k)$ is shown as

$$h(\mathbf{x}_{k,j+1}) \approx h(\mathbf{x}_{k,j}) + \left(\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\bigg|_{\mathbf{x}_{k,j}} \right) (\mathbf{x}_{k,j+1} - \mathbf{x}_{k,j}), \tag{3.18}$$

where we now define $\triangle\mathbf{x} \triangleq \mathbf{x}_{k,j+1} - \mathbf{x}_{k,j}$. From this equation we notice that the partial derivative term is equivalent to the Jacobian matrix of the nonlinear function $h()$ evaluated at some specific point. This Jacobian matrix was mentioned previously in the description of the EKF and was defined as the matrix C. Throughout this section we will derive multiple variations of this Jacobian matrix, and as such this specific Jacobian matrix is defined as

$$C_1 \triangleq \frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\bigg|_{\mathbf{x}_{k,j}},$$

where the numbered subscript represents the number of nested functions around the variable $\mathbf{x}_{k,j}$ after the partial derivative has been evaluated at the specified point. This numbered

subscript also represents one more than the number of steps the variable $\mathbf{x}_{k,j}$ has been propagated into the past before being used to evaluate the partial derivative. For this particular C matrix, a subscript of 1 has been used because a single function $h()$ contains the variable $\mathbf{x}_{k,j}$ that has not been propagated into the past. Although the interpretation of this numbered subscript may seem trivial at this point, it becomes more useful in future Jacobian matrix definitions. Using these definitions, the Taylor-series expansion of the first row of Equation (3.17) is now expressed as

$$h(\mathbf{x}_{k,j+1}) \approx h(\mathbf{x}_{k,j}) + C_1 \triangle \mathbf{x}. \tag{3.19}$$

From the second line of Equation (3.17), the Taylor-series expansion of the nonlinear term $h(f'(\mathbf{x}_k) + \mathbf{w}_{k-1})$ is shown as

$$\begin{aligned}
h(f'(\mathbf{x}_{k,j+1}) + \mathbf{w}_{k-1}) &\approx h(f'(\mathbf{x}_{k,j}) + 0) \\
&+ \left(\frac{\partial h(f'(\mathbf{x}_k) + \mathbf{w}_{k-1})}{\partial f'(\mathbf{x}_k)} \bigg|_{\mathbf{x}_{k,j},0} \right) \left(\frac{\partial f'(\mathbf{x}_k)}{\partial \mathbf{x}_k} \bigg|_{\mathbf{x}_{k,j}} \right) (\mathbf{x}_{k,j+1} - \mathbf{x}_{k,j}) \\
&+ \left(\frac{\partial h(f'(\mathbf{x}_k) + \mathbf{w}_{k-1})}{\partial \mathbf{w}_{k-1}} \bigg|_{\mathbf{x}_{k,j},0} \right) (\mathbf{w}_{k-1} - 0), \\
&\approx h(f'(\mathbf{x}_{k,j})) \\
&+ \left(\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k} \bigg|_{f'(\mathbf{x}_{k,j})} \right) \left(\frac{\partial f'(\mathbf{x}_k)}{\partial \mathbf{x}_k} \bigg|_{\mathbf{x}_{k,j}} \right) \triangle \mathbf{x} \\
&+ \left(\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k} \bigg|_{f'(\mathbf{x}_{k,j})} \right) \mathbf{w}_{k-1}, \\
&\approx h(f_1'(\mathbf{x}_{k,j})) \\
&+ \left(\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k} \bigg|_{f_1'(\mathbf{x}_{k,j})} \right) \left(\frac{\partial f_1'(\mathbf{x}_k)}{\partial \mathbf{x}_k} \bigg|_{\mathbf{x}_{k,j}} \right) \triangle \mathbf{x} \\
&+ \left(\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k} \bigg|_{f_1'(\mathbf{x}_{k,j})} \right) \mathbf{w}_{k-1}.
\end{aligned} \tag{3.20}$$

From this equation we notice that the first partial derivative term on each row is equivalent to the Jacobian matrix of the nonlinear function $h()$ evaluated at the updated point $f_1'(\mathbf{x}_{k,j})$.

This Jacobian matrix is another variation of the C matrix and is defined as

$$C_2 \triangleq \left.\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f_1'(\mathbf{x}_{k,j})}.$$

Also from this equation we notice that the second partial derivative term on the second to last row is equivalent to the Jacobian matrix of the nonlinear function $f_1'()$ evaluated at some specific point. This Jacobian matrix is similar to the previously mentioned A matrix from the description of the EKF, however, it uses the modified dynamics function we have defined in this section $f'()$ as opposed to the original dynamics function $f()$. Throughout this section we will derive multiple variations of this Jacobian matrix, and as such this specific Jacobian matrix is defined as

$$A_1 \triangleq \left.\frac{\partial f_1'(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_{k,j}}.$$

The numbered subscripts in C_2 and A_1 have similar interpretations to the C_1 matrix, and using these matrix definitions the Taylor-series expansion of the second row of Equation (3.17) is now expressed as

$$h(f'(\mathbf{x}_{k,j+1}) + \mathbf{w}_{k-1}) \approx h(f_1'(\mathbf{x}_{k,j})) + C_2 A_1 \triangle\mathbf{x} + C_2 \mathbf{w}_{k-1}. \tag{3.21}$$

From the third line of Equation (3.17), the Taylor-series expansion of the nonlinear term $h(f'(f'(\mathbf{x}_k) + \mathbf{w}_{k-1}) + \mathbf{w}_{k-2})$ is shown as

$$\begin{aligned}
&h(f'(f'(\mathbf{x}_{k,j+1}) + \mathbf{w}_{k-1}) + \mathbf{w}_{k-2}) \\
&\approx h(f'(f'(\mathbf{x}_{k,j}) + 0) + 0) \\
&+ \left(\left.\frac{\partial h(f'(f'(\mathbf{x}_k) + \mathbf{w}_{k-1}) + \mathbf{w}_{k-2})}{\partial f'(f'(\mathbf{x}_k) + \mathbf{w}_{k-1})}\right|_{\mathbf{x}_{k,j},0,0}\right)\left(\left.\frac{\partial f'(f'(\mathbf{x}_k) + \mathbf{w}_{k-1})}{\partial f'(\mathbf{x}_k)}\right|_{\mathbf{x}_{k,j},0}\right)\left(\left.\frac{\partial f'(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_{k,j}}\right)\triangle\mathbf{x} \\
&+ \left(\left.\frac{\partial h(f'(f'(\mathbf{x}_k) + \mathbf{w}_{k-1}) + \mathbf{w}_{k-2})}{\partial f'(f'(\mathbf{x}_k) + \mathbf{w}_{k-1})}\right|_{\mathbf{x}_{k,j},0,0}\right)\left(\left.\frac{\partial f'(f'(\mathbf{x}_k) + \mathbf{w}_{k-1})}{\partial \mathbf{w}_{k-1}}\right|_{\mathbf{x}_{k,j},0}\right)\mathbf{w}_{k-1} \\
&+ \left(\left.\frac{\partial h(f'(f'(\mathbf{x}_k) + \mathbf{w}_{k-1}) + \mathbf{w}_{k-2})}{\partial \mathbf{w}_{k-2}}\right|_{\mathbf{x}_{k,j},0,0}\right)\mathbf{w}_{k-2},
\end{aligned}$$

$$
\begin{aligned}
&\approx h(f'(f'(\mathbf{x}_{k,j})))\\
&+\left(\left.\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f'(f'(\mathbf{x}_{k,j}))}\right)\left(\left.\frac{\partial f'(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f'(\mathbf{x}_{k,j})}\right)\left(\left.\frac{\partial f'(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_{k,j}}\right)\triangle\mathbf{x}\\
&+\left(\left.\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f'(f'(\mathbf{x}_{k,j}))}\right)\left(\left.\frac{\partial f'(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f'(\mathbf{x}_{k,j})}\right)\mathbf{w}_{k-1}\\
&+\left(\left.\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f'(f'(\mathbf{x}_{k,j}))}\right)\mathbf{w}_{k-2},\\
&\approx h(f_2'(\mathbf{x}_{k,j}))\\
&+\left(\left.\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f_2'(\mathbf{x}_{k,j})}\right)\left(\left.\frac{\partial f_1'(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f_1'(\mathbf{x}_{k,j})}\right)\left(\left.\frac{\partial f_1'(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_{k,j}}\right)\triangle\mathbf{x}\\
&+\left(\left.\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f_2'(\mathbf{x}_{k,j})}\right)\left(\left.\frac{\partial f_1'(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f_1'(\mathbf{x}_{k,j})}\right)\mathbf{w}_{k-1}\\
&+\left(\left.\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f_2'(\mathbf{x}_{k,j})}\right)\mathbf{w}_{k-2}.
\end{aligned}
\tag{3.22}
$$

Proceeding in a similar manner as the previous two steps, we now make an additional set of C and A matrix definitions. These Jacobian matrix variations are defined as

$$
\begin{aligned}
C_3 &\triangleq \left.\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f_2'(\mathbf{x}_{k,j})},\\
A_2 &\triangleq \left.\frac{\partial f_1'(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f_1'(\mathbf{x}_{k,j})},
\end{aligned}
$$

where the numbered subscripts in C_3 and A_2 have similar interpretations to the C_1 matrix. Using these matrix definitions, the Taylor-series expansion of the third row of Equation (3.17) is now expressed as

$$
h(f'(f'(\mathbf{x}_{k,j+1})+\mathbf{w}_{k-1})+\mathbf{w}_{k-2}) \approx h(f_2'(\mathbf{x}_{k,j})) + C_3A_2A_1\triangle\mathbf{x} + C_3A_2\mathbf{w}_{k-1} + C_3\mathbf{w}_{k-2}. \tag{3.23}
$$

Continuing to evaluate the Taylor-series expansion for the nonlinear term of each subsequent row of Equation (3.17), and utilizing general definitions for all future C and A

Jacobian matrix variations as

$$C_i \triangleq \left.\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f'_{i-1}(\mathbf{x}_{k,j})} \quad \forall i \in (2, 3, \cdots, N),$$
$$A_i \triangleq \left.\frac{\partial f'_1(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{f'_{i-1}(\mathbf{x}_{k,j})} \quad \forall i \in (2, 3, \cdots, N-1),$$

and utilizing the following simplification definitions as

$$\begin{aligned} h'_1(\cdot) &\triangleq h(\cdot), \\ h'_2(\cdot) &\triangleq h(f'_1(\cdot)), \\ h'_3(\cdot) &\triangleq h(f'_2(\cdot)), \\ &\vdots \\ h'_N(\cdot) &\triangleq h(f'_{N-1}(\cdot)), \end{aligned}$$

where the numeric subscripts have a similar interpretation as the numeric subscripts on the modified Jacobian matrices defined previously, ultimately results in the following linearized form of Equation (3.17) as

$$\begin{bmatrix} \mathbf{y}_k \\ \mathbf{y}_{k-1} \\ \mathbf{y}_{k-2} \\ \mathbf{y}_{k-3} \\ \vdots \\ \mathbf{y}_{k_N} \end{bmatrix} = \begin{bmatrix} h'_1(\mathbf{x}_{k,j}) \\ h'_2(\mathbf{x}_{k,j}) \\ h'_3(\mathbf{x}_{k,j}) \\ h'_4(\mathbf{x}_{k,j}) \\ \vdots \\ h'_N(\mathbf{x}_{k,j}) \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 A_1 \\ C_3 A_2 A_1 \\ C_4 A_3 A_2 A_1 \\ \vdots \\ C_N A_{N-1} \cdots A_1 \end{bmatrix} \triangle \mathbf{x}_k$$

$$
+\begin{bmatrix}
0 & 0 & 0 & 0 & \dots & 0 \\
0 & C_2 & 0 & 0 & \dots & 0 \\
0 & C_3A_2 & C_3 & 0 & \dots & 0 \\
0 & C_4A_3A_2 & C_4A_3 & C_4 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & (C_NA_{N-1}\cdots A_4A_3A_2) & (C_NA_{N-2}\cdots A_4A_3) & (C_NA_{N-3}\cdots A_4) & \dots & C_N
\end{bmatrix}
\begin{bmatrix}
\mathbf{w}_k \\ \mathbf{w}_{k-1} \\ \mathbf{w}_{k-2} \\ \mathbf{w}_{k-3} \\ \vdots \\ \mathbf{w}_{k_N}
\end{bmatrix}
$$

$$
+\begin{bmatrix}
\mathbf{v}_k \\ \mathbf{v}_{k-1} \\ \mathbf{v}_{k-2} \\ \mathbf{v}_{k-3} \\ \vdots \\ \mathbf{v}_{k_N}
\end{bmatrix}
\tag{3.24}
$$

From the first two terms on the right side of the equation we notice that the process-noise variables $\mathbf{w}_i, \forall i = (k, k-1, \cdots, k_N)$, do not appear. From the third term of the equation, which represents the contributions due to the process noise, we see that the state variable $\mathbf{x}_{k,j+1}$ does not appear. The process-noise term is now only a function of the previous guess of the state variable $\mathbf{x}_{k,j}$ through the various Jacobian C_i and A_i matrices. This shows that the process-noise terms and the updated nonlinear state term have been completely decoupled by use of the Taylor-series expansion, which is an important result allowing linear least-squares theory to be utilized.

A few additional definitions will now be made to simplify Equation (3.24). Let us define the measurement vector as $Y_k = [\mathbf{y}_k, \dots, \mathbf{y}_{k_N}]^\top$, the nonlinear function vector evaluated at the previous guess of the state variable as $\mathcal{H}' = [h'_1(\mathbf{x}_{k,j}), \dots, h'_N(\mathbf{x}_{k,j})]^\top$, the process-noise vector $W_k = [\mathbf{w}_k, \dots, \mathbf{w}_{k_N}]^\top$, the measurement noise vector $V_k = [\mathbf{v}_k, \dots, \mathbf{v}_{k_N}]^\top$, the matrix that describes the expected measurements evolving from the ending state $\mathbf{x}_k$ backwards in

time as

$$\mathcal{O} = \begin{bmatrix} C_1 \\ C_2 A_1 \\ C_3 A_2 A_1 \\ C_4 A_3 A_2 A_1 \\ \vdots \\ C_N A_{N-1} \cdots A_1 \end{bmatrix}, \tag{3.25}$$

and the matrix which propagates the process noise backwards in time as

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & C_2 & 0 & 0 & \dots & 0 \\ 0 & C_3 A_2 & C_3 & 0 & \dots & 0 \\ 0 & C_4 A_3 A_2 & C_4 A_3 & C_4 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & (C_N A_{N-1} \cdots A_4 A_3 A_2) & (C_N A_{N-2} \cdots A_4 A_3) & (C_N A_{N-3} \cdots A_4) & \dots & C_N \end{bmatrix}. \tag{3.26}$$

Using these definitions Equation (3.24) can be simplified as

$$Y_k = \mathcal{H}' + \mathcal{O} \triangle \mathbf{x}_k + G W_k + V_k. \tag{3.27}$$

Moving $\mathcal{H}'$ to the other side of the equation, defining the difference between the measurements and the estimated measurements as $D = Y_k - \mathcal{H}'$, and the combined process and measurement noise as $\xi_k = G W_k + V_k$ results in

$$D = \mathcal{O} \triangle \mathbf{x}_k + \xi_k. \tag{3.28}$$

Since $\mathbf{w}$ and $\mathbf{v}$ are zero-mean Gaussian, then ξ_k is also zero-mean Gaussian with covariance Ξ shown as

$$\Xi = E[\xi_k \xi_k^\top] = G E[W_k W_k^\top] G^\top + E[V_k V_k^\top], \tag{3.29}$$

where the expected value of $W_k W_k^\top$ is defined as $\mathbf{Q} = E[W_k W_k^\top]$, and the expected value of $V_k V_k^\top$ is defined as $\mathbf{R} = E[V_k V_k^\top]$.

With the linearized Equation (3.24) expressed in the simplified form seen in Equation (3.28), we can finally proceed with the least-squares methods required by the Gauss-Newton method. Temporarily ignoring the process and measurement noise, the least-squares solution to Equation (3.28) is shown as

$$\triangle \mathbf{x}_k = (\mathcal{O}^\top \mathcal{O})^{-1} \mathcal{O}^\top D. \tag{3.30}$$

If the process and measurement noise are included, then the maximum likelihood estimate (MLE) of the ending state and the covariance are shown as

$$\triangle \mathbf{x}_k = (\mathcal{O}^\top \Xi^{-1} \mathcal{O})^{-1} \mathcal{O}^\top \Xi^{-1} D, \tag{3.31}$$

$$P_k = (\mathcal{O}^\top \Xi^{-1} \mathcal{O})^{-1}. \tag{3.32}$$

Having solved for the residual of the estimated ending state $\triangle \mathbf{x}_k$, we now perform a critical step in the Gauss-Newton method which is to calculate the new state estimate as

$$\mathbf{x}_{k,j+1} = \mathbf{x}_{k,j} + \triangle \mathbf{x}_k. \tag{3.33}$$

This updated state is then used to seed the next iteration of calculating the residual, and this process is repeated until the sum of the residual drops below some threshold τ_{GN}, or a predefined maximum number of iterations has been reached ℓ_{GN}.

The Gauss-Newton method, which we have used as a nonlinear regression technique, utilizes the least-squares method after appropriate linearization from the Taylor-series approximation. To incorporate the noise statistics within our solution we have used the maximum likelihood estimator within the Gauss-Newton framework. Our use of least-squares and MLE methods to solve for the desired state estimate, matches the methods originally used in Niedfeldt's implementation of the RANSAC algorithm [12]. As described previously, the RANSAC algorithm requires the use of a minimal subset of measurements that are selected randomly from a batch of data. This comes as a result of the batch of data being corrupted by gross outliers and secondary targets. Since we assume measurements may come from multiple targets, we also assume that multiple measurements can be received per measure-

ment scan. We also assume there may be time steps where measurements from the targets may be missed.

Due to these conditions, the least-squares and MLE solutions in Equations (3.30) and (3.32), respectively, will need to be modified slightly to only use the randomly selected minimum subset of measurements. Since we use the same linear methods as Niedfeldt within the Gauss-Newton method, we can utilize the solution he has developed for extracting the minimum subset of measurements. Slight differences will result, however, since we are solving for the residuals of the states as opposed to the states themselves, we are solving for the ending states as opposed to the starting states, and we use the difference vector D as opposed to the measurement vector by itself.

Modified forms of Niedfeldt's equations are now defined as

$$\Phi_k = \begin{bmatrix} \mathbf{1}_{\psi_k} \otimes I_m & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{\psi_{k-1}} \otimes I_m & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{1}_{\psi_{k_N}} \otimes I_m \end{bmatrix}, \tag{3.34}$$

where $\Phi_k \in \mathbb{R}^{m\Psi_k \times mN}$ is a matrix which is used to associate the correct time indices to all the measurements received during that time step, $\mathbf{1}_{\psi_\kappa}$ is a column vector of ψ_κ ones, $I_m \in \mathbb{R}^{m\times m}$ is an identity matrix, and $\otimes$ is the standard Kronecker product operator. Additionally, ψ_κ is the number of measurements received at the κ time step, and Ψ_k is the total number of measurements in the measurement window at the current time step. Utilizing Φ_k with the windowed measurements results in a modified form of Equation (3.27) as

$$\begin{bmatrix} \mathbf{y}_k[\psi_k] \\ \vdots \\ \mathbf{y}_k[1] \\ \mathbf{y}_{k-1}[\psi_{k-1}] \\ \vdots \\ \mathbf{y}_{k_N}[1] \end{bmatrix} = \Phi_k\mathcal{H}' + \Phi_k\mathcal{O}\triangle\mathbf{x}_k + \Phi_k G W_k + \Phi_k V_k. \tag{3.35}$$

The binary indicator matrix provides a means of selecting the minimum subset of measurements from the complete measurement window and is shown as

$$B(\mathbf{d}) = \begin{bmatrix} \mathbf{0}_{m\times m(\mathbf{d}_1-1)} & I_m & \mathbf{0}_{m\times m(\Psi_k-\mathbf{d}_1)} \\ \vdots & \vdots & \vdots \\ \mathbf{0}_{m\times m(\mathbf{d}_d-1)} & I_m & \mathbf{0}_{m\times m(\Psi_k-\mathbf{d}_d)} \end{bmatrix}, \tag{3.36}$$

where $\mathbf{d} = [\mathbf{d}_1, \ldots, \mathbf{d}_d]^\top$ is a vector that contains the randomly selected indices of the complete measurement window at the current time step. This binary indicator matrix is premultiplied to both sides of Equation (3.35) shown as

$$B(\mathbf{d})Y_k^\Phi = B(\mathbf{d})\Phi_k\mathcal{H}' + B(\mathbf{d})\Phi_k\mathcal{O}\triangle\mathbf{x}_k + B(\mathbf{d})\Phi_k\xi_k, \tag{3.37}$$

where Y_k^Φ is the windowed set of measurements gathered from our measurement device. Letting $\overline{D} = B(\mathbf{d})(Y_k^\Phi - \Phi_k\mathcal{H}')$, $\overline{\mathcal{O}} = B(\mathbf{d})\Phi_k\mathcal{O}$, and $\overline{\xi}_k = B(\mathbf{d})\Phi_k\xi_k$, results in the final modified form of Equation (3.28) as

$$\overline{D} = \overline{\mathcal{O}}\triangle\mathbf{x}_k + \overline{\xi}_k, \tag{3.38}$$

where the modified covariance matrix is shown as

$$\begin{aligned} \overline{\Xi} &= E[\overline{\xi}_k\overline{\xi}_k^\top], \\ &= B(\mathbf{d})\Phi_k E[\xi_k\xi_k^\top]\Phi_k^\top B(\mathbf{d})^\top, \\ &= B(\mathbf{d})\Phi_k\Xi\Phi_k^\top B(\mathbf{d})^\top. \end{aligned} \tag{3.39}$$

The resulting least-squares solution of the residual as compared to Equation (3.30) is now shown as

$$\triangle\mathbf{x}_k = (\overline{\mathcal{O}}^\top\overline{\mathcal{O}})^{-1}\overline{\mathcal{O}}^\top\overline{D}, \tag{3.40}$$

and the MLE of the residual of the ending states and covariance as compared to Equations (3.30) and (3.32), respectively, are now shown as

$$\triangle \mathbf{x}_k = (\overline{\mathcal{O}}^\top \overline{\Xi}^{-1} \overline{\mathcal{O}})^{-1} \overline{\mathcal{O}}^\top \overline{\Xi}^{-1} \overline{D}, \tag{3.41}$$

$$P_k = (\overline{\mathcal{O}}^\top \overline{\Xi}^{-1} \overline{\mathcal{O}})^{-1}. \tag{3.42}$$

Although the derivation of the nonlinear model hypothesis generation step using the Gauss-Newton method is complete, we now describe additional modifications made to this solution that help minimize oscillations, prevent divergence, and help the solution converge to the correct states. The first modification is to put an upper limit on the step size taken by each of the states between iterations as

$$\tau_{\triangle \mathbf{x}} = \begin{bmatrix} \tau_{\triangle \mathbf{x}_1} \\ \tau_{\triangle \mathbf{x}_2} \\ \vdots \\ \tau_{\triangle \mathbf{x}_n} \end{bmatrix}. \tag{3.43}$$

This upper limit threshold helps reduce oscillations which may also help prevent diverging solutions that are caused by extreme oscillations. The second modification has to do with the updated state estimates. After new states have been calculated from the update equation $\mathbf{x}_{k,j+1} = \mathbf{x}_{k,j} + \triangle \mathbf{x}_k$, these states may lie in a region of unacceptable values. For all states which lie outside their acceptable range of values, we apply a generic correcting function as

$$\overline{\mathbf{x}}_{k,j+1} = r(\mathbf{x}_{k,j+1}), \tag{3.44}$$

where $\overline{\mathbf{x}}_{k,j+1}$ is used to identify the modified version of the states. Solutions to the Gauss-Newton method may not necessarily be unique, however, due to the inherent nature of the states and the regions of acceptable values, only one of these solutions may be acceptable. By using the second modification we have just described, the solution resulting from the Gauss-Newton method is guaranteed to converge to a range of acceptable values, if it is to converge at all, for each of the states.

Having defined the nonlinear model hypothesis generation step, we now define the nonlinear inlier detection function $\text{Inlier}(Y_k, \mathbf{x}_k, \tau_\text{R})$ used within the RANSAC algorithm, Line 4, as

$$\chi_k = \left\{ ij \in \{1, \ldots, \Psi_k\} : \left| Y_k \left[\sum_{\forall \kappa = \{k, k-1, \ldots, k-i+2\}} \psi_\kappa + j \right] - h'_i(\mathbf{x}_k) \right| < \tau_R \right\},$$
$$\forall i = \{1, \ldots, N\}, \forall j = \{\psi_{k-i+1}, \ldots, 1\}, \tag{3.45}$$

where the error threshold parameter has been expanded in a similar way as the R-RANSAC algorithm as $\tau_\text{R} \in \mathbb{R}^m$, and the general choice of this parameter is seen to be

$$\tau_\text{R} = \begin{bmatrix} s_1 \sigma_{y_1} \\ s_2 \sigma_{y_2} \\ \vdots \\ s_m \sigma_{y_m} \end{bmatrix}. \tag{3.46}$$

From Equation (3.45) we see that the inlier function checks if the difference between each measurement in the measurement window and the predicted measurement based on the the estimated ending state propagated backwards in time, is less than the error threshold parameter. The result of this function is the set of all ij products that meet this criteria, and that represent the index location of each measurement in the measurement window that is an inlier to the estimated trajectory.

The final step of the RANSAC algorithm that needs to be defined for the nonlinear case is the smoothing step, Line 12. Similar to the nonlinear extension of R-RANSAC, the smoothing step for the RANSAC algorithm is also performed using an EKF. To use the EKF to smooth the trajectory, a couple of steps must first be taken. First, we must calculate the states at the beginning of the measurement window by propagating the estimated ending states backwards in time using Equation (3.14) as

$$\mathbf{x}_{k_N} = f'_{N-1}(\mathbf{x}_k). \tag{3.47}$$

Defining the time step of the random sample that occurred first in time is as k_{t_1}, the estimated covariance matrix must also be propagated backwards in time over the time steps from k_{t_1} to k_N using the same propagation equations shown in Equations (3.4) and (3.5). Now that the states at the beginning of the measurement window have been calculated with a corresponding covariance matrix, the final step is to propagate these forward in time to the current time step using the EKF propagation and update steps shown in Equations (3.3), through (3.9), where we use all measurements from the consensus set in the update step.

3.2.4 General Improvements Made to R-RANSAC

In the development of the ER-RANSAC algorithm shown above we have come across a few general improvements that can be used with any implementation of R-RANSAC.

The first feature we have added is a two-element good-track threshold. The two elements include the beginning and ending thresholds τ_ρ^b and τ_ρ^e, respectively, where the following condition must be true, $\tau_\rho^b > \tau_\rho^e$. The beginning threshold is used to determine if a previously bad track has gained enough inlier support to be identified as a good track. The ending threshold has the opposite effect, where it is used to determine if a previously good track has lost enough inlier support to now be identified as a bad track. The benefits of this change are two fold. First, the higher beginning threshold allows us to prevent false tracks from accidentally being deemed as a good track. Second, the lower ending threshold allows us to maintain tracks we know to be good even in a temporary reduction in measurement support, which improves track continuity.

The second improvement we have made is to expand the error threshold parameters for the inlier functions used within the RANSAC and R-RANSAC algorithms from a single number in $\mathbb{R}$ to a vector of parameters in $\mathbb{R}^m$ as shown in Equation (3.11), and (3.46). The reason for this change is that the each of the m sensor returns might have differing error statistics and they may present values with different units. By expanding the error threshold to $\mathbb{R}^m$, this allows us to set multiple error threshold values based on the error statistics and units of each of the specific sensor returns. The benefit of this change is that we can correctly identify if a measurement is an inlier to a specific model or not.

The third improvement has to do with the results of the RANSAC algorithm. In the previous implementations of the RANSAC algorithm used within the R-RANSAC framework, the iteration with the largest consensus set was used to initialize a track inside R-RANSAC, irrespective of the size of the consensus set. An optional feature that was proposed in the original RANSAC algorithm was to only return the calculated estimate if the size of the consensus set was above a certain threshold [38]. We now bring back this feature in our current implementations of RANSAC within the R-RANSAC framework using a consensus set threshold τ_{CS}. There are a couple of reasons why this feature is beneficial. First, this feature tends to reduce the number of stored models within R-RANSAC which directly reduces the amount of computation performed within R-RANSAC. A drawback of this change, however, is that if there are less stored models in R-RANSAC, then a greater number of measurements will be outliers to existing models which means the RANSAC algorithm will have to be run a greater number of times. In spite of this drawback, another benefit of this feature is that it helps to improve the the quality of tracks being initialized by RANSAC. In the ideal case, the RANSAC algorithm would be performed after the measurement window has been completely filled with measurements from the target we are trying to track. This will increase the likelihood that the RANSAC algorithm converges to an estimate that is close to the true states of the target because it has more measurements to draw from. If the RANSAC algorithm is executed with only a few measurements from the target, then it will likely result in an estimate of the state that is not close to the true state of the target. By only accepting results from the RANSAC algorithm when large consensus sets are obtained, we are effectively guaranteeing that we have performed RANSAC with a measurement window that is largely filled with measurements from the target, and that results in the formation of more accurate state estimates used to initialize tracks.

The final improvement was developed to help optimize the R-RANSAC algorithm when implemented in code to reduce the computational overhead caused by numerous matrix multiplications. In the RANSAC algorithm a minimum subset of measurements from the measurement window are selected at random and are used to obtain a least-squares or MLE solution. To make the RANSAC algorithm work within the framework of R-RANSAC, one of these measurements is required to be a measurement taken from the current time step $\mathbf{y}_k$.

Since one of the measurements will always originate from the current time step, it would be ideal to limit the amount of computational overhead associated with this measurement. From the original linear implementation of RANSAC, we see that this measurement always appears on the last row of the measurement equation, which is shown here for convenience as

$$\begin{bmatrix} \mathbf{y}_{k_N} \\ \mathbf{y}_{k_N+1} \\ \vdots \\ \mathbf{y}_k \end{bmatrix} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{bmatrix} \mathbf{x}_{k_N} + \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & C_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & CA^{N-2} & CA^{N-3} & \dots & C \end{bmatrix} \begin{bmatrix} \mathbf{w}_{k_N} \\ \mathbf{w}_{k_N+1} \\ \vdots \\ \mathbf{w}_k \end{bmatrix} + \begin{bmatrix} \mathbf{v}_{k_N} \\ \mathbf{v}_{k_N+1} \\ \vdots \\ \mathbf{v}_k \end{bmatrix}.$$

The reason the measurement from the current time step appears on the last row is because the states that are being estimated are the states at the beginning of the measurement window, $\mathbf{x}_{k_N}$. The time difference between this estimated state and the current measurement, therefore, spans the entire measurement window requiring numerous matrix multiplications within the $\mathcal{O}$ and G matrices to correctly propagate the initial states into the same time step as the current measurement.

The solution we have developed to eliminate these numerous matrix multiplications is to instead solve for the ending states, $\mathbf{x}_k$, which corresponds to the same time step as the most recently received measurement. Because of this change in states being estimated, the derivation of the above equation must proceed in a slightly different manner. Instead of developing these equations using propagation equations that evolve forward in time, we must use propagation equations that evolve backwards in time. This is what we have done in our derivation of the nonlinear version of RANSAC as seen in Equation (3.24). From Equation (3.24) we now see that the first row of the equation contains the measurement from the current time step, and in this row we also see the $\mathcal{O}$ matrix contains a single matrix C_1 with no matrix multiplications, and the G matrix contains purely zero values with no matrices or matrix multiplications. Due to the large reduction in the number of matrix multiplications, the computational cost is significantly reduced.

3.3 Implementation of ER-RANSAC Using On-board Radar

In this section we present simulation results that demonstrate the ability to successfully track multiple maneuvering intruder aircraft located in all directions around a maneuvering ownship in the presence of spurious measurements/clutter, missed measurements, and noisy measurements. We do so using an air-based radar fixed to the ownship as the detection device and the newly developed ER-RANSAC as the MTT algorithm. The remainder of this section proceeds as follows: first we describe the dynamic model used in the simulation, then we describe the radar model, next we show the observability of our system, then we present results demonstrating the performance of the Gauss-Newton method, and finally we provide plots of the simulation results.

3.3.1 Nonlinear Constant Acceleration Model

As described in Section 3.2.1, to track maneuvering intruder aircraft we use a constant-acceleration model. Specifically we use a nonlinear dynamic model based on states which include north position p_n, east position p_e, altitude h, ground speed V_g, course angle χ, flight path angle γ, ground-speed rate $\dot{V}_g$, course angle rate $\dot{\chi}$, and flight path angle rate $\dot{\gamma}$. The state vector is defined as $\mathbf{x} = [p_n, p_e, h, V_g, \chi, \gamma, \dot{V}_g, \dot{\chi}, \dot{\gamma}]^\top$, and the nonlinear constant-acceleration sate equation is shown as

$$f(\mathbf{x}) = \begin{pmatrix} \dot{p_n} \\ \dot{p_e} \\ \dot{h} \\ \dot{V}_g \\ \dot{\chi} \\ \dot{\gamma} \\ \ddot{V}_g \\ \ddot{\chi} \\ \ddot{\gamma} \end{pmatrix} = \begin{pmatrix} V_g \cos\chi \cos\gamma \\ V_g \sin\chi \cos\gamma \\ V_g \sin\gamma \\ \dot{V}_g \\ \dot{\chi} \\ \dot{\gamma} \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.48}$$

This dynamic model is able to capture constant-acceleration maneuvers performed by the intruders including accelerating/decelerating at a constant rate, turning at a constant turn rate, and increasing/decreasing the flight path angle at a constant rate.

3.3.2 On-board Radar Sensors

As described in Section 3.2.1, to detect overtaking intruder aircraft we need a detection device that can detect intruders behind the ownship. The solution we presented is seen in Figure 3.1 which shows three radar units mounted in three directions around the ownship. One is pointed forward and the other two are pointed with offset angles in the reverse direction. This setup allows us to get full 360 degree coverage in the azimuth direction.

To develop a set of output equations for the radar sensors, we first need to define a set of ownship states including the ownship's position and attitude. These ownship states are specifically the north position, east position, altitude, roll, pitch, and yaw, and will be represented as

$$\mathbf{x}_o = [p_{n,o}, p_{e,o}, h_o, \phi_o, \theta_o, \psi_o]^\top, \tag{3.49}$$

where the subscript indicates these states are from the ownship. This subscript has only been added to the ownship states and not the intruder states.

Having defined the set of ownship and intruder states, we are now prepared to derive the output equations resulting from the radar. Each radar unit produces a set of three measurements including the range to the target r, the azimuth angle to the target measured relative to the forward direction of the aircraft in the body frame α, and the elevation angle to the target measured relative to the body-fixed horizontal plane ϵ. The output equation resulting from these three measurements is shown as

$$h(\mathbf{x}, \mathbf{x}_o) = \begin{pmatrix} r(\mathbf{x}, \mathbf{x}_o) \\ \alpha(\mathbf{x}, \mathbf{x}_o) \\ \epsilon(\mathbf{x}, \mathbf{x}_o) \end{pmatrix} = \begin{pmatrix} \sqrt{(p_n^v)^2 + (p_e^v)^2 + (h^v)^2} \\ \tan^{-1}\left(\frac{p_e^b}{p_n^b}\right) \\ \tan^{-1}\left(\frac{h^b}{\sqrt{(p_n^b)^2 + (p_e^b)^2}}\right) \end{pmatrix}, \tag{3.50}$$

where the superscript v indicates the state is expressed in the vehicle frame of the ownship, and the superscript b indicates the state is expressed in the body frame of the ownship. The vehicle frame of the ownship is defined as an inertial north-east-down reference frame centered about the ownship. The body from of the ownship is defined as a right handed coordinate system centered about the ownship with the x-axis directed out the front of the aircraft, the y-axis directed out the right wing of the aircraft, and the z-axis directed out the bottom of the aircraft.

Using the position states of the ownship and intruders, a general right handed north-east-down position vector is defined as $\mathbf{p} = [p_n, p_e, -h]^\top$. From this definition the position of the intruder in the vehicle frame of the ownship is expressed as

$$\mathbf{p}^v = \begin{pmatrix} p_n^v \\ p_e^v \\ -h^v \end{pmatrix} = \begin{pmatrix} p_n - p_{n,o} \\ p_e - p_{e,o} \\ -(h - h_o) \end{pmatrix}, \tag{3.51}$$

where this position vector will be used in the definition of the range equation.

For the azimuth and elevation angle equations, expressions must now be developed for the position vector of the intruder in the body frame of the ownship. Using the three element rotation matrix which transforms a vector from the vehicle to the body frame as

$$R_v^b(\phi, \theta, \psi) = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{bmatrix}, \tag{3.52}$$

we can now express the the position vector of the intruder in the body frame of the ownship as

$$\mathbf{p}^b = \begin{pmatrix} p_n^b \\ p_e^b \\ -h^b \end{pmatrix} = R_v^b(\phi, \theta, \psi)\mathbf{p}^v. \tag{3.53}$$

Multiplying out this product results in expressions for the positions of the intruders in the body frame of the ownship as

$$p_n^b = (p_n - p_{n,o})c_\theta c_\psi + (p_e - p_{e,o})c_\theta s_\psi - (h - h_o)(-s_\theta), \tag{3.54}$$

$$p_e^b = (p_n - p_{n,o})(s_\phi s_\theta c_\psi - c_\phi s_\psi) + (p_e - p_{e,o})(s_\phi s_\theta s_\psi + c_\phi c_\psi) - (h - h_o)s_\phi c_\theta, \tag{3.55}$$

$$h^b = -(p_n - p_{n,o})(c_\phi s_\theta c_\psi + s_\phi s_\psi) - (p_e - p_{e,o})(c_\phi s_\theta s_\psi - s_\phi c_\psi) + (h - h_o)c_\phi c_\theta. \tag{3.56}$$

Having defined the position vectors of the intruder expressed in both the vehicle frame as seen in Equation (3.51) and the body frame as seen in Equations (3.54), (3.55), and (3.56), the final step is to substitute these back into Equation (3.50) upon which we have completed our derivation of the output equations for range, azimuth and elevation.

3.3.3 Observability of Nonlinear System

Before proceeding to the results section, we first demonstrate that the states resulting from the nonlinear state-space equations are indeed observable. To determine if the system is observable we need to produce an observability matrix. The system is observable if the rank of the observability matrix is equal to n, the number of states. Another way to show when the system is observable is to show when the determinant of the observability matrix is not equal to zero. The general nonlinear observability matrix is shown as

$$\mathcal{O} = \frac{\partial}{\partial x}\begin{pmatrix} h(x) \\ L_f h(x) \\ L_f^2 h(x) \\ \vdots \end{pmatrix} = \frac{\partial}{\partial x}\begin{pmatrix} h_1(x) \\ \vdots \\ h_m(x) \\ L_f h_1(x) \\ \vdots \\ L_f h_m(x) \\ L_f^2 h_1(x) \\ \vdots \\ L_f^2 h_m(x) \\ \vdots \end{pmatrix} = \frac{\partial}{\partial x}\begin{pmatrix} h_1(x) \\ \vdots \\ h_m(x) \\ \frac{\partial h_1}{\partial x} f(x) \\ \vdots \\ \frac{\partial h_m}{\partial x} f(x) \\ \frac{\partial (L_f h_1(x))}{\partial x} f(x) \\ \vdots \\ \frac{\partial (L_f h_m(x))}{\partial x} f(x) \\ \vdots \end{pmatrix}, \tag{3.57}$$

where terms are appended to the bottom of the matrix until $rank(\mathcal{O}) = n$. If we keep appending terms to the bottom of the observability matrix and the rank never becomes equal to n, then the system is not observable.

The range, azimuth, and elevation equations defined previously are large and cumbersome, which results in a large observability matrix and an even larger determinant. The determinant of the observability matrix using these equations results in hundreds of terms which become too large to manipulate or interpret. To determine the observability of the system, an assumption is needed to simplify the derivation to produce meaningful results. The assumption we make is that the ownship's attitude is $\phi = \theta = \psi = 0$. This results in the previously defined rotation matrix being equal to identity and the resulting output equations are then expressed in a simplified form as

$$r(\mathbf{x}, \mathbf{x}_o) = \sqrt{(p_n - p_{n,o})^2 + (p_e - p_{e,o})^2 + (h - h_o)^2}, \tag{3.58}$$

$$\alpha(\mathbf{x}, \mathbf{x}_o) = \tan^{-1}\left(\frac{p_e - p_{e,o}}{p_n - p_{n,o}}\right), \tag{3.59}$$

$$\epsilon(\mathbf{x}, \mathbf{x}_o) = \tan^{-1}\left(\frac{h - h_o}{\sqrt{(p_n - p_{n,o})^2 + (p_e - p_{e,o})^2}}\right). \tag{3.60}$$

Using this simplified set of output equations we show that the states are observable for this specific attitude of the ownship. The specific details proving our specific states are observable can be found in Appendix B, and from this derivation we determine that three sets of measurements are required to make the states observable. After showing that the states are observable for this specific attitude of the ownship we then make the assumption that the states of the intruder remain observable regardless of the attitude of the ownship. This assumption appears to be correlated to a result found in Ref. [40]. From derivations in this reference, it was found that the evolution of the states of a goal node were independent of the attitude of a ground robot.

3.3.4 Gauss-Newton Results

In this section we demonstrate the performance of the Gauss-Newton method in estimating the ending states of an aircraft from a minimum subset of measurements. For

this demonstration we show the results from a single call to the Gauss-Newton function. In these results we are using the nine-state nonlinear constant-acceleration model, and the on-board radar sensor model which provides nonlinear measurements. From the observability section of this state space model, found that three sets of range, azimuth, and elevation angle measurements are needed for the minimum subset. Using this setup we provide an initial guess for the states as $\mathbf{x}_{k,j} = [0, 0, 200, 15, 0, 0, 0, 0, 0]^\top$. The evolution of these states across each iteration of the Gauss-Newton method are shown in Figure 3.3.

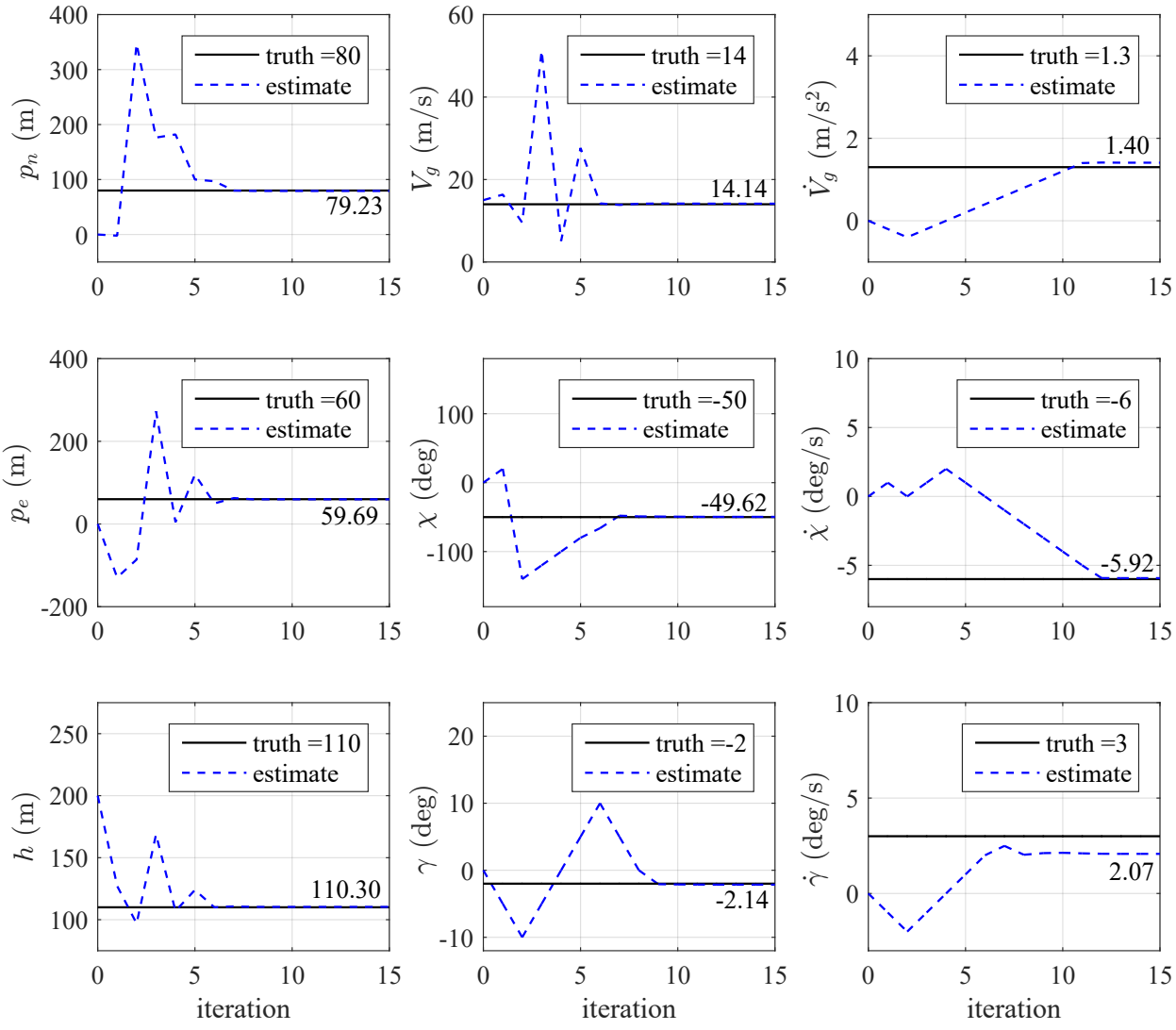


Figure 3.3: Evolution of states across each iteration of the Gauss-Newton method.

In this example we have defined the true states of the aircraft at the current time step to be equal to $\mathbf{x}_k = [80, 60, 110, 14, -50, -2, 1.3, -6, 3]^\top$ as seen by the black lines in Figure 3.3. Using these states for the current time step, we solve the nonlinear differential equation backwards in time for 50 time steps and create noisy radar measurements based on these propagated states. Using a minimum subset of three radar measurements at three different time steps, we begin executing the Gauss-Newton method. As seen in Figure 3.3, this particular call to the Gauss-Newton method took 15 iterations to converge to a solution. Since we used noisy radar measurements, the state estimates that it converges to are not exactly equal to the true states, however, we see that the final estimates accurately represent the true states. The value of the states from the final iteration of the Gauss-Newton method are also shown on each of the plots next to the final state estimate. Although the resulting solution is not exactly equal to the true states, the solution becomes more accurate as the noise on the radar measurements decreases.

Referring back to the illustration we presented of the nonlinear version of RANSAC in Figure 3.2, we remember that in the fourth frame we used to the Gauss-Newton method to iteratively solve for a model hypothesis $\hat{\mathbf{x}}'_k$. From Figure 3.3, we see that the model hypothesis for this particular run of the Gauss-Newton method is equal to

$$\hat{\mathbf{x}}'_k = \begin{pmatrix} 79.23 \text{ m} \\ 59.69 \text{ m} \\ 110.30 \text{ m} \\ 14.14 \text{ m/s} \\ -49.62 \text{ deg} \\ -2.14 \text{ deg} \\ 1.40 \text{ m/s}^2 \\ -5.92 \text{ deg/s} \\ 2.07 \text{ deg/s} \end{pmatrix}. \tag{3.61}$$

Also Referring back to the illustration in Figure 3.2, we remember that in the fifth frame we evaluated the nonlinear dynamic equation backwards in time using the the model hypothesis states as the starting point. As we propagated these states backwards in time we

ended up with a trajectory that was used for inlier detection needed to find the consensus set. Up to this point we have shown that the Gauss-Newton method is successfully able to create a model hypothesis of the states at the final time step, as $\hat{\mathbf{x}}'_k$. For the last part of the results in this section, we show that that the process of solving the nonlinear dynamic equation backwards in time does indeed create a trajectory that passes through each of the three minimum subset radar measurements, thus demonstrating that the states generated from the Gauss-Newton method will indeed be effective in creating a trajectory for inlier detection. The original Gauss-Newton model hypothesis and the resulting trajectory are shown in Figures 3.4 and 3.5

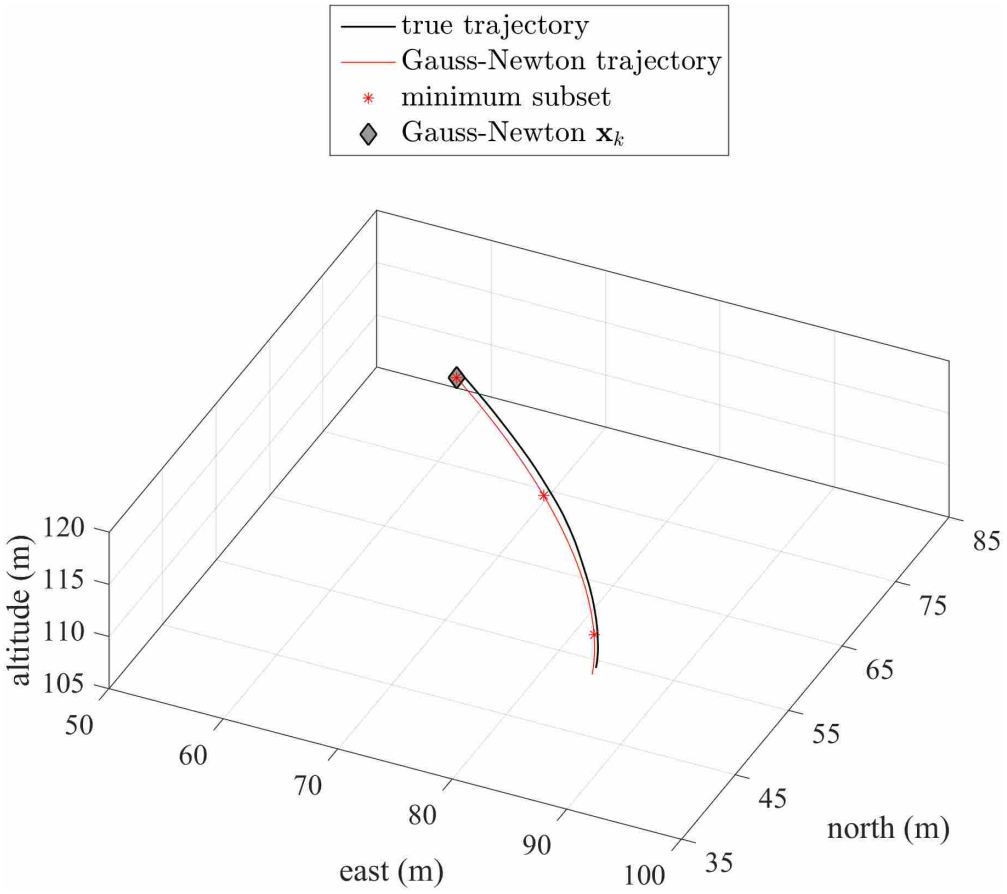
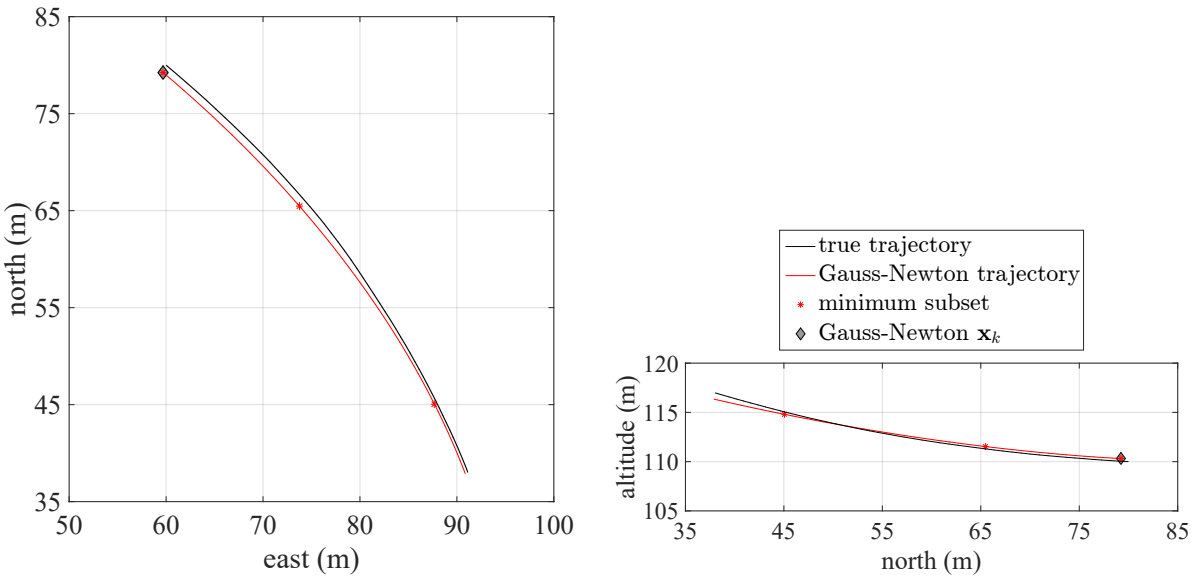


Figure 3.4: The true trajectory, minimum subset of radar measurements, and resulting Gauss-Newton trajectory.



(a) Top view.

(b) Side view.

Figure 3.5: The true trajectory and Gauss-Newton trajectory shown in 2D.

In Figures 3.4 and 3.5, we have also plotted the true trajectory of the aircraft and the three radar measurements used as the minimum subset. The two random measurements occurred 1.5 and 3.9 seconds in the past compared to the current time step. From these figures we see that the estimated trajectory passes directly through each of the three radar measurements, and that because the three radar measurements do not have large noise properties, the resulting trajectory follows very closely to the true trajectory.

These results complete the analysis of the Gauss-Newton method, showing that it is indeed a valid method for generating a model hypothesis. The results of this Gauss-Newton method would then need to be integrated into the remaining steps of the nonlinear RANSAC algorithm and subsequently into the ER-RANSAC algorithm, which is what we do in the next section by integrating all the pieces of ER-RANSAC together to track multiple dynamic targets.

3.3.5 Simulation Results

To present the tracking results for this section we have utilized the improved simulation environment described in Appendix C which has been developed in Matlab/Simulink.

For our simulation each of the aircraft are modeled using a full 12-state dynamic model controlled by an autopilot similar in structure to one described in Ref. [19]. The specific flight paths taken by each of the aircraft are shown in Figures 3.6 and 3.7.

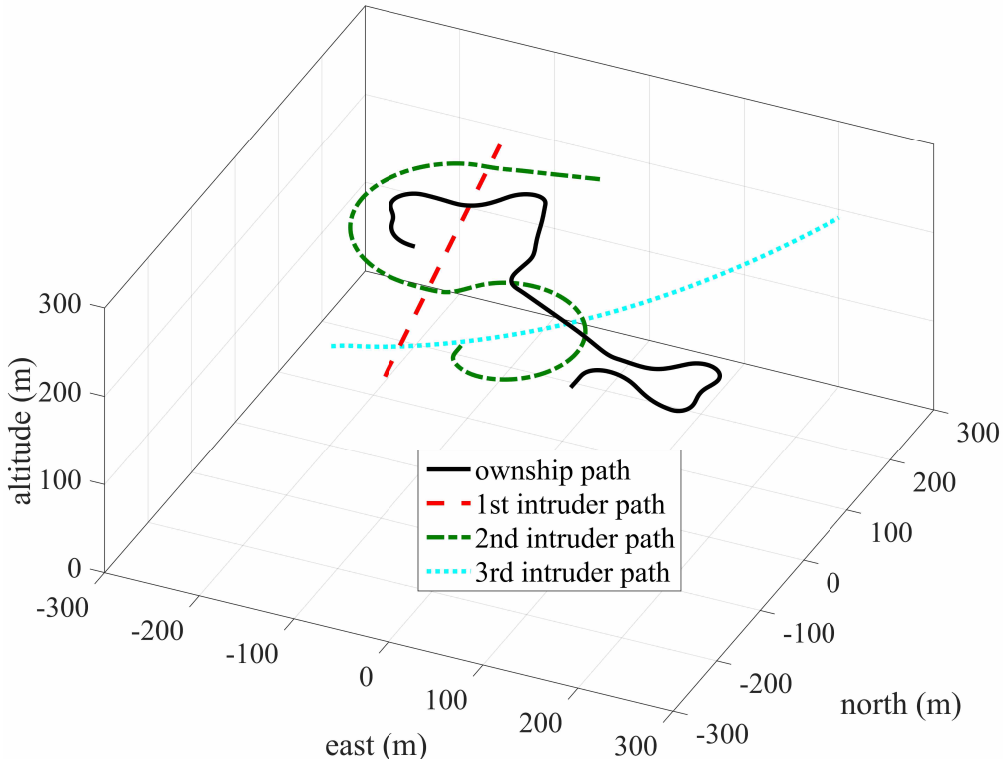
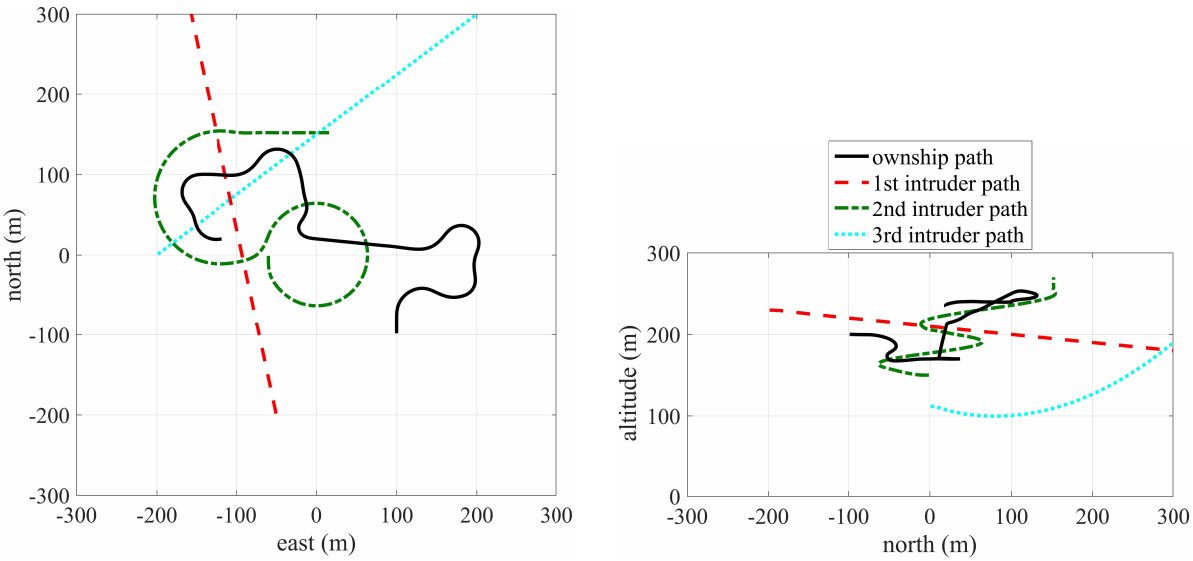


Figure 3.6: Flight paths of the ownship and three intruder aircraft.

In our previous discussion of desired tracker properties we stated that the tracker needed to be able to track multiple maneuvering intruders. As such we have included three intruder aircraft with varying flight-path maneuvers as seen in Figures 3.6 and 3.7. For the first intruder we have defined a flight path with a constant velocity, heading and flight-path angle. The constant flight-path angle results in the aircraft descending in altitude instead of a constant-altitude flight path. For the second intruder, we have defined an upwards spiraling flight path which changes course-rate direction half way through the simulation. This aircraft is also commanded to accelerate at a constant value followed by a command to decelerate at a constant value. This results in an aircraft with a constant acceleration, constant turn rate, and constant flight-path angle. Finally, for the third intruder we have

defined a flight path with a constant velocity, constant heading, and constant flight-path angle rate. This intruder is initialized with a negative flight-path angle, which results in the aircraft first descending, then leveling off, then climbing with an increasing flight-path angle.



(a) Top view.

(b) Side view.

Figure 3.7: Flight paths shown in 2D.

The ownship which is used in this simulation is also modeled to include an on-board radar sensor. Specifically, the ownship is carrying three radar units oriented in the three directions described in previous sections. Since these three radar units are fixed to the body of the ownship, their orientation changes as the attitude of the ownship is altered. The range measurements received from the radar units are not affected by the orientation of the radar, however, the azimuth and elevation angle measurements are affected by the orientation. In addition to the attitude, the position of the ownship also has an affect on the received radar measurements. The position, however, affects the received range measurements in addition to the azimuth and elevation measurements. The specific effects of the ownship position and attitude on the radar measurements were derived previously, shown by Equations (3.50), (3.51), (3.54), (3.55), and (3.56). These output equations are the output equations used within the simulated radar model, and the EKF update and Gauss-Newton steps of the ER-RANSAC algorithm. To demonstrate that the ER-RANSAC tracking algorithm correctly

deals with these variations in radar measurements, we have given the ownship a highly maneuverable flight path, also seen in Figures 3.6 and 3.7.

The radar model used in this simulation calculates the true range, azimuth and elevation measurements based on the true states of the ownship and intruders. Starting with this complete list of radar measurements, we then simulate missed measurements by randomly removing measurements based on a specified probability of detection. For the results in this section we have used a probability of detection of 0.8. Normally distributed random noise is then added to each of the sensor measurements based on their unique standard deviations. In addition to missed measurements and noisy measurements, we have also added clutter measurements to the simulation. For the clutter we have added a random number of measurements based on a uniform distribution of integers from zero to three. The final step in this radar model was to assign each of these radar measurements to the correct radar unit based on which azimuth direction it was pointing. The simulated radar units also had a limited field of view (FOV) in elevation, so any measurements outside this elevation threshold were eliminated from the output.

Using the three different intruder aircraft trajectories and the maneuvering ownship trajectory, the simulated radar measurements are transformed into their respective north-east-down components and are plotted with the actual flight paths flown by the aircraft as seen in Figures 3.8 and 3.9.

Another desired tracker property includes the ability to track intruders which may be overtaking the ownship from behind. The three radar units described above are mounted on the ownship in such a way as to provide 360-degree coverage around the ownship in the azimuth direction. The maneuvering flight paths of the ownship and intruders described above were also designed to ensure that the intruder aircraft are located behind the ownship at various times throughout the simulation. Since the three radar units are oriented at three different azimuth angles compared to the ownship's forward direction, we have applied an offset to the azimuth returns from the two backwards pointing radar units. This allows us to use a single measurement equation within the ER-RANSAC tracking algorithm. This also results in the ability to plot the radar returns from all three radar on the same plot as seen in Figure 3.10.

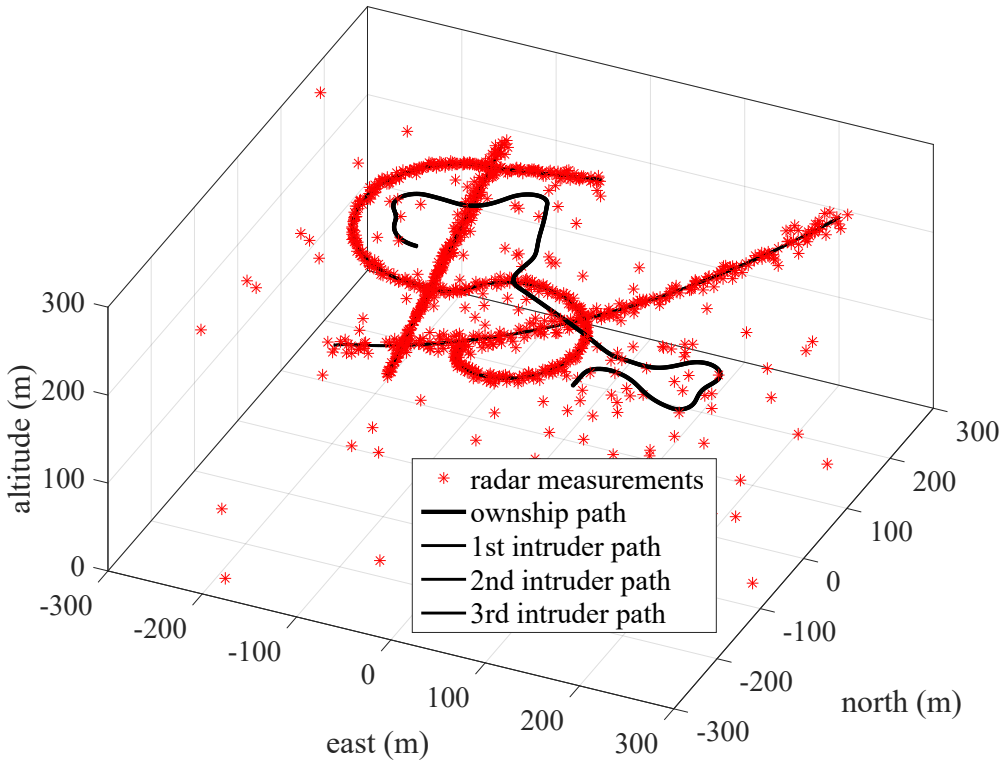
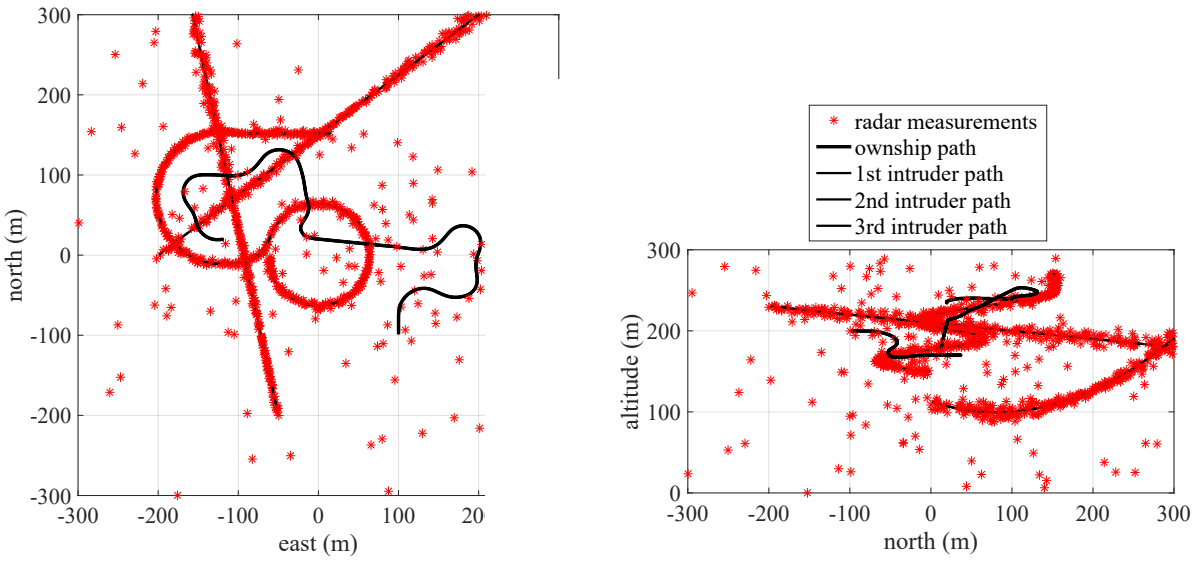


Figure 3.8: Flight paths and radar measurements shown in the inertial reference frame.



(a) Top view.

(b) Side view.

Figure 3.9: Flight paths with radar measurements shown in 2D.

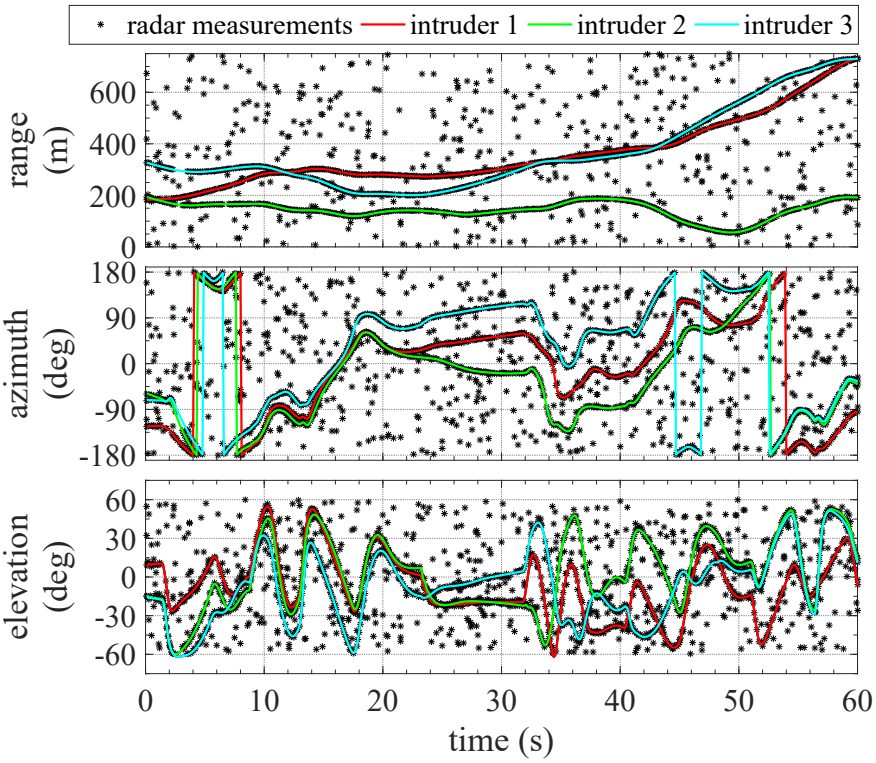


Figure 3.10: Radar measurements in ownship body frame.

Observing the azimuth measurements in Figure 3.10 we can distinguish between the radar measurements received from each of the three radar units. The first radar unit is mounted in the forward pointing direction on the ownship. Since the field of view (FOV) of each radar is 120 degrees in azimuth, all radar measurements with an azimuth angle between -60 and 60 degrees originate from the first radar unit. The second radar is mounted backwards and to the right at an azimuth angle of 120 degrees. All azimuth angle measurements between 60 and 180 degrees, therefore, originate from the second radar unit. Finally, the third radar unit is mounted backwards and to the left at an azimuth angle of -120 degrees. This results in all azimuth measurements between -60 and -180 degrees.

Also from Figure 3.10 we can see the effects of the limited elevation FOV. The elevation FOV used in this simulation spans from -60 to 60 degrees, and as a result we notice that there are no radar measurements below -60 degrees or above 60 degrees. Upon careful inspection we also see that 2nd and 3rd intruders briefly drop below the -60 degree elevation FOV boundary at approximately 3 seconds into the simulation. The 2nd intruder leaves the

FOV for about 0.5 seconds, while the 3rd intruder leaves the FOV for about 1 second. By leaving the elevation FOV of the radar, we temporarily lose all three range, azimuth, and elevation measurements from these aircraft which is most noticeable in the range plot.

Having shown the simulation will properly test each of the desired properties of the tracker, we now move to the results of the estimated intruder tracks resulting from the ER-RANSAC tracking algorithm. For the results in this simulation have used a sample rate of 0.1 seconds, R-RANSAC parameters shown in Table 3.3, RANSAC parameters shown in Table 3.4, and Gauss-Newton parameters shown in Table 3.5.

Table 3.3: R-RANSAC parameters.

Parameter	Value
N	150
$\mathcal{M}$	10
τ_{RR}	$(s_r\sigma_r, s_\alpha\sigma_\alpha, s_\epsilon\sigma_\epsilon)$
s_r	17.5
s_α	3.5
s_ϵ	3.5
σ_r	0.4 m
σ_α	1 deg
σ_ϵ	1 deg
τ_ρ^b	0.4
τ_ρ^e	0.3
τ_{CS}	0.3
τ_{T}	20
MD	15
Q	(0.0001 , 0.0001, 0.01, 0.01, 0.01, 0.0005, 0.0009, 0.001, 0.00001)
R	$(\sigma_r^2, \sigma_\alpha^2, \sigma_\epsilon^2)$
τ_{x_i}	(10 m, 10 m, 10 m, 3 m/s, 0.5 deg, 0.5 deg, 1 m/s^2, and 0.3 deg/s, 0.3 deg/s)

Table 3.4: RANSAC parameters.

Parameter	Value	Parameter	Value
ℓ	10	τ_{R}	$(s_r\sigma_r, s_\alpha\sigma_\alpha, s_\epsilon\sigma_\epsilon)$
γ	$0.8N$	s_r, s_α, s_ϵ	2.5

Table 3.5: Gauss-Newton parameters.

Parameter	Value	Parameter	Value	Parameter	Value
$\tau_{\triangle p_n}$	1000 m	$\tau_{\triangle V_g}$	50 m/s	$\tau_{\triangle \dot{V}_g}$	0.2 m/s^2
$\tau_{\triangle p_e}$	1000 m	$\tau_{\triangle \chi}$	20 deg	$\tau_{\triangle \dot{\chi}}$	1 deg/s
$\tau_{\triangle h}$	1000 m	$\tau_{\triangle \gamma}$	5 deg	$\tau_{\triangle \dot{\gamma}}$	1 deg/s
ℓ_{GN}	30	τ_{GN}	1e-10		

For the state correcting function used within the Gauss Newton method we perform the following checks. For the estimate of V_g we required that the solution be greater than zero. If $V_g < 0$ m/s, we perform the following functions $\overline{V}_g = |V_g|$, and $\overline{\chi} = \chi + \pi$. For the estimate of χ we require that the solution lie between -180 and 180 degrees. If $\chi < -180$ deg, we add 2π until the solution is greater than -180 degrees. If $\chi > 180$ deg, we subtract 2π until the solution is less than 180 degrees. Finally, for the estimate of γ we require that the solution lie within -90 and 90 degrees. If $\gamma > 90$ deg, we perform the following function $\overline{\gamma} = 89.999$ deg. If $\gamma < -90$ deg, we perform the following function $\overline{\gamma} = -89.999$ deg. We do not set the flight-path angle exactly equal to 90 or -90 degrees because, as we have shown in the nonlinear observability derivation in Appendix B, this will make the course angle state unobservable.

Using these tracking parameters we plot the estimated states as a function of time as seen in Figures 3.11, 3.12, and 3.13. In each of these figures we compare the estimated states with the true states of the intruders.

In Figure 3.11 we have plotted the position states of the aircraft which include the position north, position east, and altitude. In addition to the true and estimated position states, we have also plotted the transformed radar measurements. From this figure we see

that the position estimates of ER-RANSAC follow very closely with their respective radar positions. Since there was not any bias in the radar measurements, this means the position estim closely.

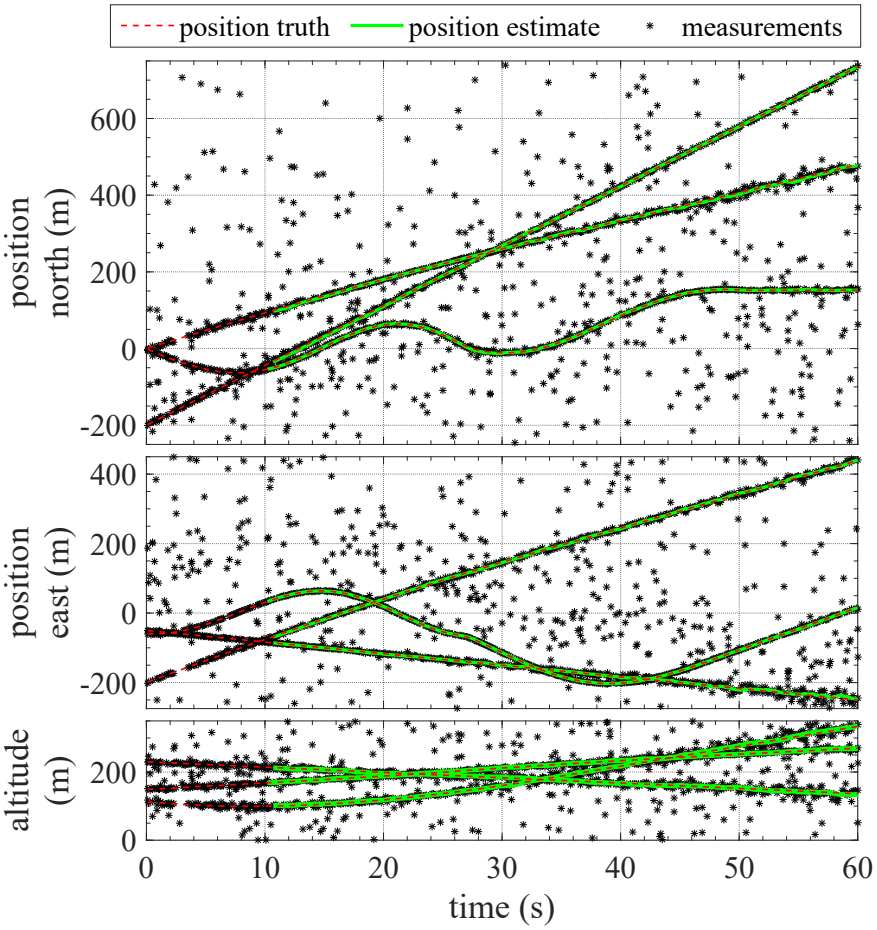


Figure 3.11: ER-RANSAC position estimates.

Also from this figure we see that we did not receive any tracks of the targets until approximately 10 to 11 seconds into the simulation. Since we used a step size of 0.1 seconds and a measurement window size of 150, the measurement window will not be completely full until 15 seconds into the simulation. Using the fact that we had a consensus set threshold within RANSAC of 0.3, and that we had a probability of detection of 0.8, the earliest a track could have been created within this simulation would be 5.6 seconds into the simulation. After a track has been initialized it must exist for a certain amount of time before it is counted as a good track. Since we used a minimum track threshold time of 20, then the earliest a track could have been deemed a good model would be 2 seconds after it has been

created, or 7.6 seconds into the simulation. The fact that our tracks were not actually deemed as good tracks until roughly 10 to 11 seconds, or about 2.4 to 3.4 seconds after they could have been deemed as good tracks, can be explained by a couple of factors. For our minimum subset used within RANSAC, we randomly selected two measurements in addition to the measurement at the current time step. The likelihood that both of these measurements originate from a single intruder, and that both of these measurements are spaced far enough apart, is low. Also, for the RANSAC algorithm we only iterate a maximum of 10 times. Although our tracks were not deemed good tracks until roughly 2.4 to 3.4 seconds after they could have been, they were deemed good tracks roughly 4 to 5 seconds before the measurement window was completely filled.

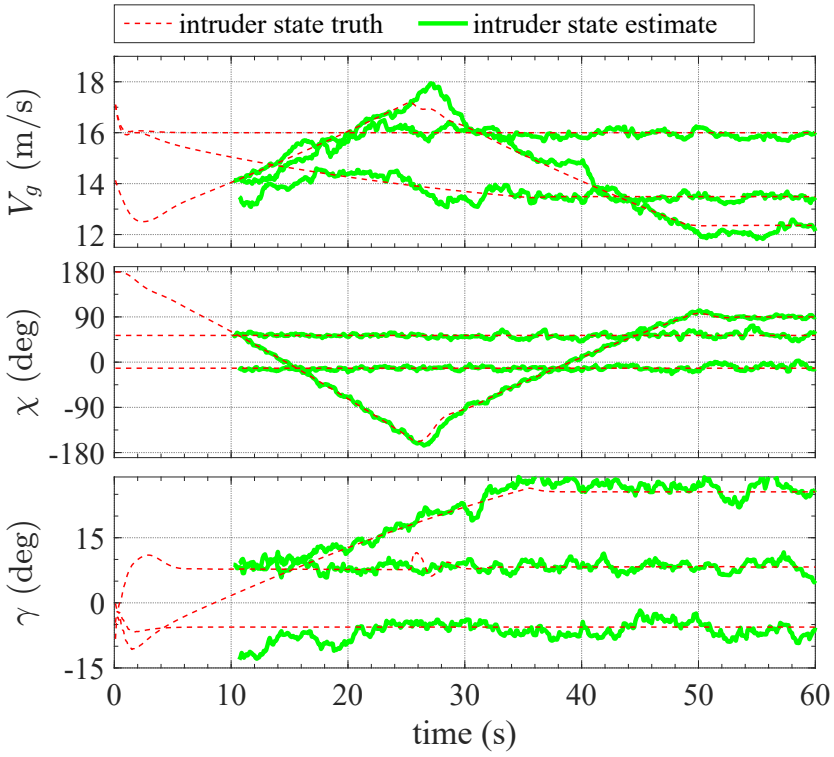


Figure 3.12: ER-RANSAC velocity estimates.

In Figure 3.12 we have plotted the states that relate to the velocity of the aircraft which include the ground speed, course, and flight-path angle. Although the course and flight-path angle are not measurements of speed, they do provide a direction for the velocity

of the aircraft. The estimated states seen in this figure are slightly more noisy than the position estimates seen earlier, however, they still track the true states relatively well.

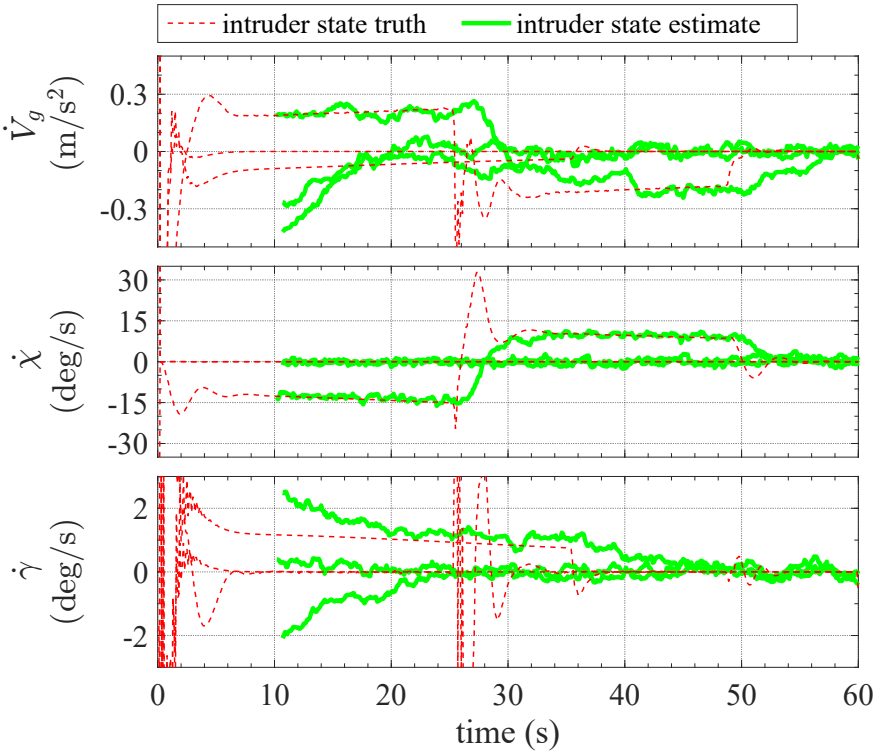


Figure 3.13: ER-RANSAC acceleration estimates.

In Figure 3.13 we have plotted the states that relate to the acceleration of the aircraft which include the ground-speed rate, course rate, and flight-path angle rate. The first observation to make from this figure is that we have successfully created maneuvers for each of the intruders which include constant accelerations, constant turning rates, and constant flight-path angle rates. From the estimates of these states we see that ER-RANSAC was able to successfully lock onto each of these states resulting from a highly maneuverable aircraft. The second observation to make is to note how well the Gauss-Newton method did in initializing each of the states resulting from a nonlinear dynamic model. This same observation can also be seen in Figures 3.11 and 3.12, where we see that the initial estimates of each of the states were close to the true states of the intruder aircraft.

CHAPTER 4. COMPLETE DAA SYSTEM USING GBR

A detect-and-avoid system has many subcomponents that must all be working seamlessly together to create a fully operational DAA system. These subcomponents include a detection device, processor platform, and DAA algorithms including target detection and tracking, collision detection, and collision avoidance path planning. At Brigham Young University, research is actively being done to create such a system. The ultimate goal is to create a complete DAA system that is fully autonomous and which uses an air-based detection device.

Initial efforts by Klaus have been performed to create such a system [30]. In his research, Klaus was able to integrate all the pieces of a DAA system together in a system that included both hardware and simulated components. He was then able to extend this system using the state information from small UAS hardware, including two fixed-wing autopilot-controlled aircraft, as opposed to simulated aircraft. The state information from these unmanned aircraft were fed into a program that generated simulated radar measurements and performed the remaining DAA algorithms. New flight paths obtained from these DAA algorithms were then sent to the ownship aircraft that performed the new flight path in real time. The simulated radar and DAA algorithms were run on a ground station PC, and the states were obtained and the waypoints were sent through a wireless communications link to the aircraft.

Currently, additional research is being done to create a complete DAA system using actual radar hardware in place of a simulated radar. A team of students under the direction of Dr. Karl Warnick and Dr. Doran Wilde are attempting to create a radar unit that will ultimately be mounted on-board a small UAS, and which therefore must meet the size weight and power (SWaP) requirements of small UAS. An additional team of students under the direction of Dr Tim McLain and Dr Randy Beard are working to create the target detection

and tracking, collision detection, and collision avoidance algorithms that will be used in conjunction with the radar hardware being developed. Results from these current efforts are the primary contributions shown in this chapter.

To achieve the ultimate goal of using an air-based detection device in the DAA solution, we have first chosen to utilize the radar hardware in a ground-based radar (GBR) DAA setup. This is an important step as it allows for greater debugging capabilities when trying to develop a completely new sensor to detect small UAS with a small radar cross section (RCS). Although an air-based DAA system is the ultimate goal, a ground-based solution comes with certain benefits that make it an attractive alternative option the the air-based DAA system. Airborne DAA systems require scaling sensors down to small UAS sizes which often requires compromises in range, field of view, measurement accuracy, or processing speed. Such compromises reduce the overall capability of the DAA system, and consequently, decrease the assurance of safety. In addition, carrying sensors on board reduces the UAS payload capability. Ground-based DAA systems do not require modifications to the UAS airframe, do not consume the UAS payload capacity, and reduce power consumption resulting in longer flight times. The ground-based DAA system provides an alternative means of complying with the FAA sense-and-avoid regulations. A ground-based DAA system consists of a ground control station that includes all sensors, communication, processing and logic. The main limitation of a ground-based DAA systems is that the aircraft which utilize this DAA system are restricted to fly in a confined volume of airspace, whereas, for air-based DAA systems the aircraft are free to fly within any volume of airspace.

For the results in this chapter we first show that we have been able to create a complete DAA system using a ground-based radar setup within simulation. These simulation efforts will then be extended to a complete DAA system using actual radar hardware in a GBR setup, shown in Chapter 5. The primary contributions of these two chapters include the implementation of a target detection and tracking algorithm using R-RANSAC, the integration of the subcomponents of the DAA system into fully functional simulation and hardware implementations, and other various contributions in the radar digital signal processing, and collision detection and collision avoidance algorithms.

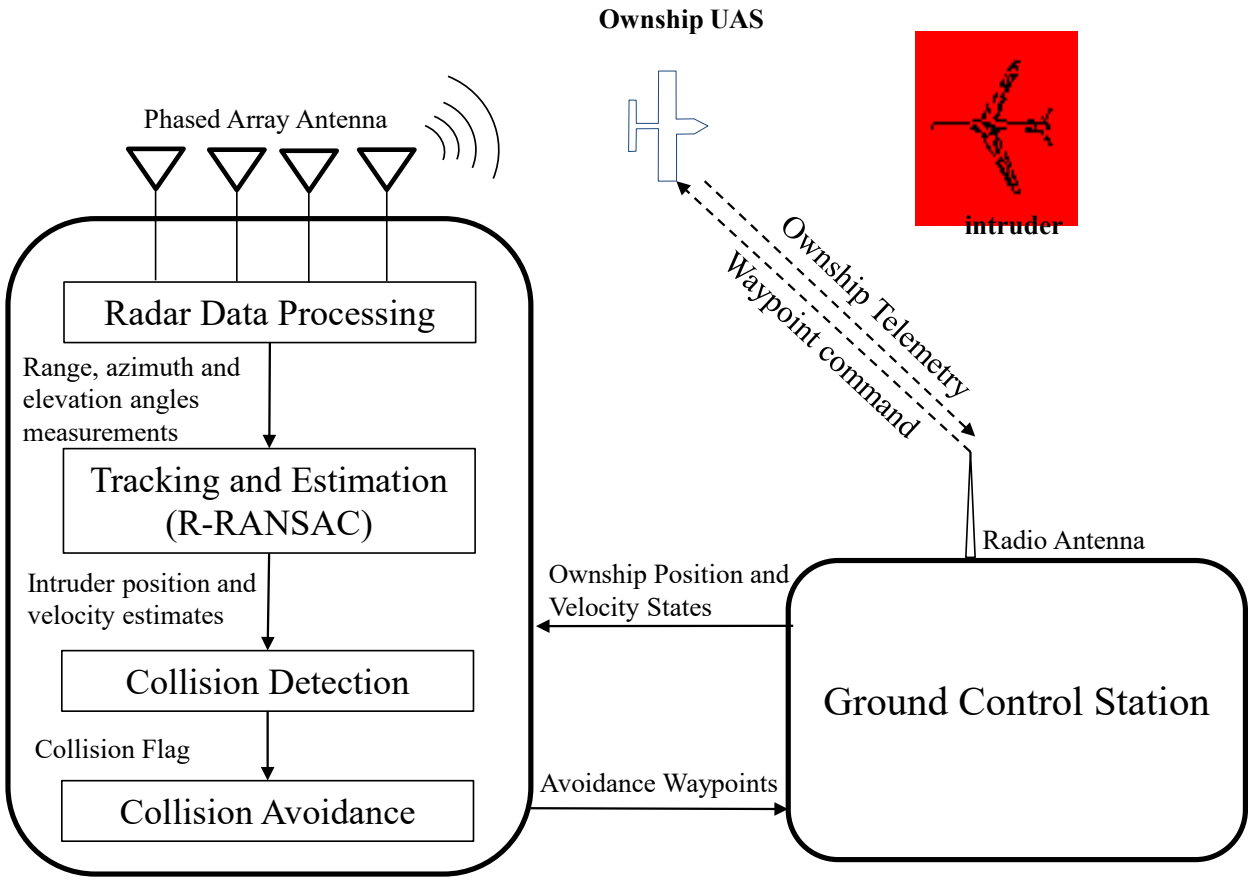


Figure 4.1: Ground-based radar detect-and-avoid system structure diagram.

The system shown in Figure 4.1 is a complete GBR DAA system for small UAS. It is viable for both fixed-wing and multirotor aircraft, and could reasonably be extended for larger UAS outside of the small UAS definition. As shown in Figure 4.1, radar returns from all of the targets are received by a phased-array antenna. The radar data is processed to produce range and azimuth and elevation angles to all targets. In the tracking step, the target's measurements are processed using the recursive-RANSAC (R-RANSAC) algorithm [12] to estimate the state estimates of potential intruders, and to distinguish the ownship. After the R-RANSAC filtering, the distance at the closest point of approach is computed to identify possible collisions. If a collision threat is detected, the intruder position and velocity estimates and an activation flag are passed into the collision avoidance algorithm. Once the collision level of the avoidance logic has been activated, a new collision-free path is generated

using a two-step path planning algorithm. In the first step, an initial suboptimal path is generated using A* search. In the second step, a simulated chain of unit masses connected by springs and dampers evolves in a simulated force field. The chain is described by a set of ordinary differential equations that is driven by virtual forces to find the steady-state equilibrium. The final output of the DAA system is a revised set of ownship waypoints that are transmitted to the ownship.

We now describe each of these subcomponents in greater detail as follows. In Section 4.1 we describe the simulated GBR model, in Section 4.2 we integrate the nonlinear radar output equations within a linear version of R-RANSAC, in Section 4.3 we describe a deterministic collision detection algorithm, in Section 4.4 we describe the two-step collision avoidance algorithm, and in Section 4.5 we show simulation results using each of these subcomponents.

4.1 Ground-based Phased-array Radar

Figure 4.2 shows the typical operating volumes associated with a ground-based DAA system. In this configuration, the ground-based sensor detects air traffic in a fixed volume of airspace called the surveillance volume. The ownship flies in a volume of airspace referred as the operation volume. The size and geometry of the operation volume is dependent on the surveillance volume, minimum detection range, and other dynamics characteristics of the UAS, like the minimum turning radius. The size of the operation volume should depend on (1) the minimum required detection range to be able to detect and track the intruder, (2) the time required to evaluate the encounter scenario, (3) the time required to plan an avoidance maneuver if required, and (4) the time required to take an evasive action. A drawback to using the ground-based DAA system is that it provides a static coverage volume, which may be less than the operating range of the UAS. Also, using ground-based DAA introduces the issue of maintaining a reliable, and efficient data link between the ground control station and the ownship. In addition, local terrain may also reduce the surveillance volume, and introduce noise in the measured information.

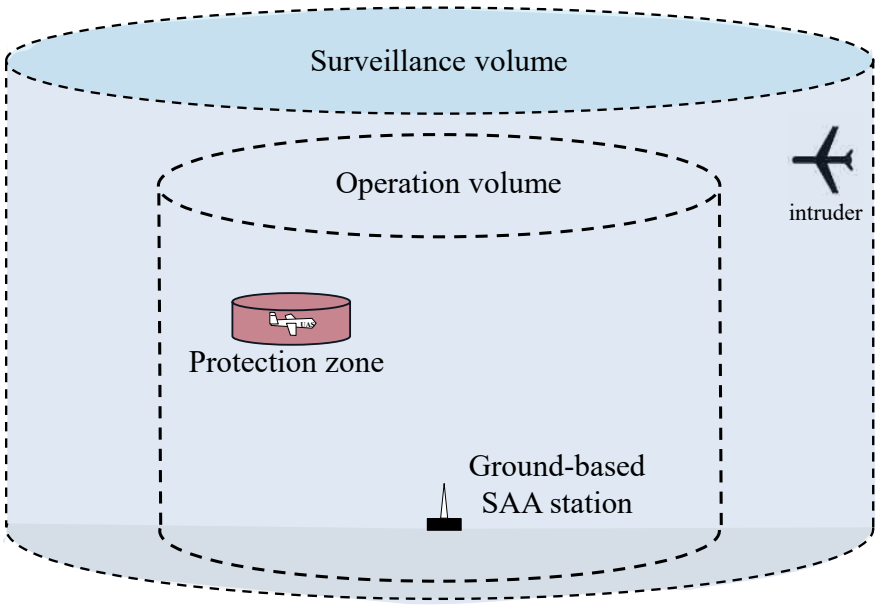
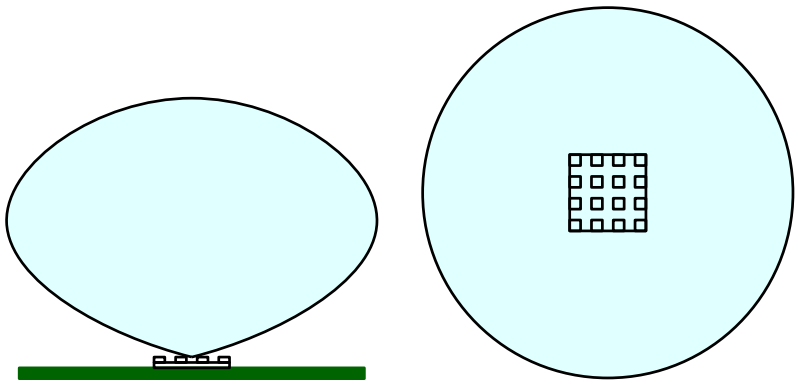


Figure 4.2: Surveillance and operating volumes associated with the ground-based radar DAA system.

For our specific ground-based DAA system we are attempting to use radar, which will provide measurements of range, azimuth angle, and elevation angle. These three measurements are needed to track the altitude of the aircraft, in addition to its horizontal position states. The radar antenna configuration we are using is a planar phased-array radar with a wide field of view angled directly at the sky. Due to beamwidth characteristics of radar, this would create a 3D surveillance volume above the radar in the form of an inverted raindrop as seen in Figure 4.3.

The primary development of a simulated GBR was done by Jonathan Spencer, Michael Boren, and Kaleo Roberts. This simulated radar was designed to mirror current efforts in the development of actual radar hardware. As such, a frequency-modulated, continuous-wave (FMCW) radar system has been designed with a four-by-four planar array antenna pattern. These array antennas are manufactured using low-cost printed circuit board (PCB) methods. Using the propagation path difference between each of these receiver array elements, the direction of arrival of a target can be found. This is done by first digitizing the signal and then performing digital beamforming. Digital beamforming allows the user to form many beams

and track multiple targets at different angles simultaneously. The four-by-four planar array pattern allows the user to determine a two element angle of arrival including the azimuth and elevation angle measurements.



(a) Side view. (b) Top view.

Figure 4.3: Radar surveillance volume using planar phased array.

Most modern radar systems, including air traffic control (ATC) radars, employ digital signal processing (DSP) on the signals generated by the radar. The different processing techniques range from simple operations such as averaging radar pulses to more sophisticated methods like synthetic aperture radar (SAR) imaging. The required DSP techniques depend on the type of radar system used and the desired information to be extracted. For our proposed ground-based radar, we need to extract range, azimuth angle, and elevation angle. To extract these three metrics, the DSP tools needed are the fast Fourier transform (FFT), correlation and averaging, target range detection, and digital beamforming.

The signal produced by an FMCW homodyne radar is a sum of sinusoids of different frequencies, where each frequency corresponds to a target's range. This signal is sampled N_s times at a rate of F_s to bring it into the digital domain. The sampled data is transformed from the time domain to the frequency domain via the FFT to get a range-compressed image (RCI) for a single radar pulse. Each RCI is composed of the positive frequency bins of the FFT, where the number of bins is $N_{bins} = N_s/2$ and each frequency bin is proportional to a specific range bin. The sampling and FFT are performed on each of the N_r receiver channels.

After the RCI's are attained from each channel, they are correlated together in preparation for digital beamforming. If the signals from each receiver are combined as

$$\mathbf{v}_i = [v_1[i], v_2[i], \ldots, v_{N_r}[i]]^\top, \tag{4.1}$$

where v_k denotes the values of the RCI from each channel $k = 1, 2, \ldots, N_r$, and $i = 1, 2, \ldots, N_{bins}$ denotes the i^{th} range bin, then the correlation matrix for each range bin is formed by

$$\boldsymbol{R}_i = \mathbf{v}_i \mathbf{v}_i^H. \tag{4.2}$$

To increase the signal-to-noise ratio (SNR) of the targets, N_{avg} correlation matrices, from consecutive radar pulses, are averaged to create the averaged correlation matrix $\overline{\boldsymbol{R}}_i$.

Using these averaged correlation matrices, target range detection is performed. Targets are detected by setting an amplitude threshold based on the noise statistics of the radar. The detection system then discriminates between targets and non-targets based on whether a particular return rises above the threshold or not. Research efforts specific this thesis have resulted in the use of a thresholding scheme similar to the thresholding technique described in Ref. [41]. In this thresholding method, the radar system sets a threshold based on an estimate of the noise power in each range bin. To calculate this estimate, we must take the following steps. First, we calculate the mean of the diagonal elements of the averaged correlation matrix as

$$P_i = \frac{1}{N_r} \sum_{j \leq N_r} \overline{\boldsymbol{R}}_i[j, j], \tag{4.3}$$

where P_i represents the mean power from each of the N_r channels in the i^{th} range bin. An example of this mean power for each range bin across multiple time steps is shown in Figure 4.4. For this particular example, the averaged correlation matrix $\overline{\boldsymbol{R}}_i$ was only based on a single correlation matrix, or equivalently $N_{avg} = 1$.

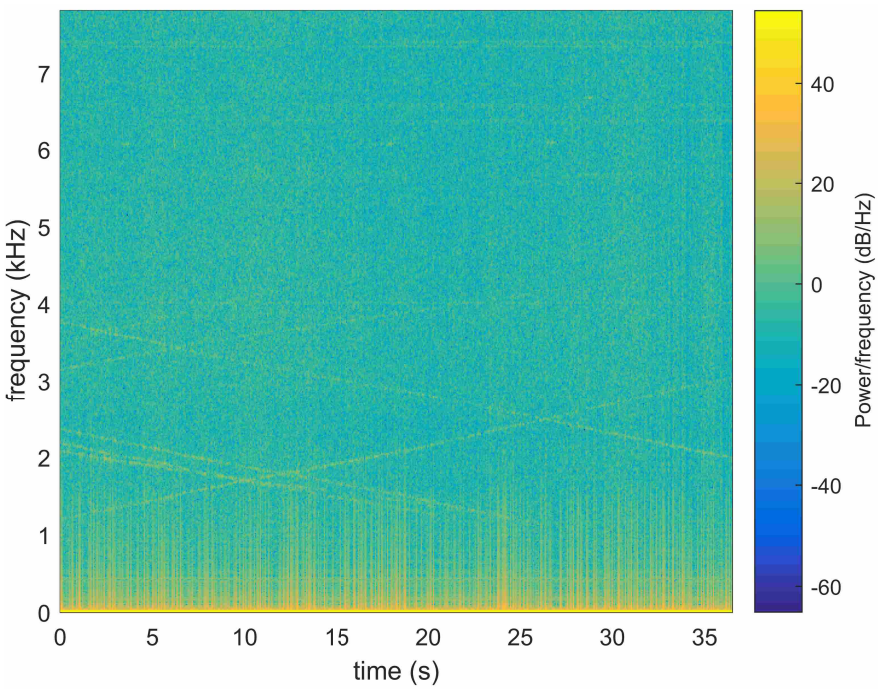


Figure 4.4: The mean power in each range bin over a discrete time window.

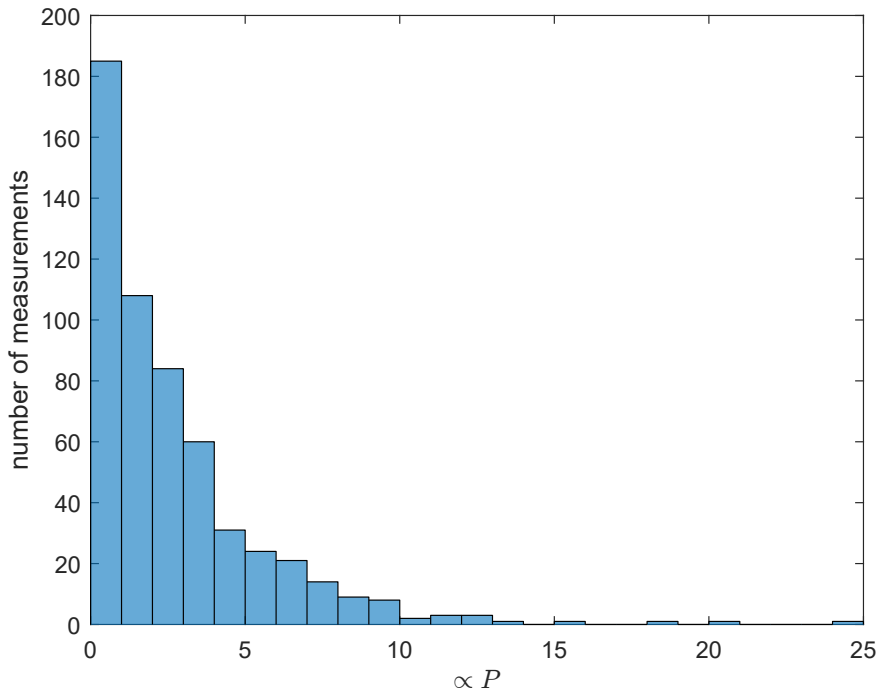


Figure 4.5: Histogram of range bin 150 across 557 time samples.

The data in Figure 4.4 was collected from the radar hardware directed at a sidewalk with multiple pedestrians walking in two different directions. From this data, we can determine the noise statistics for each range bin, and use them to create a threshold value to detect targets. As an example of how this is to be done, we create a histogram of range bin 150 across all 557 time samples as seen in Figure 4.5.

The distribution found from this histogram is representative of the distribution that will be found in each of the range bins in the data set, and which follows an exponential distribution shown as

$$p_i(P_i) = \lambda_i e^{-\lambda_i P_i},$$

where $\lambda_i = 1/\mu_i$ is the inverse of the noise power estimate in each range bin. Generally the expected value of the noise power estimate is found by saving N_{save} samples in a noise-only environment and then averaging them as

$$\mu_i = E[P_i] = \frac{1}{N_{save} - 1} \sum_{j \leq N_{save}} P_i(j).$$

By using this noise power estimate to calculate the exponential parameter λ_i for each range bin, we can write an expression for the probability of false alarm for each range bin as

$$\begin{aligned} P_{FA}^i &= \int_{T_i}^{\infty} \lambda_i e^{-\lambda_i P_i} dP_i \\ &= e^{-\lambda_i T_i}, \end{aligned}$$

where the probability of false alarm is equivalent to the integral of the pdf above a specified value. This value defines the starting point of the integral and is equivalent to the threshold value we are trying to find. By choosing an appropriate value for the probability of false alarm, this equation can be solved for the threshold as

$$T_i = \frac{\ln(1/P_{FA}^i)}{\lambda_i}. \tag{4.4}$$

Using this thresholding equation on the mean power data shown in Figure 4.4, we create a threshold for each range bin as seen by the orange line in Figure 4.6. For these threshold

values we have used a constant $P_{FA}^{i} = 0.01$. In addition to the threshold line, we have also randomly selected the first time step to provide an example of the mean power that would be expected at each time step. By comparing this mean power to the threshold level, we see that the mean power rises above the threshold in a few discrete range bins which signifies that a target has been detected in these particular range bins.

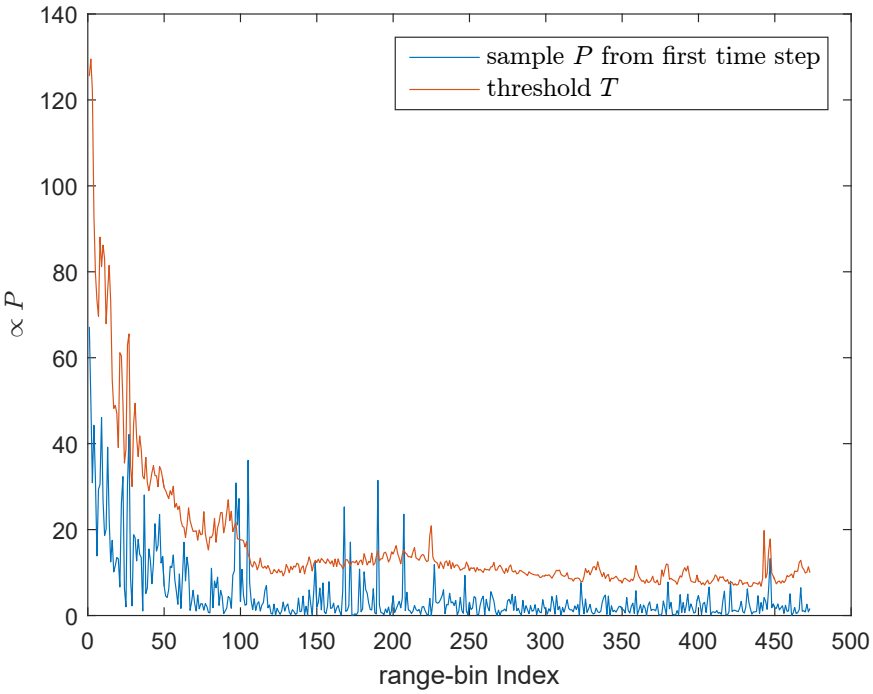


Figure 4.6: Threshold using a constant $P_{FA} = 0.01$.

Using these same threshold values on the mean power at each time step, we get the final thresholded data seen in Figure 4.7. From this figure we can see multiple diagonal lines which are a result of measurement returns from each of the walking pedestrians. We also see a significant amount of clutter measurements. These clutter measurements could be decreased by either increasing the number of correlation matrices we average, decreasing the probability of false alarm, or simply collecting measurements in a less noisy environment. Although these improvements could be made to reduce clutter, we have nonetheless shown that we have successfully created an appropriate thresholding scheme which works on real radar data.

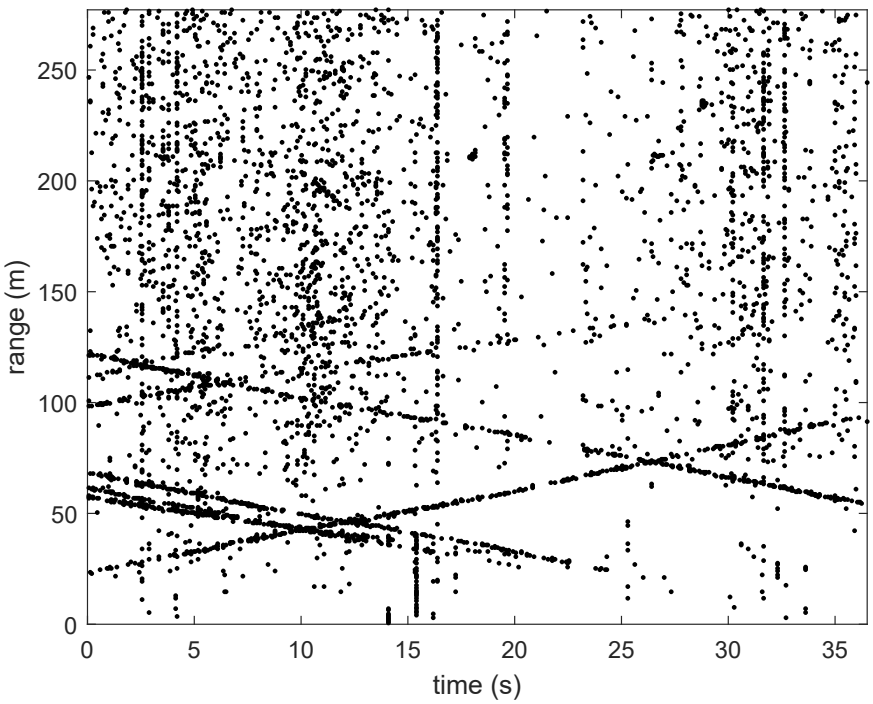


Figure 4.7: Final thresholded radar data in each range bin over a discrete time window.

Finally, digital beamforming is used to obtain the azimuth and elevation angles to each target. This utilizes $\overline{\boldsymbol{R}}[r]$ for each range bin r where a target is detected. The result of this process gives the return power as a function of range bin r, azimuth α, and elevation ϵ as

$$S[r, \alpha, \epsilon] = \mathbf{w}^H(\alpha, \epsilon)\overline{\mathbf{R}}[r]\mathbf{w}(\alpha, \epsilon), \tag{4.5}$$

where $\mathbf{w}(\alpha, \epsilon)$ is the weight vector for multiple discrete bearing angles (α, ϵ), and has been derived by Spencer in Ref. [42]. The bearing angle pair with the greatest returned power is used as the estimate of the target's azimuth and elevation angles.

4.2 Constant Acceleration R-RANSAC Tracker

As was mentioned in the desired tracker properties section of Chapter 3, to track maneuvering aircraft, an expanded set of states based on a constant-acceleration model can be used. This expanded set of states can either be implemented with a linear or nonlinear

dynamic model. In that chapter, we stated that the nonlinear model is better suited for predicting the states of the aircraft into the distant future because it appropriately models the turning, climbing, and accelerating dynamics of the aircraft. To utilize the added benefits of the nonlinear model, the collision detection and collision avoidance algorithms would both need to account for the the future flight path trajectories resulting from maneuvering aircraft. For the DAA system we are presenting in this chapter, the collision detection and collision avoidance algorithms do not account for maneuvering aircraft, but instead assume future trajectories that follow straight-line paths. As a result of this limitation, we must therefore restrict our DAA tests to scenarios where the intruders are flying straight line paths.

Since the collision detection and collision avoidance algorithms implemented in this DAA system do not account for maneuvering aircraft and the aircraft themselves are only flying straight line paths, a nonlinear tracking algorithm is no longer needed and we have chosen to utilize the original linear version of the R-RANSAC tracking algorithm. The states used within R-RANSAC will still however be defined by the expanded set of states based on a constant-acceleration model as

$$\mathbf{x} = [p_n, p_e, h, \dot{p_n}, \dot{p_e}, \dot{h}, \ddot{p_n}, \ddot{p_e}, \ddot{h}]^\top. \tag{4.6}$$

Since we are also using a ground-based radar as opposed to an on-board air-based radar, the nonlinear output equations for range, azimuth and elevation are slightly different and are defined as

$$h(\mathbf{x}) = \begin{pmatrix} r(\mathbf{x}) \\ \alpha(\mathbf{x}) \\ \epsilon(\mathbf{x}) \end{pmatrix}, = \begin{pmatrix} \sqrt{n^2 + e^2 + d^2} \\ \tan^{-1}\left(\frac{e}{n}\right) \\ \sin^{-1}\left(\frac{-d}{\sqrt{n^2+e^2+d^2}}\right) \end{pmatrix}, \tag{4.7}$$

where $[n, e, d]^\top$ is the north-east-down location of each aircraft relative to the radar as

$$n = p_n - n_{_{rd}},$$
$$e = p_e - e_{_{rd}},$$
$$d = -h - d_{_{rd}},$$

where $[n_{rd}, e_{rd}, d_{rd}]^\top$ is the north-east-down location of the ground-based radar station. Using the states defined above, the linear state propagation equation is shown as

$$\hat{\mathbf{x}}_k = A\hat{\mathbf{x}}_{k-1} = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_n \\ p_e \\ h \\ \dot{p_n} \\ \dot{p_e} \\ \dot{h} \\ \ddot{p_n} \\ \ddot{p_e} \\ \ddot{h} \end{bmatrix}. \tag{4.8}$$

The linear implementation of R-RANSAC also requires a linear output equation modeled as

$$\mathbf{y}_k = C\mathbf{x}_k \tag{4.9}$$

As shown in Equation (4.7), the range, azimuth, and elevation measurements are nonlinear in the states. To use the R-RANSAC algorithm, we must perform a nonlinear transformation of the range, azimuth, and elevation into three pseudo measurements of north position, east position, and altitude shown as

$$p_n = r\cos(\alpha)\cos(\epsilon) + n_{rd}, \tag{4.10}$$

$$p_e = r\sin(\alpha)\cos(\epsilon) + e_{rd}, \tag{4.11}$$

$$h = r\sin(\epsilon) - d_{radar}. \tag{4.12}$$

The C matrix in the linear output equation is therefore seen as

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.13}$$

One issue with these transformed measurements is that the noise characteristics cannot be accurately transformed into the new coordinate frame. This is a problem because the noise characteristics are used to decide if a new measurement is an inlier to an existing model. The noise characteristics in the range, azimuth, and elevation frame can be described as an inverted shallow bowl as illustrated in Figure 4.8. As the range to the target increases the diameter of this bowl also increases because σ_α and σ_ϵ remain constant. This means that the noise in the north, east, and altitude directions grows with range, and the relative noise among the three directions changes depending on the azimuth and elevation angles.

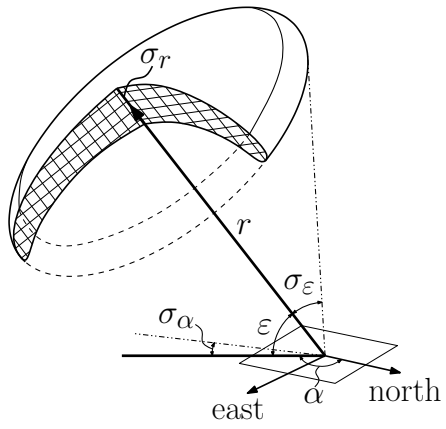


Figure 4.8: Covariance of radar measurements.

The solution used to solve this problem is to convert the estimated states into range, azimuth, and elevation using Equation (4.7), and then use τ_r, τ_α, and τ_ϵ to determine if the new measurements are inliers to the existing models. The specific inlier function needed is the nonlinear R-RANSAC inlier function described in Chapter 3 shown by Equation (3.10). If the measurements are inliers then they are converted into north position, east position, and altitude pseudo measurements as described above, and then are used in the update step of the Kalman filter. This has the advantage of maintaining the true noise characteristics for inlier detection, while still being able to use linear state space equations for the Kalman filter and other R-RANSAC equations.

Another modification is needed to make R-RANSAC work with a ground-based radar. Since a ground-based radar will provide measurements for all aircraft in its field of view, mea-

surements of the ownship will be sent to the R-RANSAC algorithm, a track will be created for the ownship, and the ownship track will be identified as an intruder. If the ownship is identified as an intruder, the path planner will create an avoidance path causing the ownship to deviate from its path when such a maneuver is not necessary. The modification made within R-RANSAC is to use the ownship's known states, received from a data link between the ownship and the radar ground control station, as a second set of measurements which are used to update an additional ownship consensus set χ_o and inlier ratio, ρ_o. These additional metrics require that we store the j^{th} R-RANSAC tracks as an increased eight-tuple $\mathcal{M}^j = (\hat{\mathbf{x}}^j, P^j, \chi^j, \rho^j, t^j, \mathcal{L}^j, \chi_o^j, \rho_o^j)$. The extra ownship measurements are only used to update the ownship consensus set and resulting ownship inlier ratio, and are not used in the measurement update step of the Kalman filter in updating the state estimates. To determine if a given set of ownship measurements are an inlier to a given model we first convert the ownship position states into range, azimuth, and elevation values using Equation (4.7), and then use the same nonlinear R-RANSAC inlier function used with the true radar measurements. If the current ownship measurement is indeed an inlier to a specific model, the ownship consensus set and ownship inlier ratio are updated. If the ownship inlier ratio ρ_o for any of the $\mathcal{M}^j$ models goes above the threshold $\tau_{\rho,o}$, then a flag is set identifying it as the ownship. If any of the models are identified as the ownship, they are subsequently not used within the collision detection or collision avoidance algorithms.

4.3 Collision Detection

Using the estimated states of the intruders found from the R-RANSAC algorithm, we must now predict if future collisions are expected to occur using a collision-detection algorithm. As was mentioned in Section 4.2, the collision-detection algorithm implemented in this DAA system does not account for maneuvering aircraft, but instead assumes future trajectories that follow straight-line paths. The primary development of this collision-detection algorithm was done by Sahawneh [43], however, additional capabilities have been added as a result of the research in this thesis.

The primary capability that has been added to this collision-detection algorithm is the ability to account for future waypoints of the ownship flight path. In the original

implementation, the ownship was assumed to fly in a straight line for an infinite duration of time. If the ownship has more than two waypoints, however, the transitions between each set of waypoints may cause the ownship to change its flight path direction, thus nullifying the assumption. This added capability takes the collision-detection algorithm one step closer to being able to detect collisions of maneuvering aircraft by considering the known maneuvers of the ownship, however, it still does not account for the maneuvers of the intruder aircraft.

The other capabilities have to do with understanding the various collision encounter scenarios which will lead to a collision. As a result of implementing the original collision-detection algorithm, we have discovered that a few flight encounter scenarios were not considered in the original collision detection logic. As a result, the original collision detection logic has been corrected to account for each of these scenarios. These additional capabilities will now be explained followed by a description of the updated collision-detection algorithm.

The safety volume which is used to determine if a collision has occurred was previously shown in Figure 2.2 of Chapter 2. This volume is in the form of a hockey puck with radius R_s and height h_s. By using this shape for the safety volume, the collision detection logic must be separated into two parts for the horizontal and vertical components, respectively. The basic idea of the collision-detection algorithm is to first find the times when the intruder is within the horizontal and vertical safety boundaries. These times are then compared to determine if the intruder is within the horizontal and vertical safety boundaries during the same time period. If they are within both boundaries during the same time period, then we conclude that a collision is expected to occur.

In addition to the intruder physically entering the safety volume at some future time, one of two additional conditions must be met for the collision-detection algorithm to produce a collision detection flag. For the first condition, the predicted collision must occur within some time threshold τ_T. For the second condition, both the intruder's horizontal and vertical distance from the ownship must fall within some horizontal and vertical distance threshold $\tau_{\mathbf{p}}$ and τ_h, respectively.

In the original collision-detection algorithm, the time at which the intruder exits the vertical boundary is compared to the time at which it enters the horizontal boundary. If the vertical exit time is greater than the horizontal entry time, then a collision is identified. This

is, however, only part of the logic necessary. In reality, all four entry and exit times for the horizontal and vertical components must be compared to determine if a collision will occur. A diagram showing each of these entry and exit times is found in Figures 4.9 and 4.10 for the horizontal and vertical dimensions, respectively.

Figures 4.9 and 4.10 are meant to demonstrate the entry and exit times resulting from every possible encounter scenario. Each of the scenarios shown were included in the original collision-detection algorithm, except one. The encounter scenario that was not included in the original collision-detection algorithm is seen in Figure 4.9 by the fourth intruder which starts inside the horizontal safety boundary. This type of scenario could occur for example if the intruder was descending on the ownship from above or ascending into the ownship from below. This additional encounter scenario is therefore included in the updated collision detection logic.

The added waypoint following capability, the correct entry and exit time logic, and the additional encounter scenario logic are all added to an improved collision-detection algorithm seen in Algorithms 3, 4, and 5. These algorithms are now described in detail. The inputs to the collision-detection algorithm include the horizontal and vertical positions of the ownship $\mathbf{p}_o = [p_{n,o}, p_{e,o}]^\top$ and h_o, respectively, the ownship ground speed V_g, the horizontal and vertical positions of the intruder $\mathbf{p}_i = [p_{n,i}, p_{e,i}]^\top$ and h_i, respectively, the horizontal and vertical velocities of the intruder $\mathbf{v}_i = [v_{n,i}, v_{e,i}]^\top$ and $v_{h,i}$, respectively, the number of future waypoints N, the future waypoint path $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$, the horizontal distance threshold $\tau_{\mathbf{p}}$, the vertical distance threshold τ_h, and the time to collision threshold τ_t.

To account for future waypoints of the ownship, the collision detection logic must be performed for discrete time intervals in the future. These time intervals start with the time at each waypoint node and end with the time at the waypoint node immediately following. The collision-detection algorithm, therefore, iterates through each of the waypoint segments (Line 2), where the start time has been initialized on Line 1. As will be shown later, this start time will be updated at each successive waypoint node. Next we calculate the 3D position vector from the current location of the ownship to the next waypoint as $\mathbf{n}$ (Line 3).

Table 4.1: Algorithm for collision detection

Algorithm 3 Collision Detection Algorithm

Input: Ownship horizontal and vertical positions $\mathbf{p}_o = [p_{n,o}, p_{e,o}]^\top$ and h_o, respectively, ownship ground speed V_g, intruder horizontal and vertical positions $\mathbf{p}_i = [p_{n,i}, p_{e,i}]^\top$ and h_i, respectively, intruder horizontal and vertical velocities $\mathbf{v}_i = [v_{n,i}, v_{e,i}]^\top$ and $v_{h,i}$, respectively, number of future waypoints N, future waypoint path $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$, horizontal distance threshold $\tau_{\mathbf{p}}$, vertical distance threshold τ_h, and time to collision threshold τ_t.

$t_{start} \leftarrow 0$
for $i \leftarrow 1$, i++, while $i \leq N$ **do**
  $\mathbf{n} \leftarrow \mathbf{w}_i - [\mathbf{p}_o^\top, h_o]^\top$
  $t_{\mathbf{n}} \leftarrow \frac{\|\mathbf{n}\|}{V_g}$
  $\mathbf{v}_o = \mathcal{V}_o(1,2)$, $v_{h,o} = \mathcal{V}_o(3)$, where $\mathcal{V}_o = V_g \frac{\mathbf{n}}{\|\mathbf{n}\|}$
  $\mathbf{p}_r \leftarrow \mathbf{p}_i - \mathbf{p}_o$
  $\mathbf{v}_r \leftarrow \mathbf{v}_i - \mathbf{v}_o$
  $[t_{ent}^{\mathbf{p}}, t_{ext}^{\mathbf{p}}]^\top \leftarrow \text{EntExtPosition}(\mathbf{p}_r, \mathbf{v}_r, R_s)$
  $h_r \leftarrow h_i - h_o$
  $v_{h,r} \leftarrow v_{h,i} - v_{h,o}$
  $[t_{ent}^h, t_{ext}^h]^\top \leftarrow \text{EntExtAltitude}(h_r, v_{h,r}, h_s)$
  $t_{ent} \leftarrow \max t_{ent}^{\mathbf{p}}, t_{ent}^h$
  $t_{ext} \leftarrow \min t_{ext}^{\mathbf{p}}, t_{ext}^h$
  if i=1 **then**
    $\|\mathbf{p}\|_{r,0} \leftarrow \|\mathbf{p}\|_r$
    $h_{r,0} \leftarrow h_r$
  end if
  $t_{col} = t_{start} + t_{ent}$
  if $t_{ent} \leq t_{ext}$ and $t_{ent} \leq t_{\mathbf{n}}$ and $((\|\mathbf{p}\|_{r,0} \leq \tau_{\mathbf{p}}$ and $h_{r,0} \leq \tau_h)$ or $t_{col} \leq \tau_T)$ **then**
    $\mathbf{C}(i) \leftarrow 1$
  else
    $\mathbf{C}(i) \leftarrow 0$
  end if
  $t_{start} \leftarrow t_{start} + t_{\mathbf{n}}$
  $\mathbf{p}_i \leftarrow \mathbf{p}_i + t_{\mathbf{n}}\mathbf{v}_i$
  $h_i \leftarrow h_i + t_{\mathbf{n}} v_{h,i}$
  $\mathbf{p}_o \leftarrow \mathbf{w}_i(1,2)$
  $h_o \leftarrow \mathbf{w}_i(3)$
end for
col $\leftarrow \sum_{i \leq N} \mathbf{C}_i > 0$

Using this 3D position vector and the ground speed of the ownship, we calculate the time to the next waypoint node $t_{\mathbf{n}}$ (Line 4). Also using the 3D position vector and the ground speed of the ownship, we calculate the 3D velocity vector of the ownship at the current waypoint node $\mathcal{V}_o$. The first two elements of this velocity vector is the horizontal velocity vector of the ownship $\mathbf{v}_o$, and the third element is the vertical velocity of the ownship $v_{h,o}$ (Line 5). Next we calculate the relative horizontal position and velocity vectors between the intruder and the ownship as, $\mathbf{p}_r$ and $\mathbf{v}_r$, respectively. (Lines 6-7). Using these relative horizontal position and velocity vectors and the horizontal safety radius, we then run the EntExtPosition() function, shown in Algorithm 4, which will be described later, to find the horizontal entrance and exit times as, $t^{\mathbf{p}}_{ent}$ and $t^{\mathbf{p}}_{ext}$, respectively (Line 8). Next we calculate the relative vertical position and vertical velocity between the intruder and the ownship as, h_r and $v_{h,r}$, respectively (Lines 9-10). Using these relative vertical position and velocity values and the vertical safety height, we then run the EntExtAltitude() function, shown in Algorithm 5, which will be described later, to find the vertical entrance and exit times as, t^h_{ent} and t^h_{ext}, respectively (Line 11). Next we find the union of the horizontal and vertical time intervals to define when the intruder is within both the horizontal and vertical safety boundaries simultaneously. The entrance time of the union is calculated as the maximum of the entrance times from the horizontal and vertical dimensions t_{ent} (Line 12). The exit time for this union is calculated as the minimum of the exit times from the horizontal and vertical dimensions t_{ext} (Line 13). Next we save the relative horizontal and vertical distances between the two aircraft from their initial locations as $\|\mathbf{p}\|_{r,0}$ and $h_{r,0}$, respectively (Lines 14-17), and then use the updated entry time t_{ent} to calculate the time to collision as measured relative to the time when the algorithm was first initiated t_{col} (Line 18).

We are now set up to perform the primary logic used to determine if a collision is expected to occur during the current waypoint segment (Line 19). This logic includes three conditions that must all be true. The first condition compares the entrance and exit times resulting from the union of the horizontal and vertical entrance and exit times. For the intruder to be within the horizontal and vertical safety boundaries at the same time, then the entrance time must be less than or equal to the exit time. The second condition compares the entrance time to the time to the next waypoint node. For a collision to occur during

this waypoint segment, the entrance time must be less than or equal to the time to the next node. The final condition was explained previously and has to do with the horizontal and vertical distance thresholds and time threshold. For the horizontal and vertical distance threshold comparison, we are using the distances from the initial location of the ownship and intruder, which will not change between iterations of the main for loop. For the time threshold comparison, we are using the time to collision as measured relative to the time when the algorithm was first initiated. If each of the three conditions mentioned have been satisfied, then a collision flag for the current waypoint segment $\mathbf{C}(i)$ is set to one (Line 20). If one of these three conditions is not satisfied, then the collision flag for the current waypoint segment is set to zero. The last step within the for loop is to update the start time and position variables (Lines 24-28). The start time is updated by adding the previous start time with the time to the next node (Line 24). The intruder's horizontal and vertical starting position and altitude at the time of the next waypoint is updated according to a constant velocity model (Lines 25-26). The ownship's horizontal and vertical starting position and altitude at the time of the next waypoint is updated by simply using the location of the next waypoint node (Lines 27-28). The final step of the collision-detection algorithm is to sum the elements of the collision flag vector (Line 30). If this sum is greater than zero, then a collision was detected on at least one of the waypoint line segments followed by the ownship and we conclude that a collision is expected to occur.

Next we describe the EntExtPosition() function shown in Algorithm 4. The purpose of this algorithm is to calculate times at which the intruder enters and exits the horizontal safety radius around the ownship. The equations used within this algorithm were developed by Sahawneh [43], however, we have combined the necessary logic into a single algorithm. For the description of this algorithm we will provide specific examples from Figure 4.9. The inputs to this algorithm are the relative horizontal position and velocity vectors along with the safety radius. First we check if the distance between the two aircraft is greater than the safety radius (Line 1). This condition can be seen in Figure 4.9 by intruders 1, 2, and 3, which are outside the safety radius. Next we check if the two aircraft are converging (Line 2). This is seen by intruders 1 and 2 at the top of the figure. Next we calculate the time to closest point of approach t_{cpa} and the resulting distance at the closest point of approach d_{cpa}

(Lines 3-4). Next we check if the distance at the closest point of approach is greater than the safety radius (Line 5). This is seen by intruder 1. If each of these conditions have been met, the intruder is not expected to penetrate the horizontal safety radius and the entrance and exit times are therefore set to infinity (Line 6). If the distance at the closest point of approach is less than the safety radius, then the intruder will penetrate the horizontal

Table 4.2: Algorithm to find the entrance and exit times horizontally.

Algorithm 4 EntExtPosition Algorithm

Input: Relative position $\mathbf{p}_r$, relative velocity $\mathbf{v}_r$, horizontal safety radius R_s.

1: **if** $\|\mathbf{p}_r\| > R_s$ **then**
2: **if** $\mathbf{p}_r^\top \mathbf{v}_r < 0$ **then**
3: $t_{cpa} \leftarrow \frac{-\mathbf{p}_r^\top \mathbf{v}_r}{\|\mathbf{v}_r\|^2}$
4: $d_{cpa} \leftarrow \sqrt{\|\mathbf{p}_r\|^2 + t_{cpa}\mathbf{p}_r^\top \mathbf{v}_r}$
5: **if** $d_{cpa} > R_s$ **then**
6: $t_{ent}^{\mathbf{p}}, t_{ext}^{\mathbf{p}} \leftarrow \infty$
7: **else**
8: $t_{ent}^{\mathbf{p}} \leftarrow \frac{-\mathbf{p}_r^\top \mathbf{v}_r - \sqrt{\Delta}}{\|\mathbf{v}_r\|^2}$, $t_{ext}^{\mathbf{p}} \leftarrow \frac{-\mathbf{p}_r^\top \mathbf{v}_r + \sqrt{\Delta}}{\|\mathbf{v}_r\|^2}$, where $\Delta \leftarrow (\mathbf{p}_r^\top \mathbf{v}_r)^2 - \|\mathbf{v}_r\|^2 (\|\mathbf{p}_r\|^2 - R_s^2)$
9: **end if**
10: **else**
11: $t_{ent}^{\mathbf{p}}, t_{ext}^{\mathbf{p}} \leftarrow \infty$
12: **end if**
13: **else**
14: $t_{ent}^{\mathbf{p}} \leftarrow 0$
15: **if** $\|\mathbf{v}_r\| = 0$ **then**
16: $t_{ext}^{\mathbf{p}} \leftarrow \infty$
17: **else**
18: $t_{ext}^{\mathbf{p}} \leftarrow \frac{-\mathbf{p}_r^\top \mathbf{v}_r + \sqrt{\Delta}}{\|\mathbf{v}_r\|^2}$, where $\Delta \leftarrow (\mathbf{p}_r^\top \mathbf{v}_r)^2 - \|\mathbf{v}_r\|^2 (\|\mathbf{p}_r\|^2 - R_s^2)$
19: **end if**
20: **end if**
21: **return** $[t_{ent}^{\mathbf{p}}, t_{ext}^{\mathbf{p}}]^\top$

safety radius and the entrance and exit times are calculated according equations developed in Ref. [43] (Lines 7-8). This is seen by intruder 2. If the intruder is outside the safety radius and not converging, then a collision is not expected to occur and the entrance and exit times are set to infinity (Lines 10-11). This is seen by intruder 3. Now we consider the case where the intruder starts within the horizontal safety radius (Line 13). This is seen by intruder 4. This was the scenario that was not taken into account in the original collision-detection algorithm. Since these intruders are already inside the horizontal safety radius, the entrance time is set to zero (Line 14). If the relative horizontal velocity between the two aircraft is zero, then the intruder will never leave the horizontal safety radius and the exit time is set to infinity (Lines 15-16). If on the other hand the relative velocity is not equal to zero, then the intruder will exit the horizontal safety radius at some instant in time according the equations developed in Ref. [43] (Lines 17-18). Having considered all possible cases for the horizontal entrance and exit times, we return these values to the main collision-detection algorithm.

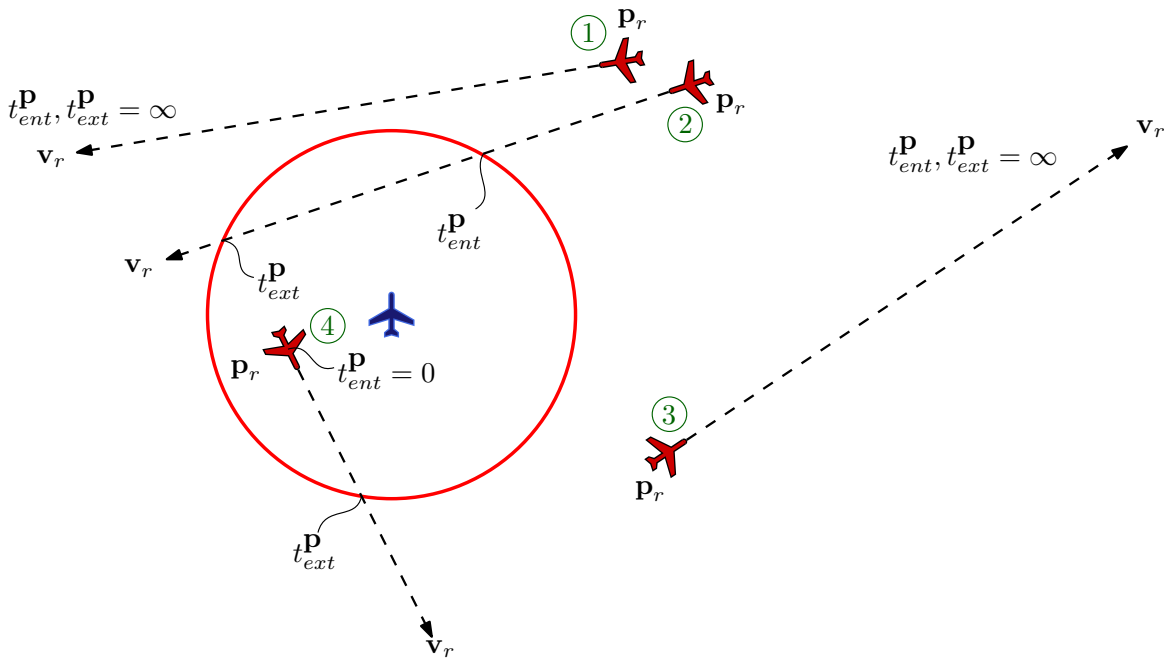


Figure 4.9: Horizontal encounter scenarios.

Finally we describe the EntExtAltitude() function shown in Algorithm 5. The purpose of this algorithm is to calculate times at which the intruder enters and exits the vertical safety height above and below the ownship. For the description of this algorithm we will provide specific examples from Figure 4.10. The inputs to this algorithm are the relative altitude and altitude velocity values along with the safety height. First we check if the magnitude of the altitude difference between the two aircraft is greater than the safety height (Line 1). This condition is seen by intruders 1, 2, 3, 4, 5, and 6. Next we check if the two aircraft are converging (Line 2). This is seen by intruders 3 and 4. Then we check if the intruder is above the ownship (Line 3). This is seen by intruder 3. If each of these conditions have been met, the intruder is expected to penetrate the vertical safety height from above. The entrance time is then found from the time it takes to hit the top of the safety volume (Line 4). The exit time is found form the time it takes to hit the bottom of the safety volume (Line 5). Alternatively the intruder may start below the ownship (Line 6). This is seen by intruder 4. In this case the intruder will penetrate the vertical safety height from below. The entrance time is therefore found from the time it takes to hit the bottom of the safety volume (Line 7). The exit time is now found from the time it takes to hit the top of the safety volume (Line 8). If the intruder is outside the vertical safety volume and is not converging to the ownship, then the intruder is not expected to penetrate the vertical safety volume and the entrance and exit times are set to infinity (Lines 10-11). This is seen by intruders 1, 2, 5, and 6. Now we consider the case where the intruder starts within the vertical safety height (Line 13). This is seen by intruders 7, 8, 9, 10, 11, and 12. Since these intruders are already inside the vertical safety height, the entrance time is set to zero (Line 14). If the relative vertical velocity between the two aircraft is zero, then the intruder will never leave the vertical safety height and the exit time is set to infinity (Lines 15-16). This is seen by intruders 8 and 11. If on the other hand the relative velocity is not equal to zero, then the intruder will exit the vertical safety height at some future point in time (Line 17). This is seen by intruders 7, 9, 10, and 12. If the intruder is traveling in a downwards direction, then the exit time is set equal to the time it takes to hit the bottom of the safety volume (Lines 18-19). This is seen by intruders 9 and 12. If the intruder is traveling in an upwards direction, then the exit time is set equal to the time it takes to hit the top of the safety volume (Lines 20-21). Having

considered all possible cases for the vertical entrance and exit times, we return these values to the main collision-detection algorithm.

Table 4.3: Algorithm to find the entrance and exit times vertically.

Algorithm 5 EntExtAltitude Algorithm

Input: Relative altitude h_r, relative altitude velocity $v_{h,r}$, altitude safety height h_s.

if $|h_r| > \frac{h_s}{2}$ **then**
  if $h_r v_{h,r} < 0$ **then**
    if $h_r > 0$ **then**
      $t^h_{ent} \leftarrow t_{top} \leftarrow \frac{h_r - h_s/2}{v_{h,r}}$
      $t^h_{ext} \leftarrow t_{bottom} \leftarrow \frac{h_r + h_s/2}{v_{h,r}}$
    else
      $t^h_{ent} \leftarrow t_{bottom} \leftarrow \frac{h_r + h_s/2}{v_{h,r}}$
      $t^h_{ext} \leftarrow t_{top} \leftarrow \frac{h_r - h_s/2}{v_{h,r}}$
    end if
  else
    $t^h_{ent}, t^h_{ext} \leftarrow \infty$
  end if
else
  $t^h_{ent} \leftarrow 0$
  if $v_{h,r} = 0$ **then**
    $t^h_{ext} \leftarrow \infty$
  else
    if $h_r > 0$ exclusive or $v_{h,r} > 0$ **then**
      $t^h_{ext} \leftarrow t_{bottom} \leftarrow \frac{h_r + h_s/2}{v_{h,r}}$
    else
      $t^h_{ext} \leftarrow t_{top} \leftarrow \frac{h_r - h_s/2}{v_{h,r}}$
    end if
  end if
end if
return $[t^h_{ent}, t^h_{ext}]^\top$

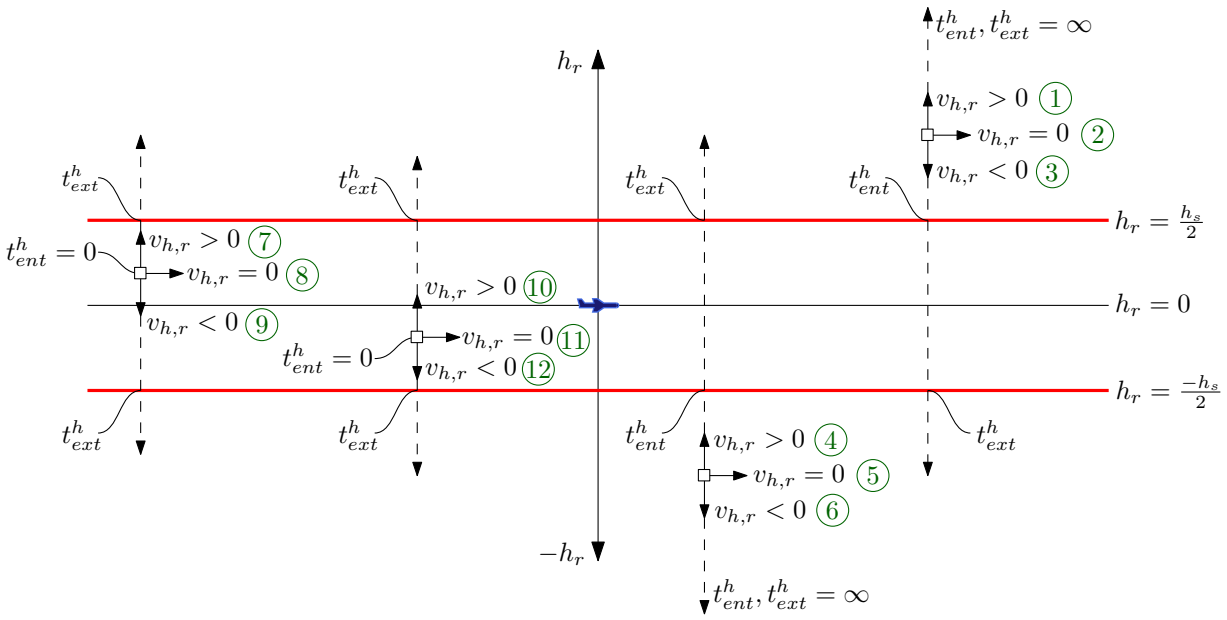


Figure 4.10: Vertical encounter scenarios.

4.4 A*/Chain-based Collision Avoidance

If a collision has been detected by the collision detection algorithm, the collision avoidance algorithm must be executed to create a collision free waypoint path. The primary development done for this algorithm was also performed by Sahawneh [43]. Although, the general concept for this algorithm was developed by Sahawneh, specific implementation details were also developed as a result of the research in this thesis.

One of the primary contributions was the integration of the each of the pieces of the DAA algorithm. Specifically this required the development of a waypoint control function. This function includes both the collision detection and collision avoidance algorithms, and is responsible for determining when new waypoint paths should be generated and for sending the new waypoints out for the ownship to start executing. To determine if the collision avoidance algorithm should be run, this waypoint control function requires that the collision detection function detect a collision for a predetermined consecutive number of times N_{col}. Once the collision detection function has been run and the ownship begins executing the current avoidance path, the intruders may deviate from their initial flight paths or the estimate of the intruders paths might change. In either case, the intruders may now be

moving in a direction that will once again result in a collision with the current waypoint flight path of the ownship, which will trigger a second execution of the collision detection algorithm. The waypoint control function must, therefore, also take on the responsibility of managing multiple collision avoidance paths that may be created throughout the duration of the collision avoidance maneuver. This is done by locking all collision avoidance waypoints that the ownship has already visited, discarding each of the remaining collision avoidance waypoints, and recalculating a new set of waypoints using the collision avoidance algorithm from the current location of the ownship to its next original waypoint.

The specific collision avoidance algorithm used in this DAA system is a two-step algorithm that first uses the A* algorithm to initialize a set of waypoints based on fixed nodes within the operation volume of the GBR [44], and second updates these waypoints based on a chain-based algorithm that uses the physical analogy of a chain placed in a force field [45,46]. The specific details of these algorithms will not be given in this thesis, however, we will describe a few of the contributions resulting from this thesis. Through the integration of this two-step algorithm within the larger complete DAA system, we found it desirable to append certain functionality within each of these algorithms to enable more robust collision avoidance planning. The details of these appendages are given in the remaining two paragraphs of this section.

In the development of the A* algorithm various costs are assigned to each of the possible nodes that the ownship could move towards. The general equation for the cost at a particular node $\mathbf{w}_b$ is shown as

$$g(\mathbf{w}_b) = g(\mathbf{w}_a) + c_d(\mathbf{i}_a, \mathbf{i}_b) + c_t(\mathbf{w}_b, \mathbf{p}_{int}^j) + c_n(\mathbf{w}_b) + c_{cpa}(\mathbf{p}_r, \mathbf{v}_r, h_r, v_{h,r}) + c_{\mathbf{V}}(\mathbf{V}_o, \mathbf{V}_i) \quad (4.14)$$

where $\mathbf{w}_a$ is the previous node, and c_i represents various costs. The cost c_d is the distance cost which is calculated based on the distance to the next waypoint and then continuing on to the final waypoint. Longer distances have a higher cost. The c_t cost is the threat cost which is calculated based on the locations of the two aircraft at the future node. The cost c_n is a nominal path cost which is calculated based on the deviation of the node from the original waypoint path of the ownship. The addition of the remaining two costs has resulted

from research efforts shown in this thesis. The c_{cpa} cost is the closest point of approach cost. This cost uses the same closest point of approach equations shown in the collision detection algorithm. If the distance at the closest point of approach multiplied by the time to the closest point of approach is small, a greater penalty is applied as seen by the equation

$$c_{cpa} = \frac{k_{cpa}}{d_{cpa} t_{cpa}}, \tag{4.15}$$

where k_{cpa} is a tunable cost parameter. The reason for including d_{cpa} in the denominator is because if a small distance at the closest point of approach is expected to occur, then a high cost should be applied to this path. The reason for including t_{cpa} in the denominator is primarily to offset the effects of a small d_{cpa} in certain situations. Specifically, if a small d_{cpa} is not expected to occur for an extended period of time, then we should temporarily reduce this cost allowing the ownship greater versatility in its flight path to overcome other threats in the immediate future. The $c_{\mathbf{v}}$ cost is a cost that rewards a node that causes the ownship to travel in a direction opposite the direction the intruder is traveling. The idea behind this cost is to try and select nodes which are behind the intruder. This cost is expressed mathematically as

$$c_{\mathbf{V}} = \frac{k_{\mathbf{V}} \mathbf{V}_o \cdot \mathbf{V}_i}{t_{cpa}}, \tag{4.16}$$

where $k_{\mathbf{v}}$ is another tunable cost parameter. If the ownship and intruder velocity vectors are in the same direction then the dot product will be a large positive number, if they are in opposite directions then the dot product will be a large negative number. The reason for including t_{cpa} in the denominator has similar reasonings as described above for the closest point of approach cost.

In the development of the chain-based algorithm various forces are applied to each of the nodes that cause the chain to settle on a more appropriate set of waypoints. One of the forces developed for this chain resulted from research efforts shown in this thesis. The force that we have added is a force that pushes the node behind the intruder if it is within a certain distance of the intruder as

$$\mathbf{F}_b = \frac{k_b}{\|\mathbf{p}_r\|}(-\mathbf{V}_i), \tag{4.17}$$

where k_b is a tunable parameter. In this equation we divide by the norm of the relative position vector. This results in large forces being applied that drive the ownship path away from the intruder as the relative position becomes small. The direction that this force is applied is opposite the direction of the intruder velocity vector. The benefit of this force is that it does not allow the ownship path to be pushed directly in front of an intruder which would certainly lead to a collision.

4.5 Simulation Results

To demonstrate the performance of the proposed ground-based radar sensor model, the R-RANSAC estimation scheme, and the collision detection and avoidance algorithms, we developed a simulation environment with a six-degree-of-freedom aircraft model for both the ownship and the intruders as described in Appendix C. The encounter geometry is constructed using typical collision encounters that include multiple intruders flying at different altitudes, approaching head-on, converging, and overtaking scenarios.

In the first encounter scenario the ownship starts at $(-400, 0, -200)^\top$ in the NED coordinate system, with an initial heading of 0 degrees measured from north and follows a straight line path at a constant speed of 13 m/s to reach a waypoint located at $(500, 0, -200)^\top$ as shown in Figure 4.11. The radar system is located at $(0, 0, 0)^\top$ and uses a simulated transmit power of 5 kW. The radar is simulated with a center frequency of 10.25 GHz, with a 500 MHz bandwidth and a 2 ms chirp time. We also simulate sampling 4096 times at a sampling frequency of 2.048 MHz. The receiver antennas are simulated as a four-by-four array for a total of 16 receiver channels. We also averaged 20 correlation matrices and used a probability of false alarm of 0.1. In the following simulations, our choice of the collision volume is a cylinder of radius, d_s=153 m (500 ft) and height, h_s =61 m (200 ft) centered on each of the intruders. All aircraft use a simulated radar cross section of 0.1 m^2.

In each encounter scenario, the intruders are following a straight-line path at a constant velocity and altitude. Figure 4.11 shows the first encounter scenario. It consists of two intruders: one is approaching head on and the other is converging from the right. The speed of the intruders are 17 m/s and 15 m/s, respectively. The altitude of both intruders is 200 m. If no collision avoidance is planned, the d_{cpa} with respect to the first and second

intruders is approximately 67.5 m and 66.5 m. Since all aircraft are flying at same altitude and the d_{cpa} is less than the horizontal safety distance d_s, then these encounters will lead to a collision. Figures 4.12, 4.13(a) and 4.13(b) shows the intruder paths and the avoidance path planned by the ownship.

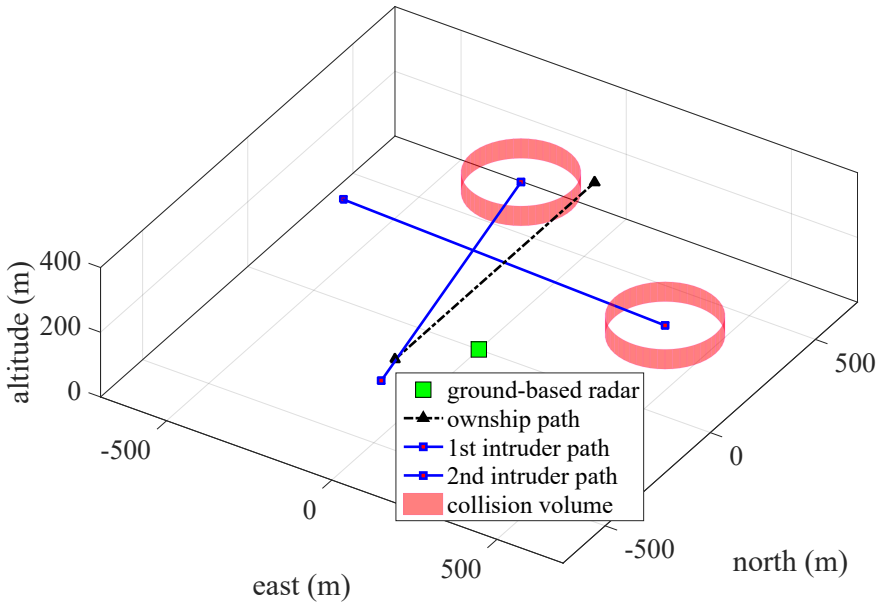


Figure 4.11: Encounter scenario number 1.

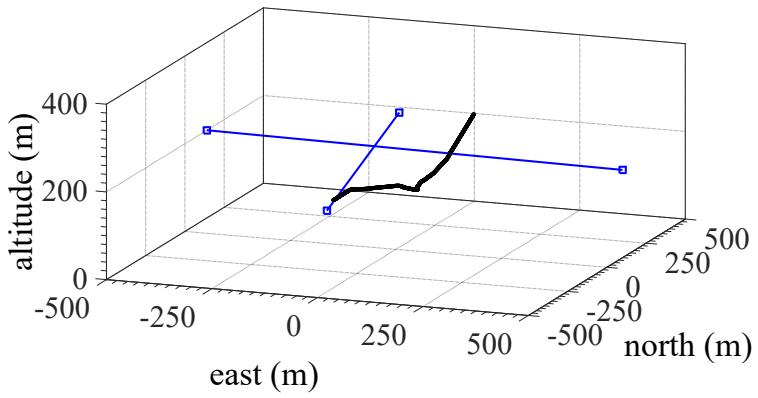
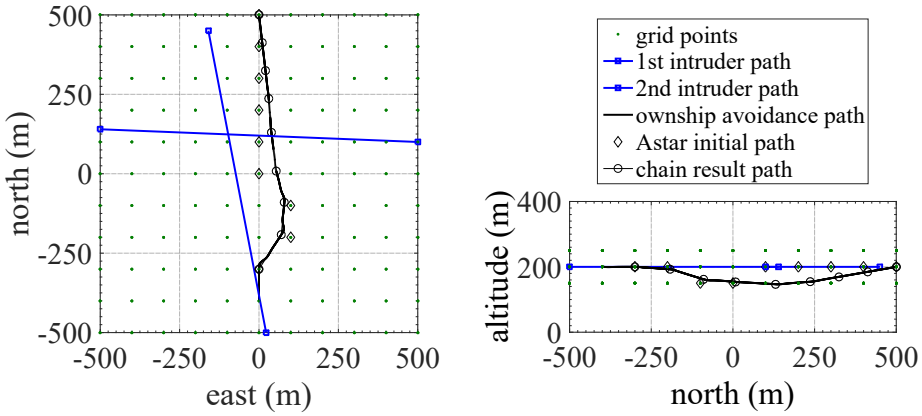


Figure 4.12: The avoidance path of the ownship.



(a) Top view of the ownship's avoidance path.

(b) Side view of the ownship's avoidance path.

Figure 4.13: Avoidance path followed by the ownship in encounter scenario number 1.

Table 4.4: Collision avoidance parameters.

A*		Chain	
Parameter	Value	Parameter	Value
k_{dist}	100	k	2.5
k_{alt}	559	k_{sh}, k_{sv}	200
$k_{\max}$	1e6	κ	2.5
ϱ	2.3	b	4
k_r	500	$\theta_{\max}$	45 deg
		λ	20
		$\gamma_{\max}$	30 deg
		$f_{\max}$	20
		δ_1	2.3
		δ_2	2.639

The parameters used for collision detection in the simulation were d_{th}= 800 meters and τ_{th}= 25 seconds. For the collision avoidance path planning algorithms a grid size of 100 meters horizontally and 50 meters vertically were used. The parameters used for collision avoidance are shown in Table 4.4.

Figure 4.14 shows the range, azimuth and elevation to all aircraft measured by the radar system. These figures predict that the principal signal decay happens at low elevation angles as the aircraft enter and exit the antenna beam. Despite the noise in the elevation angle data, range and azimuth measurements maintain a high degree of accuracy until the aircraft exits the radar sensing volume at about 600 m. The aircraft with radar measurements in the NED inertial frame are shown in Figure 4.15. Figure 4.15(a) shows that the north and east coordinates constructed from radar measurements have a high degree of accuracy, whereas Figure 4.15(b) shows that the altitude has a lower degree of accuracy because of noisy elevation measurements.

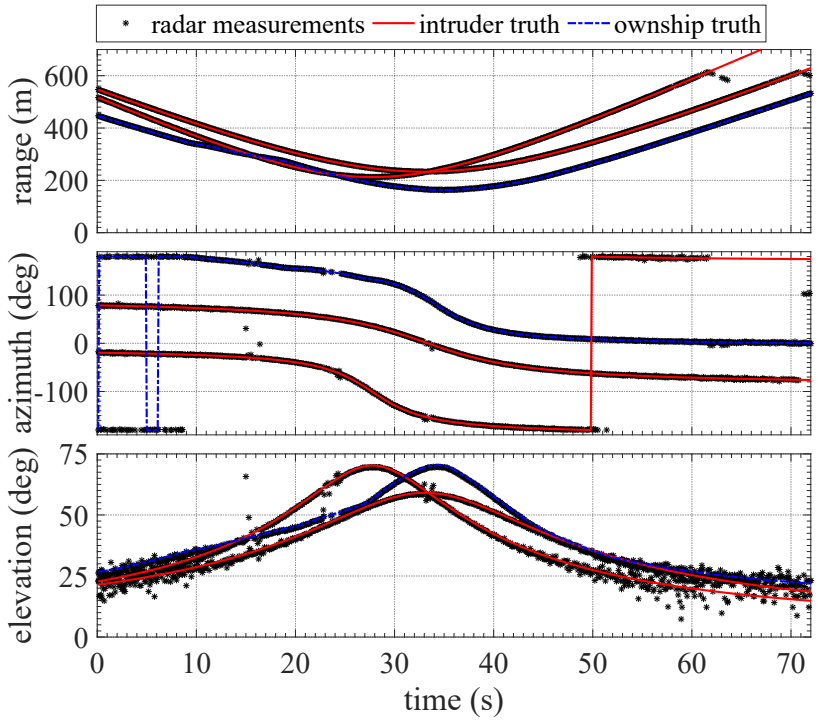
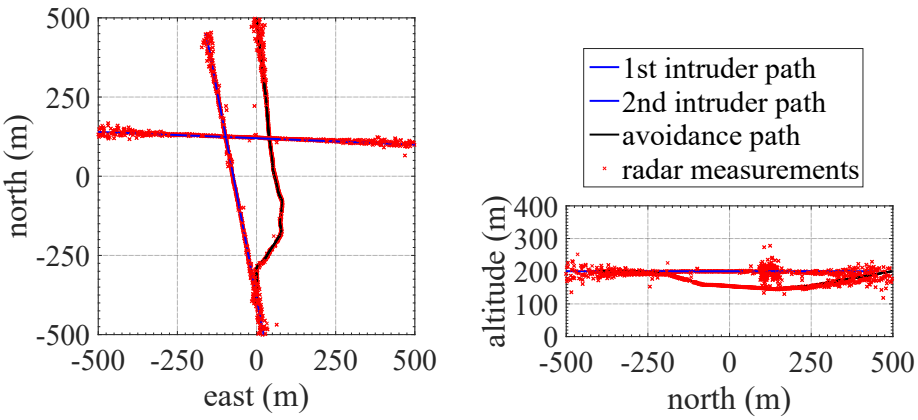


Figure 4.14: Radar measurements: range, azimuth and elevation.



(a) Top view of the radar measurements in the NED inertial frame.

(b) Side view of the radar measurements in the NED inertial frame.

Figure 4.15: Aircraft's paths with radar measurements in encounter scenario number 1.

The state estimates of position and velocity are shown in Figures 4.16 and 4.17 respectively. In this simulation we chose a sample rate of 0.1 seconds and R-RANSAC parameters shown in Table 4.5. The dynamic model used for the three aircraft in R-RANSAC is a constant-acceleration model of the north position, east position, and altitude of the aircraft. From Figures 4.16 and 4.17, we see that this dynamic model worked well within R-RANSAC to successfully track all aircraft for which measurements are received. Additionally from these figures we see that the modifications made to R-RANSAC were successfully able to distinguish the ownship track from the intruder tracks. Furthermore, we see that R-RANSAC takes about 5 seconds to initiate good tracks. This is due to the initial noisy radar elevation measurements, the sample rate at which R-RANSAC is running, and the underlying R-RANSAC design parameters. Both intruder tracks die slightly after we stop receiving radar measurements, due to the aircraft being outside the visible field of view of the radar; however, the ownship's track never dies. We also see that the estimates become more noisy at about 55 seconds due to the noisy elevation measurements that start occurring as the aircraft leave the field of view of the radar.

Table 4.5: R-RANSAC parameters.

Parameter	Value	Parameter	Value	Parameter	Value
$\mathcal{M}$	10	Q	10e-7(10,10,1,10,10,1,10,10,1)	τ_ρ	0.6
N	50	R	10e-3(1,1,1)	τ_T	40
τ_R	$2.5(\sigma_r, \sigma_\alpha, \varepsilon)$	τ_{x_i}	(10,10,10,3,3,3,1,1,1)	τ_{CMD}	15
ℓ	100	$\tau_{x_i}^{\text{own}}$	(20,20,20,10,10,10)	τ_ρ^{own}	0.2

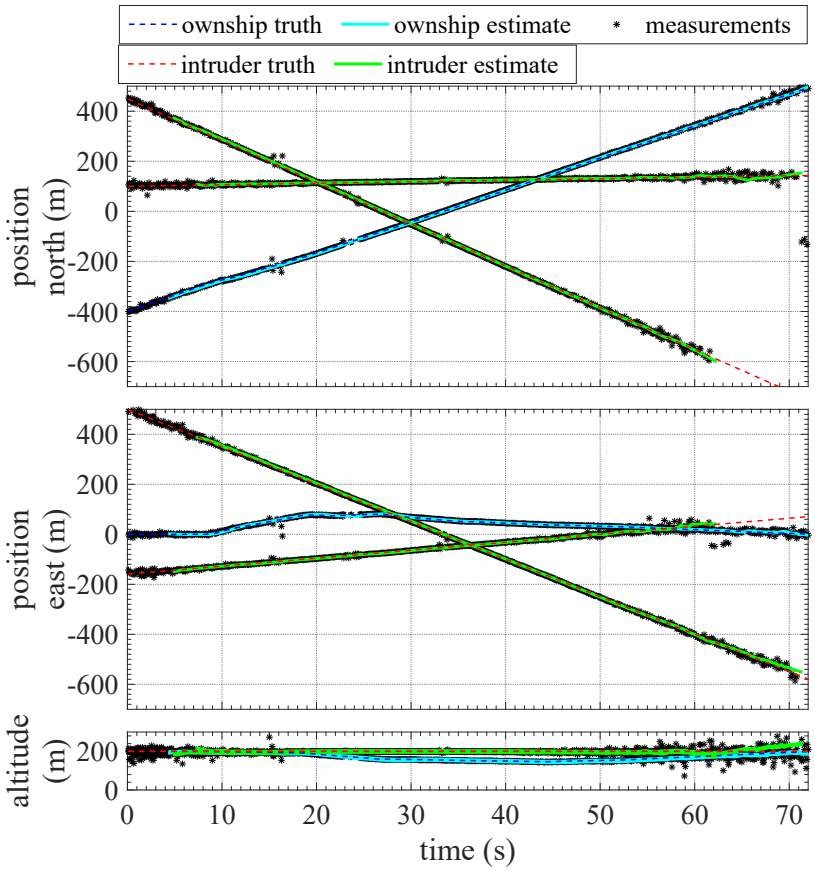


Figure 4.16: R-RANSAC tracks: position estimates of aircraft.

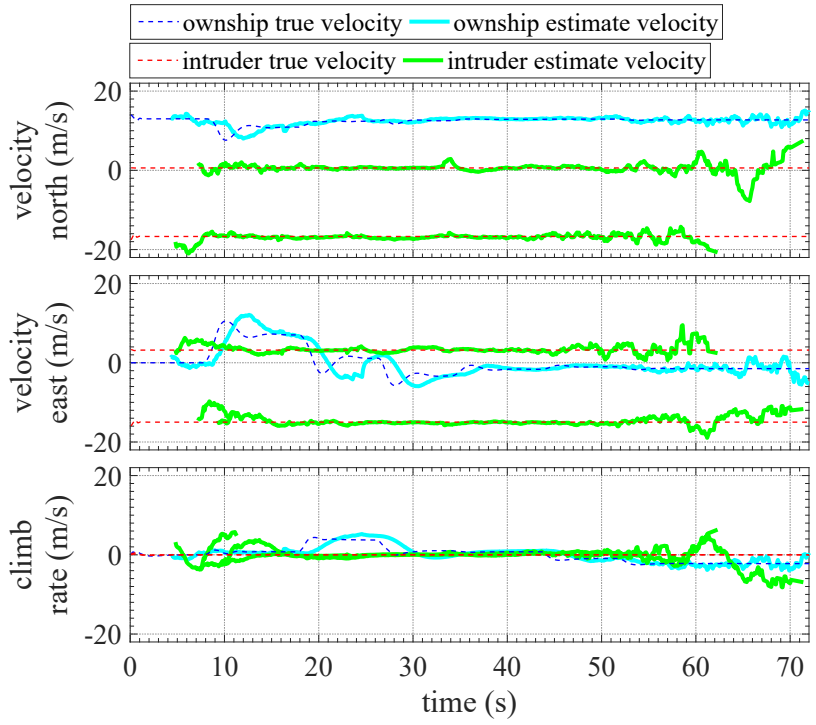


Figure 4.17: R-RANSAC tracks: velocity estimates of aircraft.

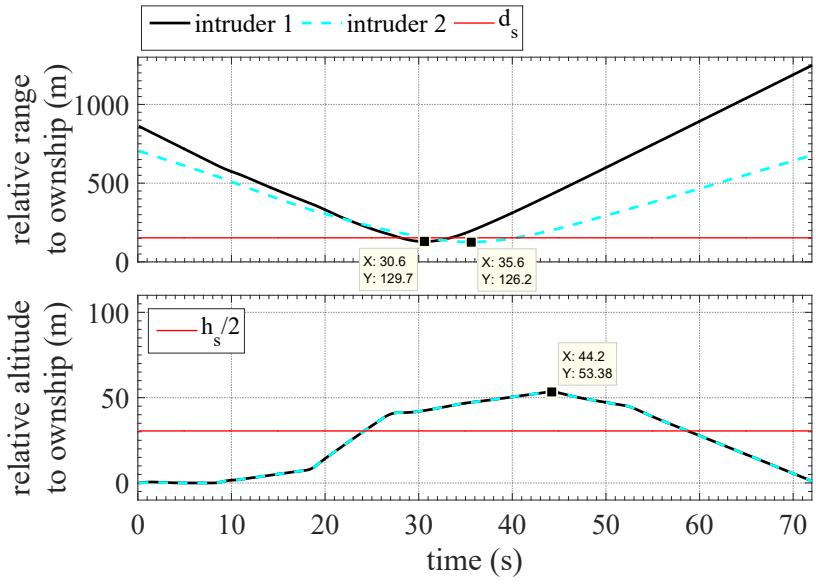


Figure 4.18: Relative range and altitude to intruders.

Figure 4.18 shows the results of the relative range and altitude to both intruders. The relative range to the first intruder falls below d_s between about 28 and 33 seconds, however, the relative altitude remains above $h_s/2$ during that same interval. For the second intruder the relative range is below d_s from about 30 to 40 seconds, and the relative altitude also remains above $h_s/2$ during that time interval. The relative altitude is below $h_s/2$ during the first 24 seconds and after about 59 seconds, however the relative range is above d_s during those same intervals. Based on these observations no collisions occurred between the ownship and two intruders.

In the second encounter geometry, the ownship initially starts at $(-300, 0, -200)^\top$ in NED coordinates system, with an initial heading of 0 degrees measured from north and follows a straight-line path at constant speed of 13 m/s to reach a waypoint located at $(500, 0, -200)^\top$ as shown in Figure 4.19. This encounter scenario consists of four intruders flying at sp e flying at constant

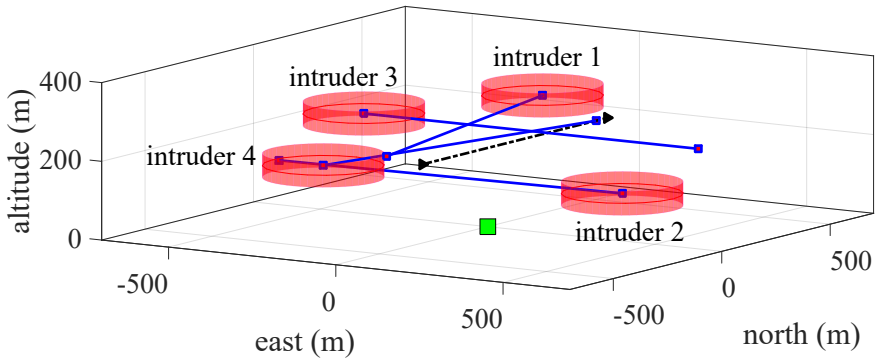


Figure 4.19: Encounter scenario number 2.

The d_{cpa} with respect to the four intruders are approximately 79.9 m, 240.3 m, 0 m and 40 m. Based on these paths, there will be a collision with the third and fourth intruders. The ownship, however, should plan an avoidance maneuver that does not lead to a collision with intruders that were not on a collision course initially. Figures 4.20, 4.21(a) and 4.21(b) show the intruders paths and the avoidance path planned by the ownship.

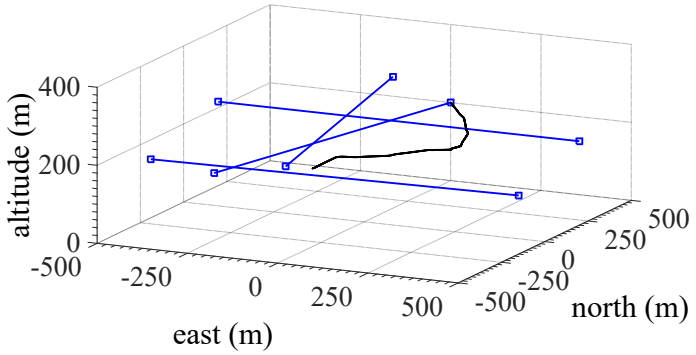
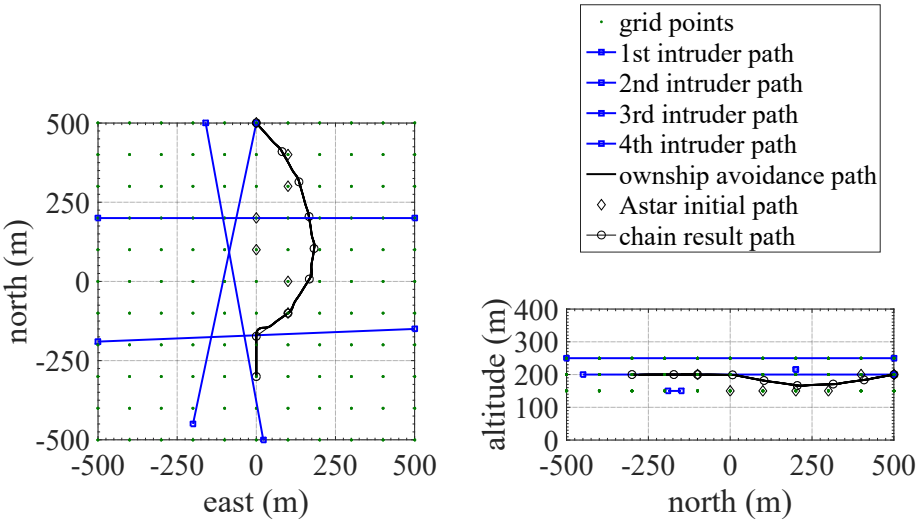


Figure 4.20: The avoidance path of the ownship.



(a) Top view of the ownship's avoidance path.

(b) Side view of the ownship's avoidance path.

Figure 4.21: Avoidance path followed by the ownship in encounter scenario number 2.

Figures 4.22 and 4.23 demonstrate similar results as Figures 4.14 and 4.15. The addition of more intruders does not significantly degrade the radar's ability to detect multiple targets. Figure 4.23(a) demonstrates that the aircraft remain well resolved, even when in close proximity.

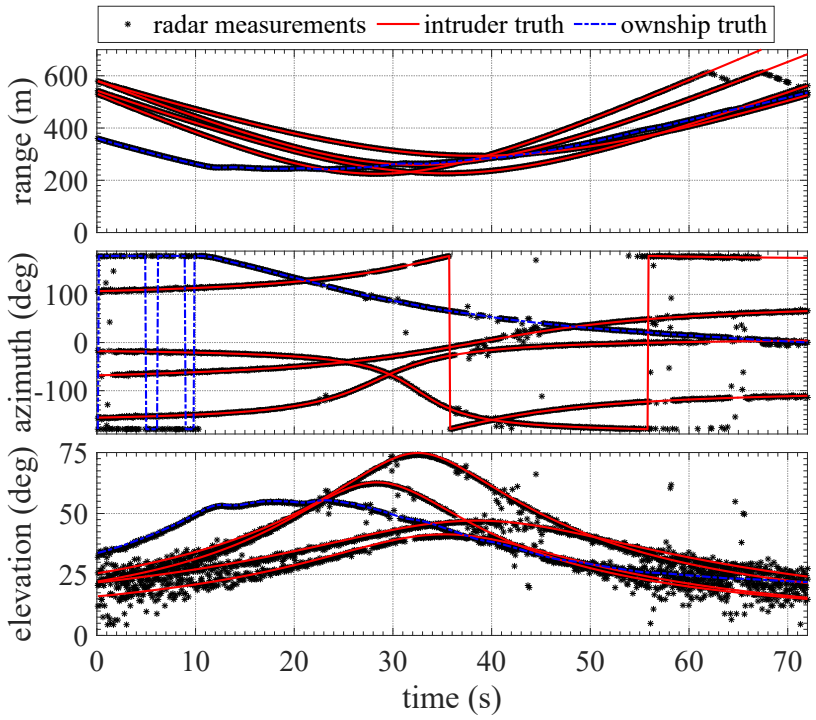
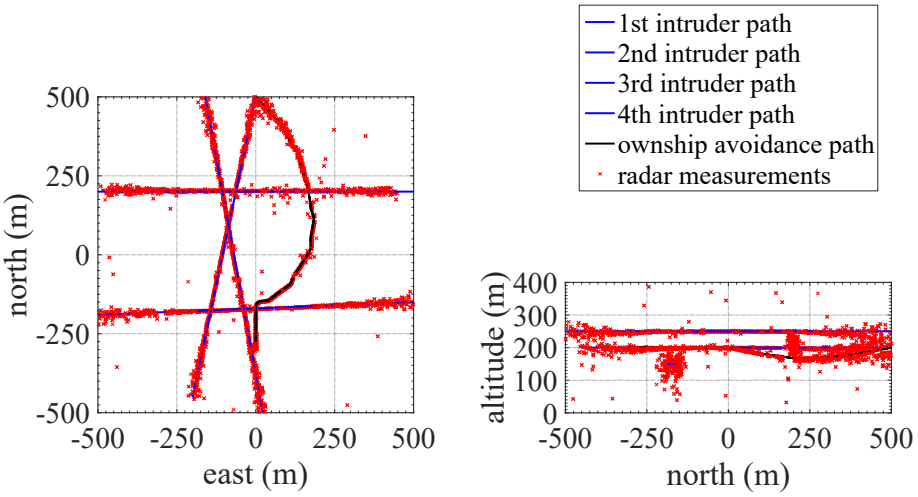


Figure 4.22: Radar measurements: range, azimuth and elevation.



(a) Top view of the radar measurements of the ownship's avoidance path.

(b) Side view of the radar measurements of the of ownship's avoidance path.

Figure 4.23: Aircraft's paths constructed using radar measurements in encounter scenario number 2.

Figures 4.24 and 4.25 show similar results to those in the first scenario, and that R-RANSAC algorithm is able to track multiple intruders. From these figures we also see that one of the intruder tracks is lost at about 45 seconds, but that it is picked up again at about 50 seconds. This is due to missed radar measurements during that same time interval.

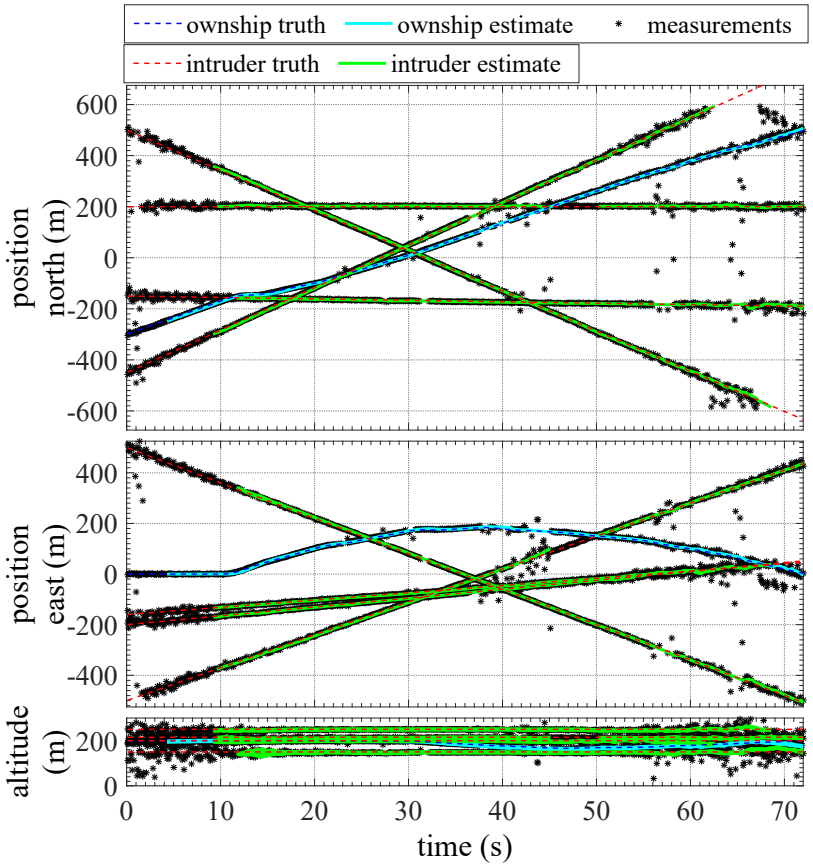


Figure 4.24: R-RANSAC tracks: position estimates of aircraft.

Figure 4.26 shows the results of the relative range and altitude to all intruders. These results show that the avoidance path safely maneuvers the ownship without any collisions with the intruders. In Figure 4.26, the avoidance planner ensures that when the relative horizontal range is less than d_s, the relative altitude is greater than $h_s/2$. For example, the relative range to the third intruder over time interval [41, 55] s is below d_s, however, over the same time interval the relative altitude is above $h_s/2$.

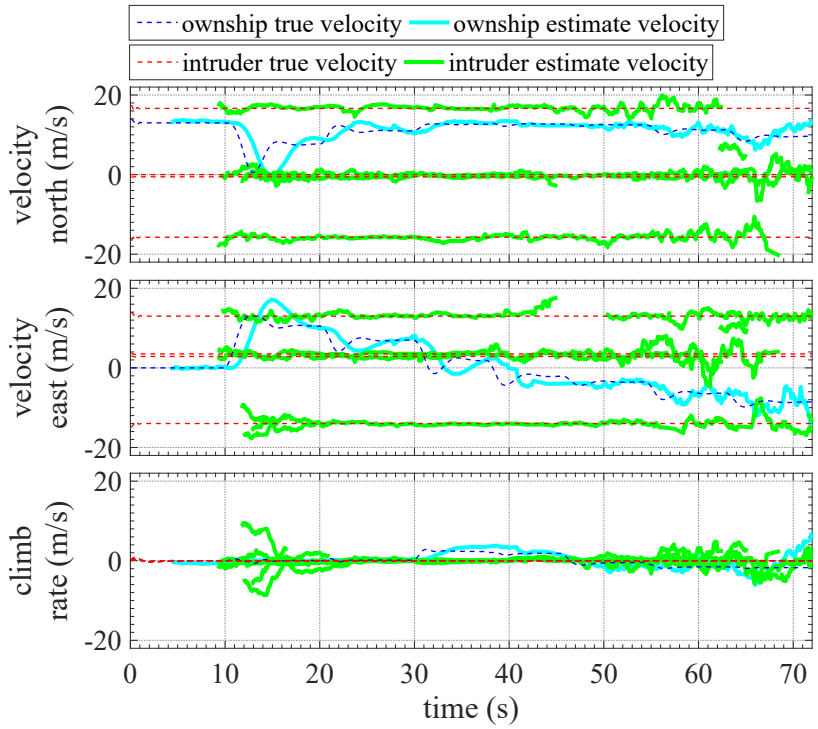


Figure 4.25: R-RANSAC tracks: velocity estimates of aircraft.

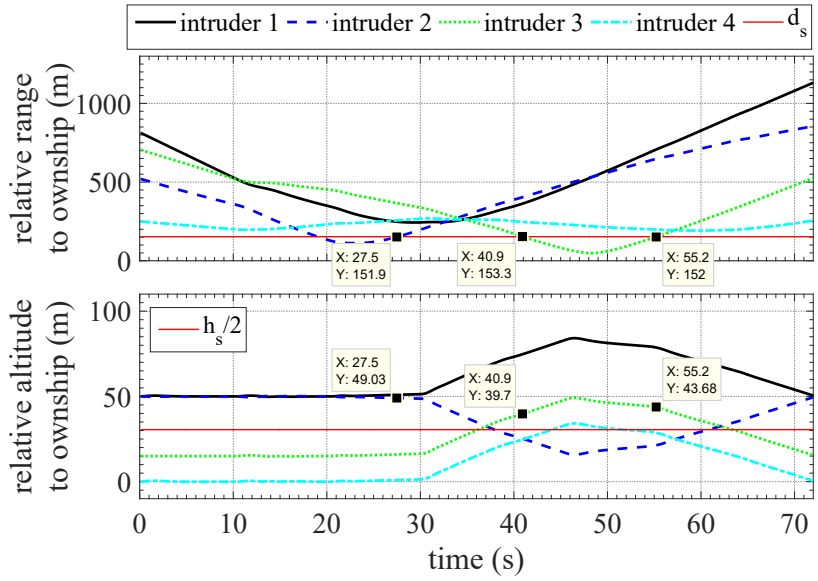


Figure 4.26: Relative range and altitude to intruders.

Another important aspect to evaluate the performance of the proposed algorithm is its ability to reduce the length of the avoidance path while avoiding the intruders. This is important because it reduces the amount of deviation from the original path, and ultimately the flight time, which is of critical importance for the small UAS with limited power resources. Table 4.6 shows that the length of the avoidance paths is fairly acceptable compared to the initial path length.

Table 4.6: Length of the avoidance path.

Scenario number	Initial path length (m)	Avoidance path length (m)
1	900	1383.8
2	800	1296.9

We have also recorded the average and maximum time required to execute the radar measurements processing logic in Table 4.7. In practice these radar processing steps will be run within the FPGA of a the radar processing board, and will not be consuming valuable processing power on the processor which runs the DAA algorithms.

Table 4.7: Radar FPGA run time.

Scenario	1		2	
Algorithm	Average time (s)	Max. time (s)	Average time (s)	Max. time (s)
Radar Processing	0.2273	0.2978	0.3393	0.4346

The run times of the DAA algorithms are also given in Tables 4.8 and 4.9. Table 4.8 shows the average and maximum times for each of the algorithms separately. Table 4.9 shows the average and maximum run times for combinations of the individual algorithms to represent the total DAA time at each time step.

Table 4.8: Run times for detect and avoid algorithms.

Scenario	1		2	
Algorithm	Average time (s)	Max. time (s)	Average time (s)	Max. time (s)
R-RANSAC	0.2384	0.3459	0.3533	0.4441
Collision detection	0.00015	0.00593	0.00211	0.03259
A*	0.1260	0.1260	0.2259	0.2259
Chain	0.0429	0.1603	0.0605	0.1541

Table 4.9: Run times for combined detect and avoid algorithms.

Scenario	1		
Algorithm	Average time (s)	Max. time (s)	N
RR	0.2509	0.3459	39
RR & CD	0.2371	0.2784	523
RR, CD, A* & Chain	0.5643	0.5643	1
RR, CD & Chain	0.2886	0.4384	37
Total	0.2414	0.5643	600
	2		
RR	0.3642	0.4197	76
RR & CD	0.3515	0.4454	475
RR, CD, A* & Chain	0.7677	0.7677	1
RR, CD & Chain	0.4373	0.5418	48
Total	0.3603	0.7677	600

At certain time steps, not every DAA algorithm is executed. At each time step, however, there are four DAA algorithm combinations that will always execute and are shown

by the first four rows in the top and bottom sections of Table 4.9. The last column of this table shows how many times each of these algorithm combinations were executing during the simulation. The average run time for a time step in scenario 1 is 0.2414 seconds, which is about two and a half times longer than real time. The average run time for a time step in scenario 2 is 0.3603 seconds which is about three and a half times longer than real time. These algorithms were run in Matlab 2015b on an Intel core i7-4770 processor. Using the total maximum time from scenario 2, we see that the combined DAA algorithm run time stays within 0.7677 seconds, however, the current implementation is coded in Matlab and was not optimized for run time. Significant speed increases could be achieved by optimizing the code and by porting it to C++.

CHAPTER 5. DAA FLIGHT RESULTS USING RADAR HARDWARE

We now demonstrate the successful implementation of a complete DAA system with real radar hardware in a ground-based setup. As a step in developing a 3D radar sensor that provides measurements of range, azimuth, and elevation, a 2D line array radar system capable of measuring range and azimuth has been developed. Since this radar only provides a single angular measurement of azimuth and not elevation, the altitude of the aircraft is unobservable; therefore, we require that all aircraft fly at a common known altitude. This requires that we modify the DAA algorithms slightly to work at a constant altitude.

The remainder of this section is organized as follows: In Section 5.1 we describe the ground control station that was used in the flight tests. In Section 5.2 we provide more details of the radar hardware and the associated digital signal processing. In Section 5.3 we describe modifications that had to be made to the DAA algorithms as a result of performing the test at a constant altitude. Finally, in Section 5.4 we describe the details of the complete test setup and provide the results of the flight experiment.

5.1 Ground-based Control Station

In a ground-based radar setup, the radar is not physically attached to the ownship aircraft, but is instead located at a ground-based control station. For the ownship to utilize the information gained from the ground-based radar sensor, there must be some sort of wireless transmission of data between the ownship and this ground-based control station. Our particular ground-based control station uses the MAVLink message protocol for this wireless communication over a 3D Robotics (3DR) radio antenna. The transmission and reception of these MAVLink messages is controlled by a Matlab/Simulink model that is located on a laptop control station. This Matlab/Simulink model is also responsible for interfacing with the radar hardware and for running the DAA algorithms including target

detection and tracking, collision detection, and collision avoidance path planning. Each of these components within the Matlab/Simulink model have been organized into four main blocks in an interconnected block diagram as seen in Figure 5.1.

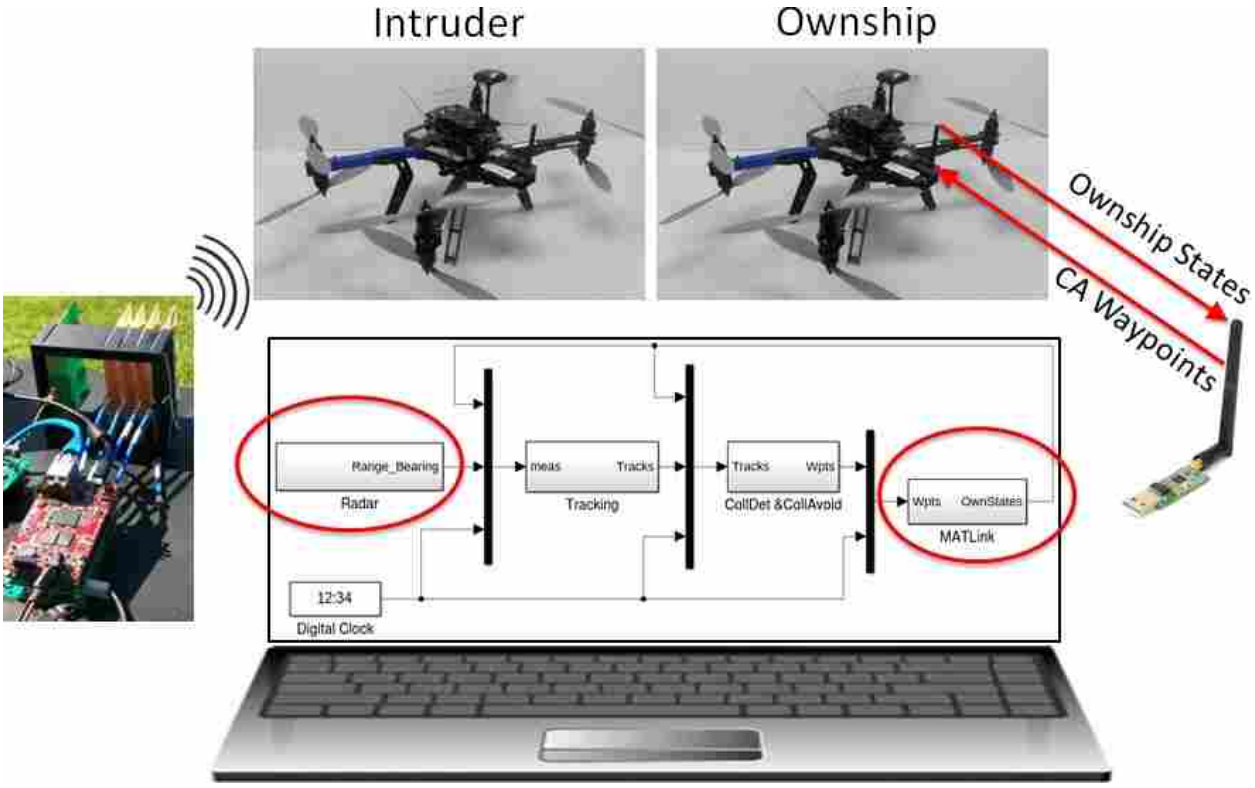


Figure 5.1: Implementation of ground-based radar control station with hardware.

The first Simulink block, located on the left side of the laptop in Figure 5.1, is the block responsible for interfacing with the radar hardware. For this block to receive radar measurements, the radar is first physically connected to the laptop using a wired connection. The information from this wired connection is then read into Simulink using a UDP receive block. The data that is received from the radar hardware includes a large packet of information that is grouped into 32-bit single-precision floating-point words. The first word provides an integer quantity that represents the number of range and azimuth measurement pairs in this current time step. The next two words include the first range measurement and the first azimuth angle measurement. Each of the remaining range and azimuth measurement pairs are then sent within successive two-word packets.

The second Simulink block is the target tracking block which runs a linear version of R-RANSAC. The primary input to this block are the radar measurements that come from the previous UDP block. The R-RANSAC algorithm is able to find targets out of the radar measurements and provide estimated state information for each of these aircraft. In addition to the radar measurements being input in this tracking block, we see that a feedback line that carries the ownship state information is also sent into the tracking block. This information is necessary for the modified R-RANSAC algorithm to identify which tracks belong to the ownship aircraft.

The third Simulink block is the collision detection/collision avoidance block. This block receives inputs from the R-RANSAC target tracks from the previous block and the feedback ownship state information. Inside this block we run the waypoint manager function in addition to the collision detection and collision avoidance algorithms. The output of this block is a comprised of four main components. The first component is a flag that indicates if a new set of waypoints needs to be sent to the ownship. The second component is an integer specifying the total number of waypoints being sent. The third component is composed of the waypoints themselves. The fourth component is an integer specifying the index of the current waypoint the ownship should be flying towards.

The fourth Simulink block is the the block responsible for the transmission and reception of MAVLink messages between the laptop control station and the Pixhawk autopilot located on the ownship aircraft. Specifically the laptop needs to transmit new collision avoidance waypoints to the autopilot, and needs to receive the state information of the ownship from the autopilot. Within this block two sub functions are used. The first function is a waypoint message formatter that gathers all the necessary information needed for a waypoint message to be sent according to the MAVLink waypoint message protocol. The second function is the MATLink S-function block used within Simulink. This function is written and compiled in C++. The primary purpose of this function is to provide a means of interfacing the MAVLink messages that must be sent and received across the 3DR radio with the Matlab/Simulink environment, hence the name MATLink. This MATLink function is a critical piece of the ground-based radar DAA solution because it allows us to utilize the DAA algorithms we have written in Matlab scripts with actual aircraft hardware. This MATLink

function was originally developed by Gary Ellingson, however we have added the capability of sending waypoint messages. In our particular setup we use the MATLink function to send waypoint messages in the correct timing sequence, and to receive the state information of the ownship aircraft that is needed for the tracking and collision detection/collision avoidance algorithms. To send waypoint messages in the correct timing sequence, a couple of considerations must be made. MAVLink messages are sent using a serial communications link. As such each waypoint is sent sequentially, one after another. The transmission of a complete set of waypoints to a Pixhawk autopilot using MAVLink messages requires a fair bit of overhead and logic. After each waypoint has been been sent to the autopilot, the autopilot then sends back an acknowledge (ACK) message that indicates the autopilot has received the current waypoint. If the autopilot does not send back an ACK message, the ground control station attempts to resend the current waypoint until the ACK message from the Pixhawk autopilot is finally received. The MATLink function must also keep track of how many waypoints have been sent. Once the final waypoint has been sent, and an ACK message from the autopilot has been received the MATLink function sets an internal flag indicating that all the waypoints have been sent. This flag is used within the MATLink function to ensure all of the waypoints have been completely sent before attempting to send a new set of waypoints. In addition to sending MAVLink messages, we also use the MATLink function to receive MAVLink messages from the autopilot. Specifically we receive messages that include the states of the ownship aircraft and the current waypoint indicator. This waypoint indicator is an integer that specifies the index of the waypoint that the ownship is currently flying towards, and is used within the collision detection/collision avoidance algorithms.

5.2 10 GHz FMCW Radar and BOARAC Processing Board

The radar hardware that is used in this GBR DAA test was developed by Dr. Karl Warnick and three of his masters students: Jonathan Spencer, Kaleo Roberts, and Michael Boren. Much of the radar processing was also performed by Dr. Doran Wilde and one of his masters students Luke Newmeyer. This radar system is shown in Figure 5.2, and the key design parameters are listed in Table 5.1. Although the test results are given for the ground-based setup, the design parameters were driven primarily for its use as an on-board

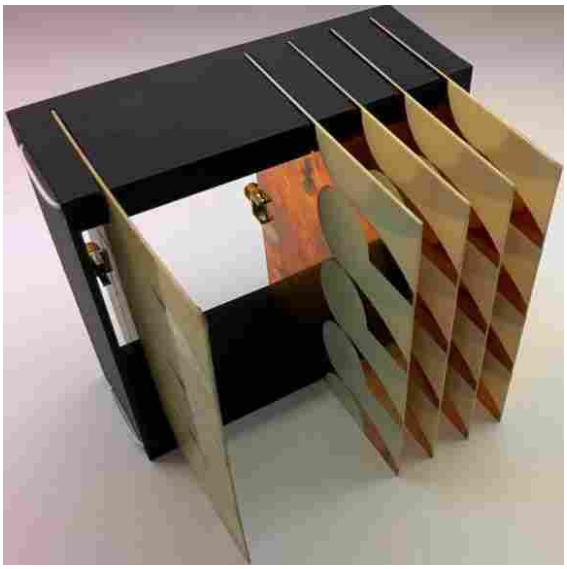
sensor for small UAS. The main design constraints were to minimize SWaP and to provide a reasonably large field of view of approximately 120 degrees horizontally and 30 degrees vertically.

Table 5.1: Radar Sensor Parameters

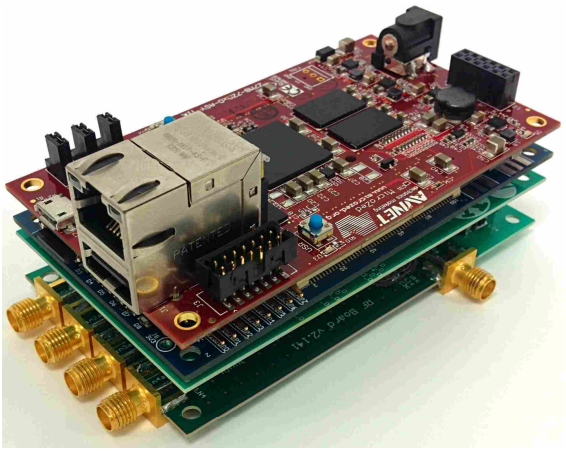
Parameter	**Value**
Weight	120 g (0.26 lbs)
Consumed power	12 W
Center frequency	10.25 GHz
Sweep duration (T_c)	2.048 ms
System noise figure (F)	6 dB
Antenna gain	12 dB
Array steerable range	120 deg
Peak channel coupling	approx. -20 dB
Size	2.25 in x 4 in x 1 in
Transmitted power (P^{rad})	250 mW
Radio frequency bandwidth	500 MHz
Intermediate frequency bandwidth	1 MHz
Antenna elevation beamwidth	18 deg
Antenna azimuth beamwidth	80 deg
Number of receive elements	4
Synthesized azimuth beamwidth	25 deg

This system uses a single transmitting antenna and a four-channel phased-array receiver that scans in the azimuth angle, shown in Figure 5.2(a). The beamwidth of the synthesized beam varies from approximately 25 degrees when steered at boresight to 32 degrees when steered near the edge of the field of view. Since these beamwidths are still quite wide, if two aircraft are within 10 to 15 degrees of each other in the same range bin, the system will be unable to resolve them. Since the range bins are quite narrow, this is a somewhat rare occurrence. Even so, the R-RANSAC tracking algorithm is designed to take into account unresolved measurements of multiple targets.

Since this radar only provides an angular measurement in one direction, we must orient this radar slightly differently than the 3D radar used in the simulation in Chapter 4. The primary states we are interested in estimating are the horizontal north position and east position of each aircraft. Instead of pointing the radar directly upwards into the sky, we



(a) Single transmitting and four-channel phased-array receiving antennas.



(b) Radar system processing boards.

Figure 5.2: Portable phased-array radar system designed for use on-board UAS.

point this radar in a horizontal direction with the azimuth angular measurements going from left to right. We do not require that this radar be perfectly level with the horizontal plane. Instead we angle this radar slightly upwards at a known elevation angle. In addition to the elevation angle of the radar, we also allow the radar to be pointed in an arbitrary heading relative to north, and the radar can be located at any known position and altitude relative to the global grid pattern used by the collision avoidance algorithm. Using this radar setup the output equations resulting from the radar are given by

$$h(\mathbf{x}) = \begin{pmatrix} r(\mathbf{x}) \\ \alpha(\mathbf{x}) \end{pmatrix}, = \begin{pmatrix} \sqrt{(p_n^i)^2 + (p_e^i)^2 + (h^i)^2} \\ \tan^{-1}\left(\frac{p_e^b}{\sqrt{(p_n^b)^2+(h^b)^2}}\right) \end{pmatrix}, \tag{5.1}$$

where the superscript i indicates the state is expressed in the inertial frame of the radar, and the superscript b indicates the state is expressed in the body frame of the radar. The inertial frame of the radar is defined as a right handed north-east-down coordinate frame centered about the radar antenna. The body frame of the radar is defined as a right-handed coordinate frame withe x axis pointing in the same direction as the radar, the y axis pointing out the right side of the radar, and the z axis pointing out the bottom of the radar.

The position vector of an aircraft in the inertial frame of the radar is defined as

$$\mathbf{p}^i = \begin{pmatrix} p_n^i \\ p_e^i \\ -h^i \end{pmatrix} = \begin{pmatrix} p_n - p_{n,r} \\ p_e - p_{e,r} \\ -(h - h_r) \end{pmatrix}, \tag{5.2}$$

where $p_{n,r}$ is the north position of the radar, $p_{e,r}$ is the east position of the radar, and h_r is the altitude of the radar. These three position values are then used to calculate the range to the target.

For the azimuth angle, expressions must now be developed for the position vector of the aircraft in the body frame of the radar. To do this we must use the two-element rotation matrix that transforms a vector from the inertial frame to the body frame as

$$R_i^b(\theta, \psi) = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ -s_\psi & c_\psi & 0 \\ s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}, \tag{5.3}$$

where θ is the elevation of the radar, and ψ is the heading of the radar measured relative to north. Using this rotation matrix we can now express the the position vector of the aircraft in the body frame of the radar as

$$\mathbf{p}^b = \begin{pmatrix} p_n^b \\ p_e^b \\ -h^b \end{pmatrix} = R_i^b(\theta, \psi)\mathbf{p}^i. \tag{5.4}$$

For radar processing, this system uses four boards shown in Figure 5.2(b). From bottom to top the boards are: radio frequency (RF) transmitter and receivers, Intermittent frequency (IF) amplification and filtering, A/D converters, and digital signal processing and control board. The bottom two RF and IF boards were developed by the group of students under the direction of Dr. Karl Warnick. The third A/D board is called the BOARAC board and was developed by Dr. Doran Wilde's students. The top board is a commercial off-the-shelf development board called MicroZed. This board contains a Xilinx Zynq All

programmable system-on-chip (SoC) that includes a field-programmable gate array (FPGA) and microprocessor. All of the DSP steps are performed on this Zynq SoC, where the specific implementation of each of these DSP steps resulted from a combined effort of the radar team and the the research performed in this thesis.

The DSP steps that are implemented on the FPGA of the Zynq SoC include the fast-Fourier transform (FFT), correlation, and averaging. The DSP steps that are implemented on the microprocessor of the Zynq SoC include target range detection and digital beamforming. Eventually, these last two DSP steps should be moved to the FPGA, and the microprocessor should be primarily used for the DAA algorithms including target tracking, collision detection, and collision avoidance.

The BOARAC board includes eight 12-bit A/D converters, however, currently only four of them are being used for the four receiver channels. The output of these four A/D converters are sent from the BOARAC board, and into the FPGA of the Zynq SoC on the MicroZed board. Using these four A/D inputs, the FPGA performs an FFT that retains 16 bits of accuracy at its output. Next the correlation step is performed, which is essentially squaring the result of the FFT. Since this value is getting squared, 32 bits of accuracy are then retained. Next the correlated values are averaged from multiple time steps to increase the SNR. To preserve additional data resulting from this correlation step, 37 bits of accuracy are then retained. Finally, these 37-bit numbers must be truncated to 32 bits before we can send them to the microprocessor. Pairs of these 32-bit values must then be created because the microprocessor is expecting to receive 64-bit messages. Within the microprocessor, the 32-bit input values are then stored as single-precision floating-point numbers. The last two DSP steps performed in the microprocessor have various contributions resulting from research in this thesis and will now be described.

The first contribution has to do with the thresholding shown previously in Equation (4.4) by $T_i = \frac{\ln(1/P_{FA}^i)}{\lambda_i}$, where i represents the index for each range bin. For the simulation results in Chapter 4, we used a constant value for the probability of false alarm. For the hardware results, however, we found it to be advantageous to use a variable P_{FA}^i. For this variable false alarm probability we have defined an inverted quadratic function that is smaller at the close range bins and bigger at the far range bins as seen in Figure 5.3. Having

a larger P_{FA}^{i} at the far range bins allows us to detect small signals out of the noise floor more easily. We found this to be necessary because the received power of a signal drops of as $1/r^4$, where r is the range to the target. The specific quadratic equation we used for this variable probability of false alarm is given by

$$\begin{aligned} P_{FA}^{i} &= a(i-b)^2 + c, \\ &= \frac{-(P_{FA,f} - P_{FA,n})}{i_{max}^2}(i - i_{max})^2 + P_{FA,f}, \end{aligned} \tag{5.5}$$

where $P_{FA,n}$ is the probability of false alarm at the near range bin, $P_{FA,f}$ is the probability of false alarm at the far range bin, i is the index of the range bin, and i_{max} is the maximum range-bin index.

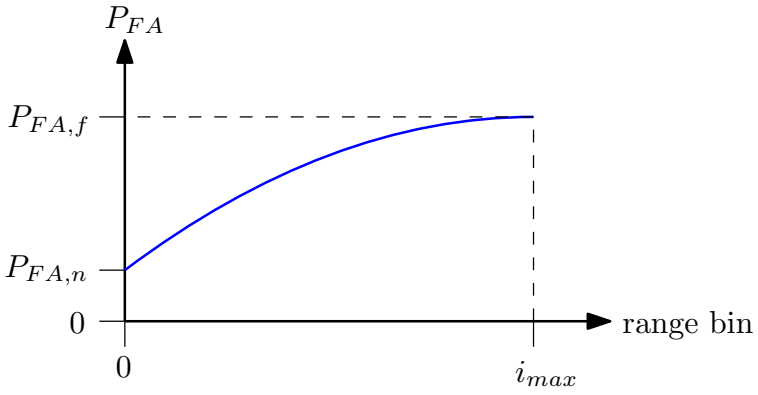


Figure 5.3: Variable probability of false alarm as a function of range bin.

The next contribution has to do with target range detection. One issue with the current radar hardware is that close objects to the radar cause overflow in the A/D converters, which then results in harmonic range detections out of the FFT. Essentially this means that a single target will show up in multiple range locations that are evenly spaced from one another. The solution that was developed to overcome this issue was to require two or more measurements in adjacent range bins to rise above the threshold. If a group of two or measurements does indeed rise above their respective thresholds, then we conclude that a target has been detected within this range-bin group. This method is able to reject harmonic

range measurements because these harmonic measurements always appear in a single range bin and therefore do not meet the requirement to be identified as a target. This logic also has the added benefit of reducing the amount of noise measurements received from the radar.

As a result of this requirement, groups of measurements are now expected to be received. Instead of outputting each of these elements as separate measurements, we have also created a system that groups these measurements into a smaller number of outputs. The first step in this process is to identify each of the groups of measurements. An example of this grouping is seen in Figure 5.4 by the groups of points that have been circled.

In this figure, we are plotting the received power for each range bin at one specific instant in time.

Also shown is a generic threshold line for each range bin and a single measurement that is not part of any group because it does not have any measurements in adjacent range bins. This measurement is depicted with a red $\times$ over it.

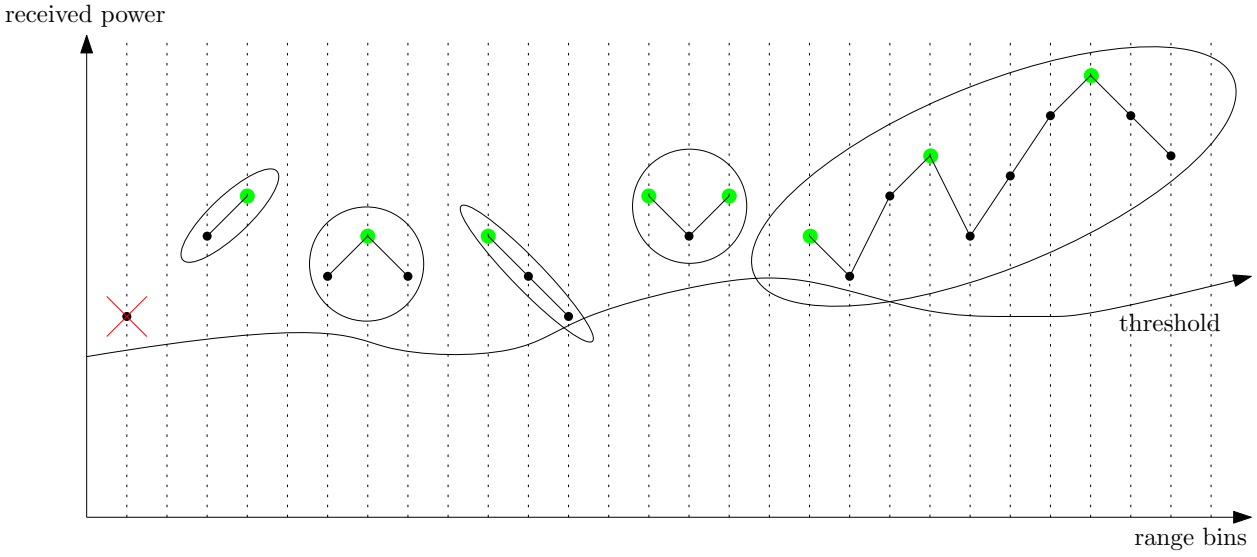


Figure 5.4: Grouping and peak identification.

After each of the groups has been identified we then perform a process of peak identification. In this process we locate all measurements that are larger than the measurements directly adjacent to them. These peaks are depicted with a green point. This process of peak identification has been used to help detect multiple targets that may be in relatively

close range bins, but are distinct enough to create separate peaks. The peaks from each of the groups are then used as the range indices where we conclude that a target is located.

The next contribution has to do with the results of target range detection and digital beamforming. In both of these methods discrete range and angle bins are identified, which results in all measurements falling within predefined circular grid locations. These DSP steps are unable to provide measurements of a target in between range and azimuth angle bins. Research efforts in this thesis have been able to overcome this limitation and will be shown next.

Once the target range detection and digital beamforming bin indexes have been calculated, we notice that the magnitude of the signal in this particular bin is larger than the magnitude of the signals in the bins immediately before and after the current bin. This is seen in Figure 5.5, where the bin index with the largest magnitude is identified as $\mathcal{B}$, the bin index to the left is identified as $\mathcal{B}-1$, the bin index to the right is identified as $\mathcal{B}+1$, and the corresponding magnitudes are identified as $\mathcal{M}_C$, $\mathcal{M}_L$, and $\mathcal{M}_R$, respectively.

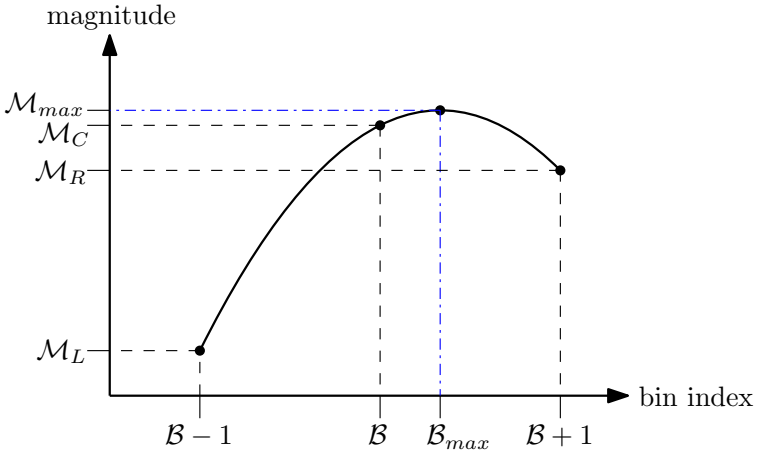


Figure 5.5: Spline fitting.

By using these three points, we can fit a second-order polynomial. From this polynomial we notice it has a maximum $\mathcal{M}_{max}$, and that this maximum corresponds to a bin index location in between two integer bin indices as $\mathcal{B}_{max}$. The equation for this fractional

bin index is shown as

$$\mathcal{B}_{max} = \mathcal{B} + \frac{\mathcal{M}_L - \mathcal{M}_R}{2(\mathcal{M}_L - 2\mathcal{M}_C + \mathcal{M}_R)}. \tag{5.6}$$

This fractional bin index is then used to update the bin indices previously found for the range and azimuth angles.

The method of fitting a second-order polynomial to three data points does quite well at finding appropriate angles in between angle bins, however, finding appropriate ranges in between range bins requires additional processing. As an object passes across multiple range bins at a constant velocity, the new data points are not uniformly spaced apart, but instead lie close to the original discrete range bin. By collecting a large sample of range data and calculating the difference between the range value and its associated discrete range bin, we see the general clustering behavior of many range bins from the histogram in Figure 5.6. For the data in this figure, a target moved through approximated 120 range bins at a slow constant velocity, for a total data sample size of 1300 measurements. In this figure we notice that the x-axis ranges between -0.5 and 0.5. This is because if the difference was larger than 0.5 or less than -0.5, then the measurement would show up in a different bin index.

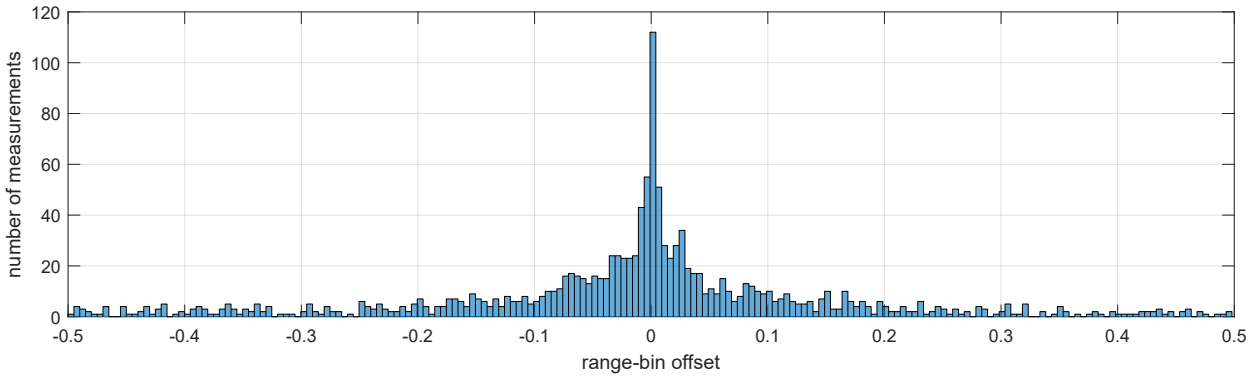


Figure 5.6: Histogram of x_{diff}.

In Figure 5.6, we see that the updated range-bin indices have clustered around the original range-bin index in roughly a double sided exponential distribution centered about zero. Since the object used to collect this data was moving across multiple range bins at a constant velocity, we need to provide a method that spreads this data away from the

discrete range bin, into a uniform distribution. To help gather the needed parameters for this operation we first take the absolute value of these difference values to create a positive exponential distribution, as shown in Figure 5.7.

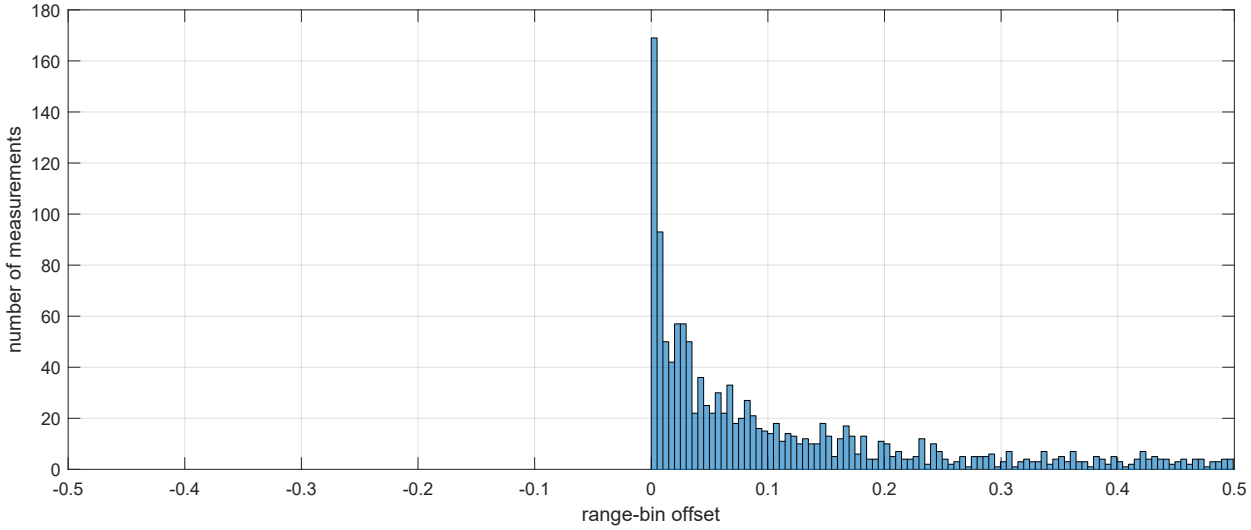


Figure 5.7: Histogram of the absolute value of x_{diff}.

The first step needed, is to convert this exponential distribution into a symmetric beta distribution ranging between 0 and 0.5 using

$$x_2 = \frac{1}{2}(1 - e^{-x_1\lambda}), \tag{5.7}$$

where x_1 represents the exponential distribution, x_2 represents the symmetric beta distribution, and λ is the exponential parameter that is equal to the inverse of the expected value of the distribution. Using $\lambda = 9.18$ in Equation (5.7) results in the symmetric beta distribution shown in Figure 5.8.

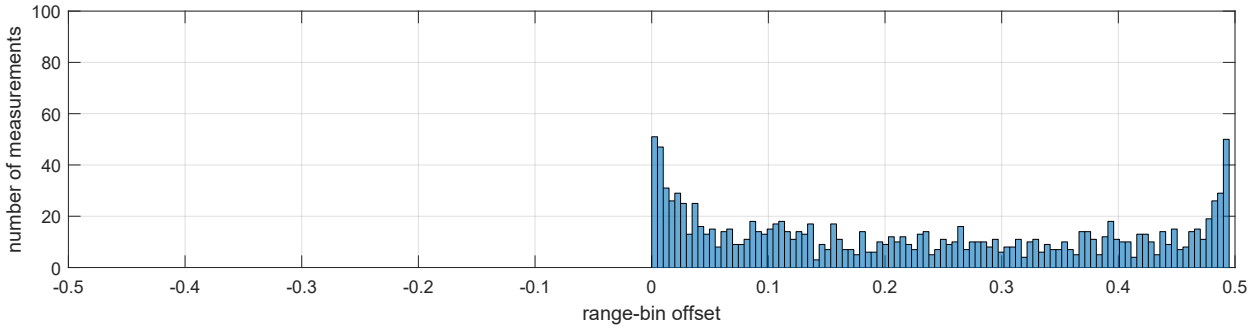


Figure 5.8: Histogram after converting the exponential distribution into a symmetric beta distribution.

The next step is to convert this beta distribution into a uniform distribution ranging between 0 and 0.5. To do this, we use interpolation within a lookup table that converts the beta distribution into a uniform distribution. The values for this lookup table come from the cumulative distribution function (cdf) of the beta distribution. The cdf of the beta distribution is found by integrating the pdf of the beta distribution shown as

$$f_x(x) = \frac{1}{B} x^{\alpha-1}(1-x)^{\beta-1} \tag{5.8}$$

where $f_x(x)$ is the pdf function, B is the beta function coefficient, and α, β are the beta function parameters. Using $B = 3.1871$, and $\alpha = \beta = 0.6$ in Equation (5.8) results in the lookup table shown graphically in Figure 5.9.

The input to this lookup table are the values from the beta distribution shown on the abscissa, and the output of this lookup table are the resulting values of the uniform distribution found on the ordinate. By using interpolation in conjunction with this lookup table, we get the uniform distribution shown in Figure 5.10.

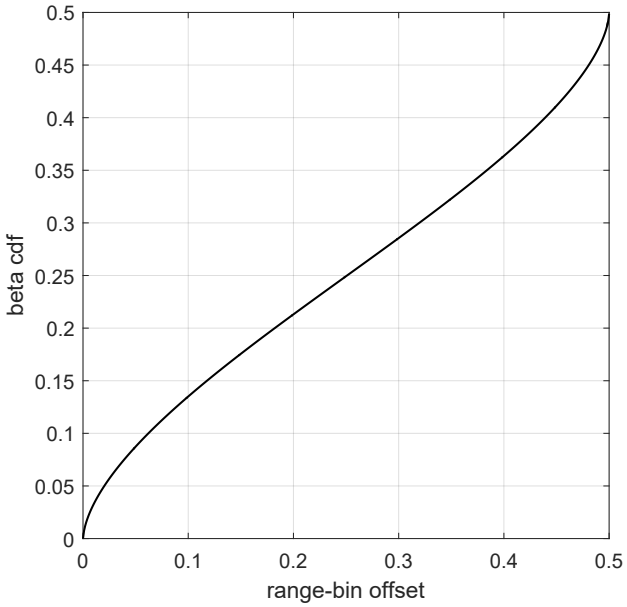


Figure 5.9: Beta function cdf.

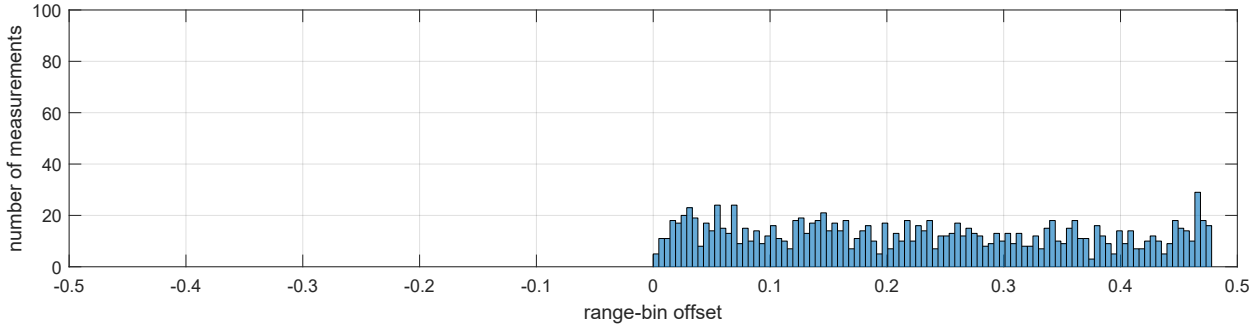


Figure 5.10: Histogram after converting the beta distribution into a uniform distribution.

The final step is to relocate the data points that were originally negative back to the negative side of the plot. This is shown in Figure 5.11. By comparing Figures 5.6 and 5.11 we see that we have successfully transformed the double-sided exponential distribution centered about a particular range bin into a uniform distribution about that same point. The operations we have developed provide a means of evenly distributing range measurements in between range bins, which will then be sent to the R-RANSAC tracking algorithm.

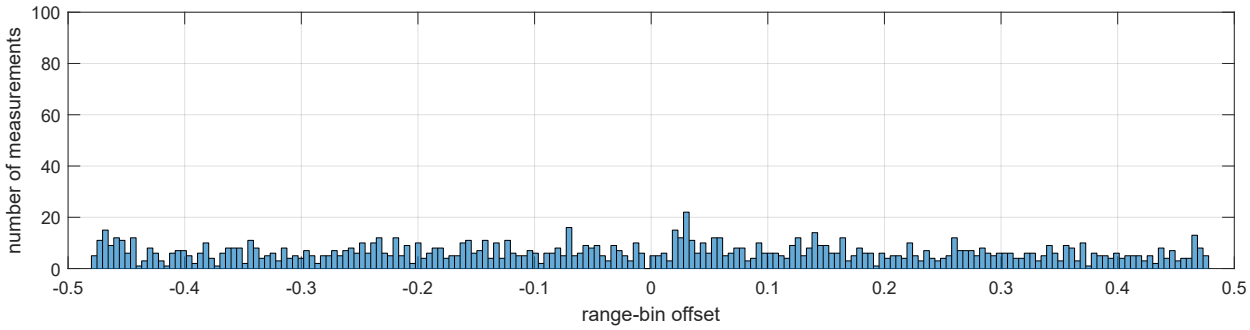


Figure 5.11: Final uniformly distributed histogram.

5.3 2D Modifications of DAA Algorithms

As was stated previously, the DAA algorithms must be modified slightly to operate at a constant altitude. In this section we give brief overview of the changes that had to be made for each DAA algorithm.

The primary change that needed to be made within R-RANSAC was in the states being estimated. Since the altitude in no longer observable from the radar measurements, we must exclude these states, which results in

$$\mathbf{x} = [p_n, p_e, \dot{p_n}, \dot{p_e}, \ddot{p_n}, \ddot{p_e}]^\top. \tag{5.9}$$

In addition to this change in states, the output equation resulting from the new ground-based radar setup must also be used shown in Equation (5.1).

For the collision-detection algorithm, we use the same algorithm as the 3D case, however, we supply the algorithm with the known altitude of the aircraft, and the vertical velocity of each aircraft is set equal to zero.

For the collision-avoidance algorithm, we use the same algorithm as the 3D case. For this collision-avoidance algorithm, however, we create a grid of flyable waypoints that only includes points from the nominal altitude. This prevents the avoidance path from choosing waypoints at varying altitudes.

5.4 Hardware Results

For the DSP of the radar signal for this flight test we sampled 4096 times at a sampling frequency of 2.048 MHz. To increase the SNR, we averaged correlation matrices across 32 time steps. For the background subtraction we collected 400 noise only measurements using a variable probability of false alarm from 0.03 at the near range bins to 0.15 at the far range bins.

Figure 5.12: 3DR X8 and Y6 aircraft used in flight test with corner reflectors.

To demonstrate the performance of the proposed ground-based radar sensor model, R-RANSAC estimation scheme, collision detection, and collision avoidance algorithm, we conducted a flight test using 3D Robotics X8 and Y6 multicopter airframes for the ownship and two intruders, with an attached corner reflector for greater detectability by the radar as seen in Figure 5.12. During this flight test, the top radar processing board was only used for digital signal processing. In the future, however, this board will also run the tracking,

collision detection, and collision avoidance algorithms. From the radar processing boards, the radar measurements were sent to a laptop, which served as a ground control station and which also ran the tracking, collision detection, and collision avoidance algorithms. Since the current radar hardware is only capable of providing angular measurements in the azimuth direction, the altitude of the tracked targets is unobservable and we therefore perform the test at a constant altitude. An explanatory sketch of the flight experiment is depicted in Figure 5.13. In this figure we see that the ground control station and radar are located on a platform that is above the ground by 1.83 meters, and the radar is angled up by 3 degrees. We also see that each of the aircraft are flying at a constant altitude of 4.5 meters, and that each of the aircraft start roughly 50 meters apart from each other.

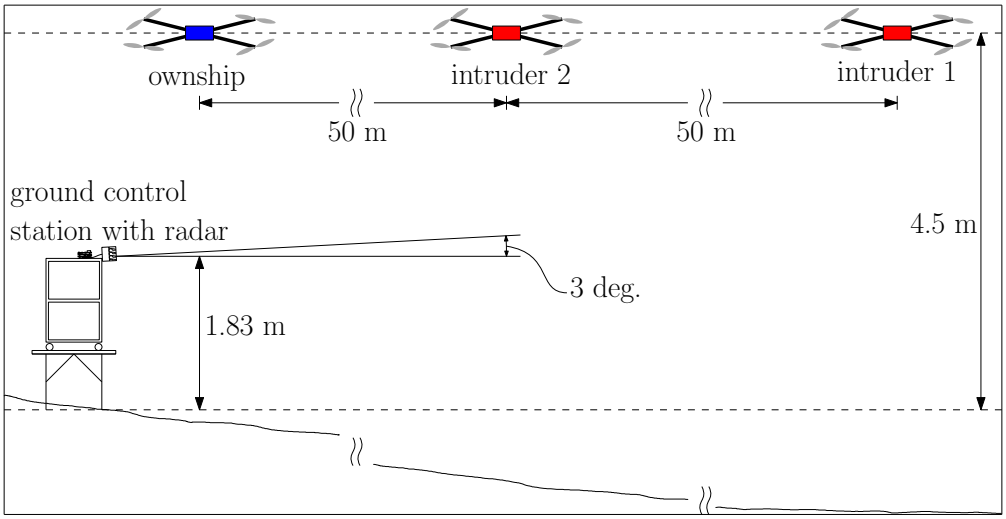


Figure 5.13: Sketch of the encounter geometry of flight test (not to scale).

For this test we have defined the paths of the ownship and intruders in the NED reference frame, however, the 2D plots have been shown in the forward-right(FR) reference frame relative to the heading of the radar. This FR reference frame comes from rotating the inertial NED reference frame about the down axis by the heading angle of the radar unit. We have shown the 2D plots in the FR reference frame because the radar was oriented with a heading of approximately 73.8 degrees relative to north due to the physical location of the flight test. The FR reference frame is also used as the reference frame within which the

collision avoidance paths will be shown. The final parameters needed for the setup of the flight experiment are the GPS home location, and the location of the ground control station and radar with respect to this home location.

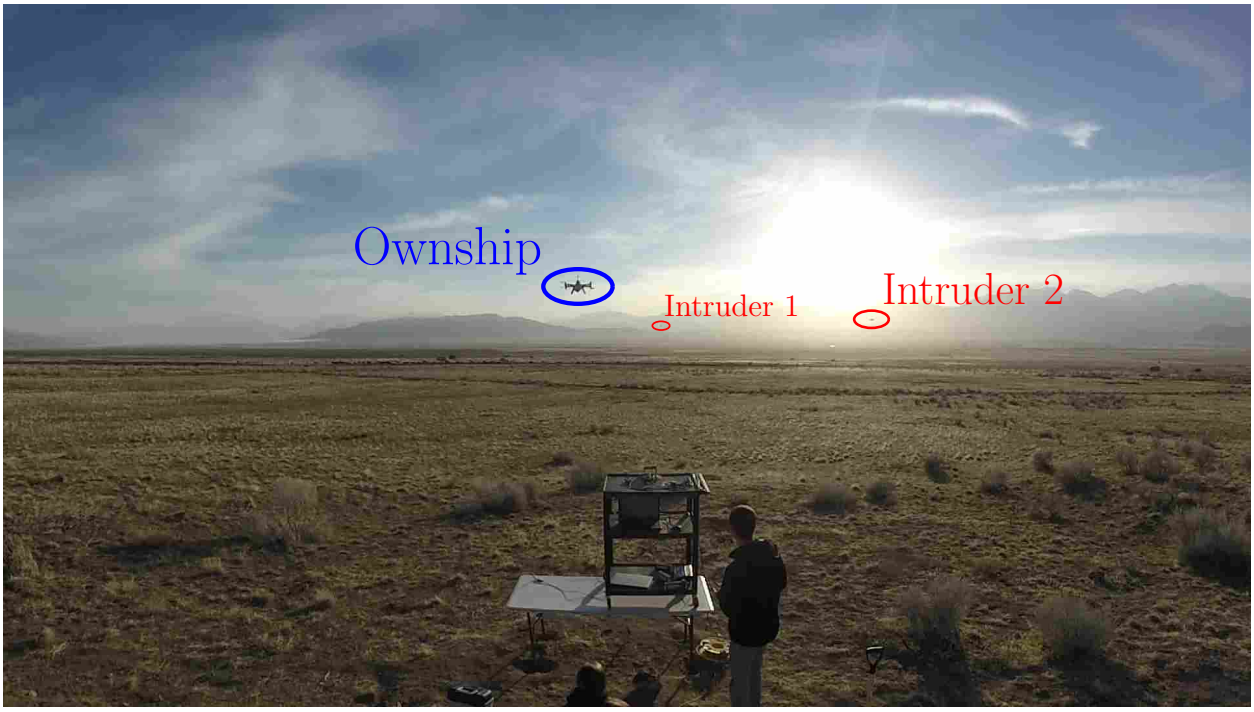


Figure 5.14: Ground control station with radar and three aircraft at the start of the test.

A photograph taken at the start of the test is shown in Figure 5.14. Within this figure, labels have been added to help identify the ownship and two intruder aircraft. The actual geometry of the planned encounter scenario is shown in Figure 5.15 by the black and blue lines, with the starting location of each aircraft shown by an $\times$.

In the encounter scenario, the ownship starts at $(1, 0)^\top$ in the FR radar coordinate system, with an initial heading of 0 degrees measured from the forward direction and follows a straight line path at a constant speed of 1 m/s to reach a waypoint located at $(101, 0)^\top$. Although the location of the ground control station is not shown, it is located at $(-2, 0)^\top$. Since our test is limited to 2D, the collision volume is also converted to 2D by using a circle of radius d_s=10 m (32.8 ft).

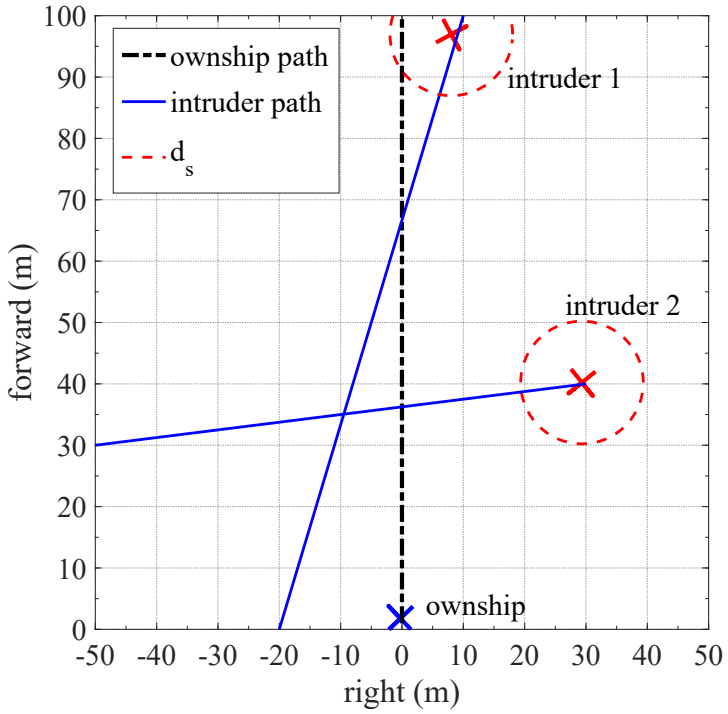


Figure 5.15: Encounter geometry of ground-based DAA flight test.

In this flight test, we follow a similar encounter scenario as scenario 1 in the simulation results section of Chapter 4. It consists of two intruders: one is approaching head-on and the other is converging from the right. These intruders follow straight-line paths at a constant velocity of 1 m/s. If no collision avoidance path is planned, the d_{cpa} with respect to the first and second intruders is approximately 8.0 m and 27.0 m. Since the d_{cpa} to the first intruder is less than the horizontal safety distance d_s, then this encounter will lead to a collision. The ownship, however, should plan an avoidance maneuver that does not lead to a collision with the second intruder. Figure 5.16 shows the intruders' paths and the avoidance path taken by the ownship. Notice that all avoidance must be done in the horizontal plane since we assumed all aircraft are flying at a constant known altitude, unlike the simulation results presented earlier in Chapter 4.

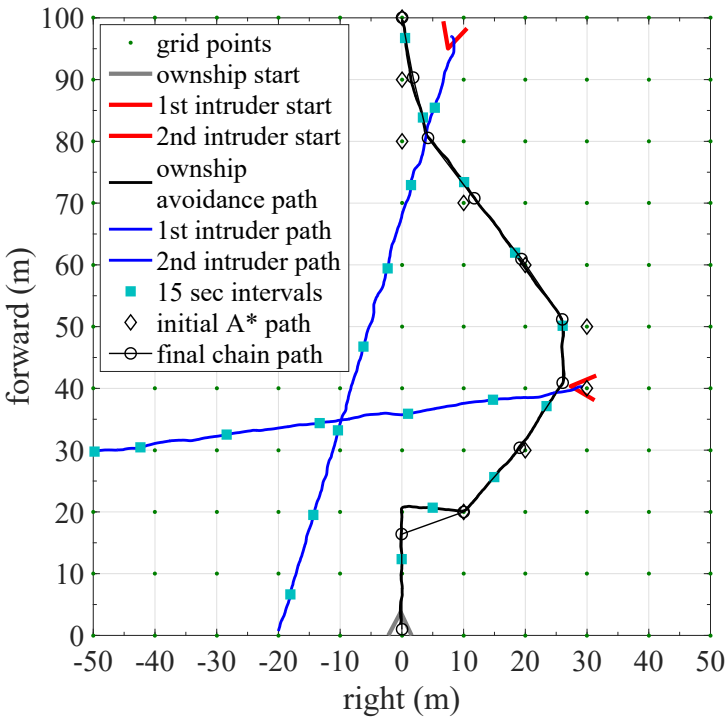


Figure 5.16: The avoidance path of the ownship and the paths of the intruders in the FR inertial frame centered about the home location.

Table 5.2: Collision avoidance parameters.

A*		Chain	
Parameter	Value	Parameter	Value
k_{dist}	10	k	1
$k_{\max}$	1e8	k_{sh}	10
ϱ	4	κ	2
k_r	1000	b	2
		$\theta_{\max}$	45 deg
		λ	20
		$f_{\max}$	50
		δ_1	4

The parameters used for collision detection in the flight test were d_{th}= 100 meters, and τ_{th}= 10 seconds. For the collision avoidance path planning algorithms a grid size of 10 meters horizontally was used. The parameters used for collision avoidance are shown in Table 5.2.

Figure 5.17 shows the range and azimuth angle to all aircraft measured by the radar system. This figure shows that the principal signal decay happens as the azimuth angle increases off boresight as the aircraft leave the antenna beam. The range data, however, remains well resolved for the duration of the test. From this figure we also see that the measured range has a slight scale factor error compared to the true range to each of the aircraft, which could be corrected through further calibration and testing. We also see that the azimuth angle does not line up perfectly with the true azimuth angle, which is due to multiple issues.

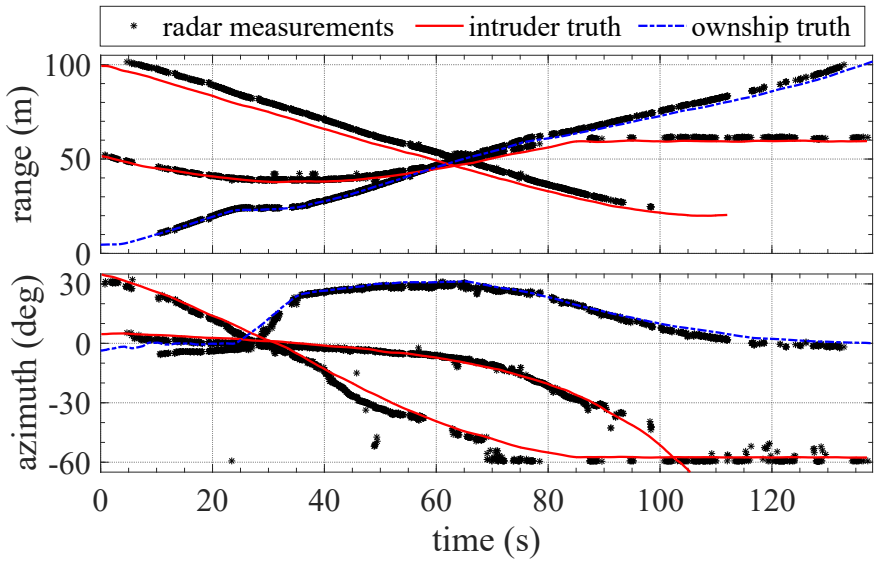


Figure 5.17: Radar measurements: range, and azimuth of ownship and intruders.

As explained earlier, during the flight test we attempted to orient the radar with a precise heading relative to north, and a specific elevation angle. We also attempted to position the radar at a specific GPS latitude and longitude location and altitude relative to the ground. Although we attempted to position and orient the radar with specific values,

errors were unavoidable that resulted in inaccuracies in our measured values when compared to truth. For ground truth data, we used the estimated GPS positions of the three aircraft from the Pixhawk autopilot state estimators that also had inaccuracies due to IMU sensor, GPS, and state estimator errors. These errors are further seen in Figure 5.18, which shows the radar measurements of each aircraft in the FR coordinate frame. Considering all possible sources of error, the radar measurements and truth line up quite well, although future calibration and testing would be beneficial.

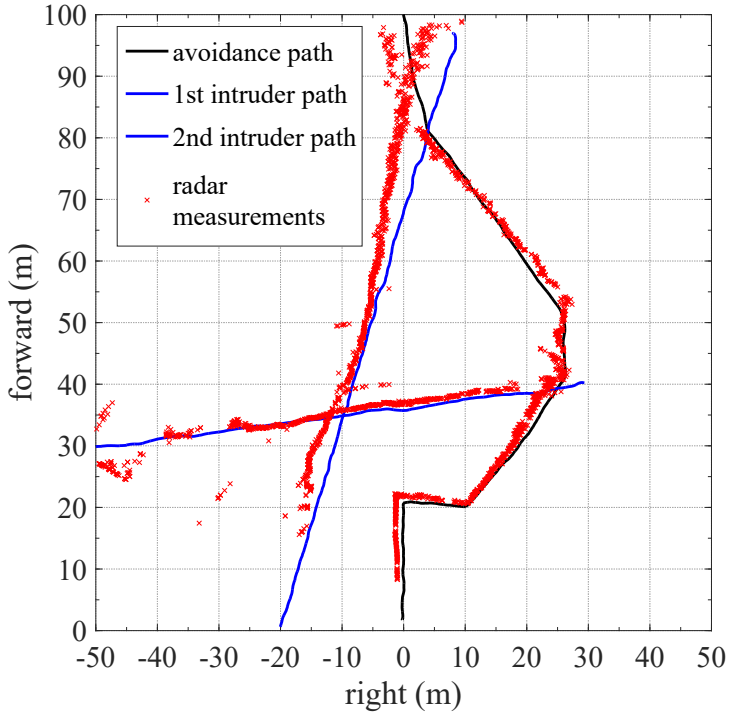


Figure 5.18: Aircraft's paths constructed using radar measurements.

The state estimates of position and velocity are shown in Figures 5.19 and 5.20, respectively. In this flight test we chose a sample rate of 0.1 seconds and R-RANSAC parameters shown in Table 5.3. The dynamic model used for the three aircraft in R-RANSAC is a constant acceleration model of the north position, and east position of the aircraft. Similar to the simulation results in Chapter 4, we see in Figures 5.19 and 5.20 that this model works within R-RANSAC to successfully track all aircraft for which measurements are received. Additionally, from these figures we see that the modifications made to R-

RANSAC were still successfully able to distinguish the ownship track from the intruder tracks while receiving the ownship states from the telemetry link via an 3DR radio. In the figures we see that R-RANSAC takes about 5 seconds for good models to appear after the first measurements are received.

Table 5.3: R-RANSAC parameters.

Parameter	Value	Parameter	Value	Parameter	Value
$\mathcal{M}$	20	Q	1e-5(100,100,166.6,166.6,2,2)	τ_ρ	0.3
N	50	R	(0.1,0.1)	τ_T	30
τ_R	(0.4267 m, 4 deg)	τ_{x_i}	(5,5,0.8,0.8,1,1)	τ_{CMD}	40
ℓ	5	$\tau_{x_i}^{\text{own}}$	(7,7,0.8,0.8)	τ_ρ^{own}	0.2

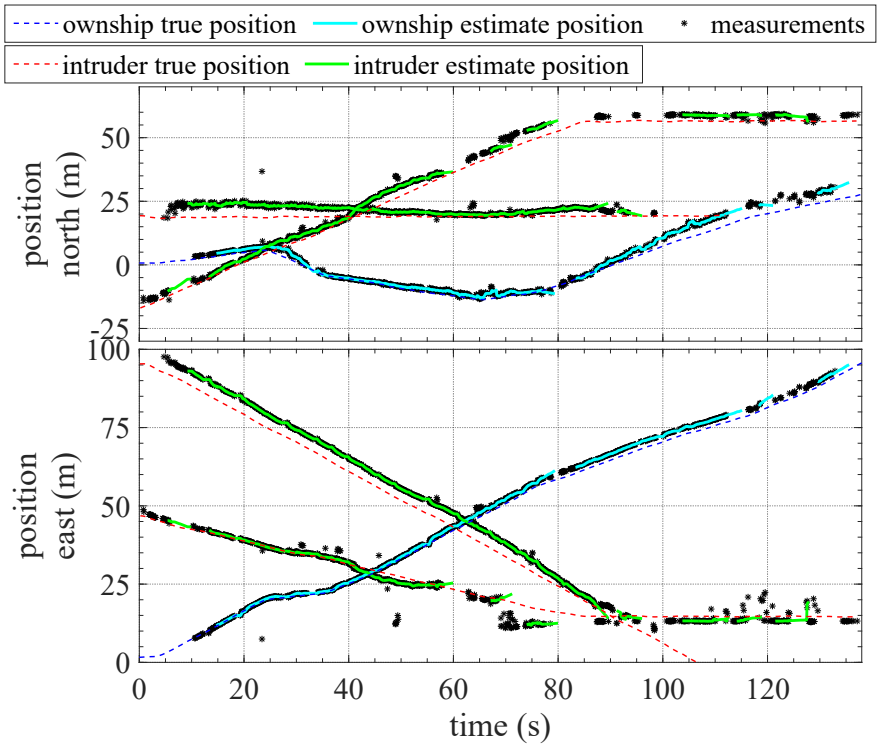


Figure 5.19: R-RANSAC tracks: position estimates of aircraft.

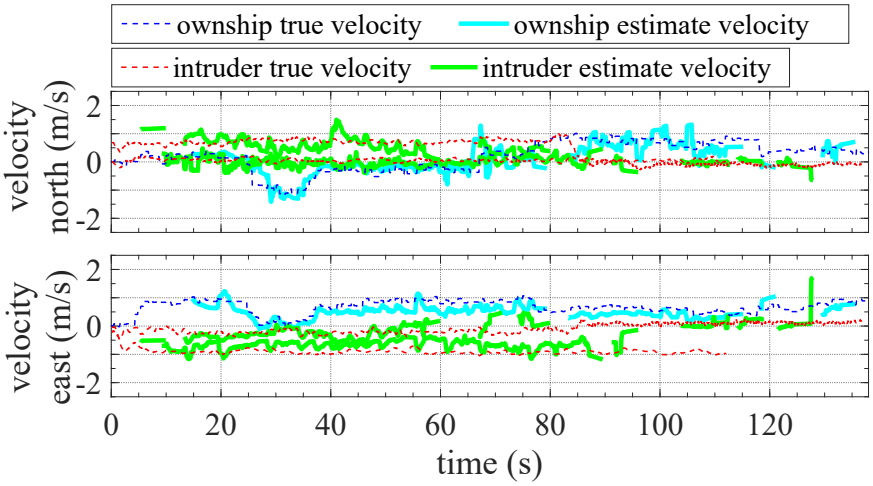


Figure 5.20: R-RANSAC tracks: velocity estimates of aircraft.

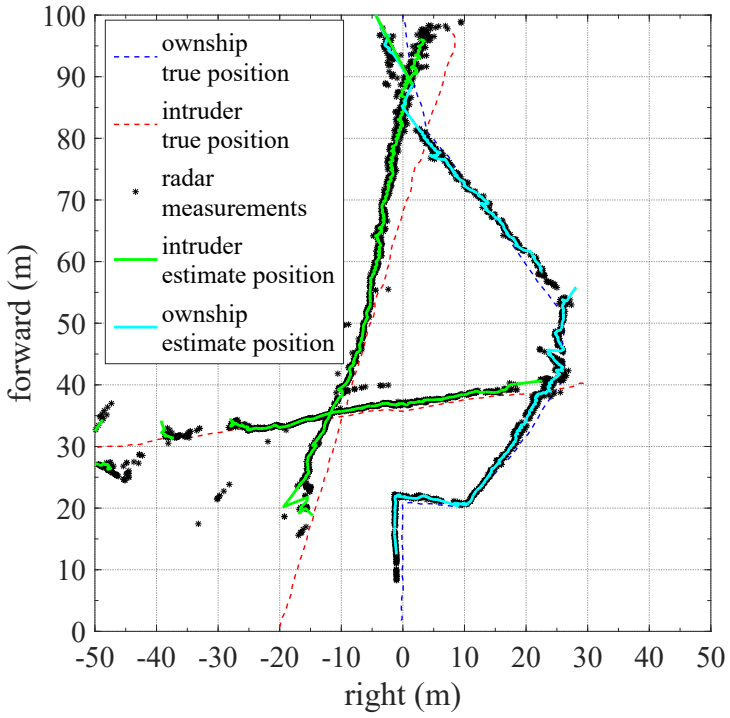


Figure 5.21: Aircraft's paths constructed using radar measurements and R-RANSAC position estimates.

The estimated position states of the aircraft have also been plotted in the radar's FR reference frame centered about the home location as seen in Figure 5.21. From this figure we see that the R-RANSAC tracks line up well with the predicted location of the radar measurements.

The relative range between the ownship and the two intruders is shown in Figure 5.22. For both intruders the relative range never falls below d_s, which means that no collisions have oc

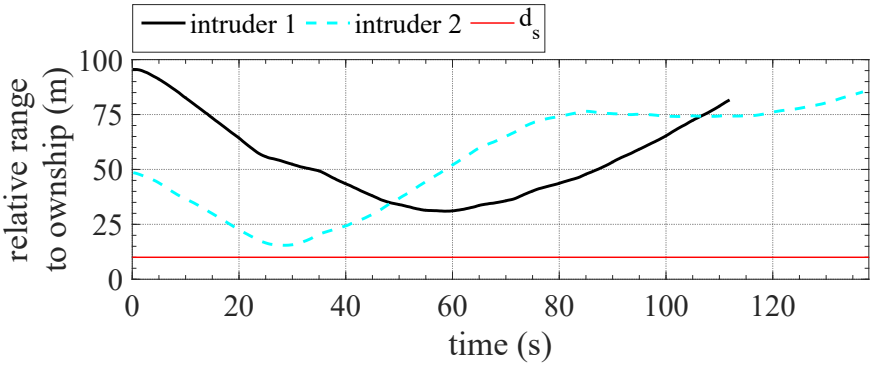


Figure 5.22: Relative range to intruders.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

6.1 Summary

The main contributions of this thesis relate to target detection and tracking elements of the detect-and-avoid (DAA) problem, and the integration of subcomponents of the DAA system into fully functional simulation and hardware implementations. Each of these contributions are specifically demonstrated in Chapters 2 through 5.

The main contribution contained within Chapter 2 is the presentation of two methods used to calculate the minimum detection range, the time-based geometric velocity vectors (TGVV) approach, and the geometric velocity vectors (GVV) approach. These methods assume that the direct head-on encounter is the scenario that requires the largest detection range. This assumption facilitates the derivation of analytical expressions for the calculation of the minimum detection range required to avoid a predefined safety volume using both dynamic and geometric models of the encounter. Using the GVV approach, a closed-form analytical expression for the minimum detection range is found by making the assumption that the ownship performs an instantaneous bank-angle maneuver. If this simplifying assumption is removed, the TGVV method can be used to numerically solve for the minimum detection range by utilizing a model of the turning dynamics of the ownship. The minimum detection range calculation takes into account the computation time involved in target tracking, risk assessment, path planning, and pilot response time delay in addition to the time required to execute the avoidance maneuver. The TGVV and GVV approaches were compared to the existing TT and GT methods and were shown to create more accurate estimates of the minimum detection range over a wide range of encounter scenarios. For every variation of the encounter parameters, the TGVV method determined a CPA estimate equal to the desired safety radius value indicating that the true minimum detection range had been found. It was also shown that as the bank-angle dynamics of the ownship became

more aggressive, the geometry-based GVV method produced results that approached those of the dynamic-model-based TGVV method. The conditions under which the TT and GT methods produce results that approach the TGVV method were also found to occur at large ownship speeds relative to the intruder speed, fast bank-angle transients, and small bank angles.

Chapter 3 contains two primary contributions. The first contribution is the formulation of the generic extended recursive-RANSAC (ER-RANSAC) algorithm, which extends the generic linear recursive-RANSAC (R-RANSAC) algorithm to nonlinear systems. The primary elements needed to extend R-RANSAC to nonlinear systems include the use of and extended Kalman filter (EKF) in the propagation and update steps within the R-RANSAC framework and in the smoothing step of the RANSAC framework, new inlier functions within R-RANSAC and RANSAC that use the nonlinear measurement equations and noise statistics of the measurements, and the use of the Gauss-Newton method within RANSAC to initialize new model hypotheses.

The Gauss-Newton method is a nonlinear regression technique which must be found through an iterative approach. The Gauss-Newton method uses the Taylor-series approximation to create linearized equations about a current estimate of the states. These linearized equations along with a measurement set can then be used to solve for the residual state difference using traditional least-squares theory. By solving for the residual state difference, we can provide an updated estimate of the states to be used for the next iteration of the Gauss-Newton method. This process is continued until the residual of the state difference falls below some threshold τ_{GN}, or the maximum number of iterations is reached ℓ_{GN}.

The second contribution within Chapter 3 is the implementation of a target detection and tracking system that robustly tracks intruder aircraft in the varied encounters a UAS might possibly experience. This system uses multiple radar units for increasing the sensor coverage to detect overtaking intruders. It also uses either the R-RANSAC or ER-RANSAC algorithms, implemented with constant-acceleration dynamic models. These recursive-RANSAC methods are used to help track multiple maneuvering aircraft in the presence of noisy, cluttered, and missed measurements. These and other results are validated using an improved detect-and-avoid simulator tool, also developed from research efforts de-

scribed in this thesis in Appendix C. In the improved simulator an increased number of intruder aircraft can be simulated from a user specified quantity. This simulator has also been restructured to provide and elegant way for the user to choose between alternate function implementations, and to provide the user a simple process for including new function implementations.

The main contribution contained within Chapters 4 and 5 is the integration of the DAA-system subcomponents into fully functional simulation and hardware implementations using a ground-based radar setup, respectively. This implementation required an intimate knowledge of each of the components of the DAA system, and as a result additional capabilities were found for each of the subcomponents to improve the overall performance of the DAA system. For the radar, a novel method for thresholding was integrated that uses the noise statistics of the received power from the radar measurements.

For the tracking algorithm a linear version of R-RANSAC was used because the collision detection algorithm only predicts collisions that result from straight line paths, which also required that the each of the intruder aircraft fly in straight line paths. Additionally, we recognize that a ground-based radar setup will provide measurements for all aircraft in its field of view, including the ownship. As such, modifications had to be made to R-RANSAC which allow for the identification of the ownship tracks. These modifications include inputting the ownship's know states into R-RANSAC, creating an additional ownship consensus set χ_o, and an additional ownship inlier ratio ρ_o. If the inlier ratio rises above a threshold $\tau_{\rho,o}$, then the track is identified as the ownship.

In the collision detection algorithm, additional encounter scenarios were considered and the integration of the ownship's known waypoint path was accounted for. For the A* step of the collision avoidance algorithm, additional costs were added which penalize a node if the closest point of approach is small or if the direction of the node is in the same direction as the intruder. For the chain-based step of the collision avoidance algorithm, a force was added that pushes the a node behind the intruder if it is within a certain distance of the intruder.

For the simulation implementation, we model a 3D ground-based radar and 3D aircraft flight paths. For the hardware implementation we use the radar hardware developed

here at Brigham Young University (BYU) in a 2D ground-based radar setup with aircraft that fly at a constant altitude. The hardware implementation had to be performed in the horizontal plane because the radar is currently only able to provide range and azimuth angle measurements. Without the additional elevation angle measurements from the radar, the altitude state information of the aircraft is unobservable. Since the hardware implementation had to be performed at a constant altitude, slight modifications had to be made to the R-RANSAC, collision detection, and collision avoidance algorithms.

Finally, extensive effort was put into creating strong and accurate radar measurements from the radar hardware. Contributions from this thesis include various DSP techniques on the radar data capable of creating range and azimuth angle measurements in between range and angle bins. After integrating each of these components of a ground-based radar DAA system together, we performed a successful flight test using three small UAS platforms, including one ownship and two intruder aircraft, thus providing the first completely successful DAA system with real hardware and no simulated components.

6.2 Recommendations for Future Work

Each of the contributions contained in this thesis have multiple areas for future research. This section describes a few of these avenues for possible future work.

6.2.1 Minimum detection range

While the direct head-on encounter is important because it requires a large minimum detection range, it would be valuable to extend the minimum detection range methods to a variety of encounter scenarios in addition to the head-on case. Additionally, validation of the minimum detection range results with flight tests to confirm the validity of the underlying assumptions and models would represent a valuable next step in defining integration requirements for UAS.

6.2.2 Tracking

In this thesis we have demonstrated the successful implementation of the ER-RANSAC algorithm, however, we have neglected to provide a detailed comparison with the traditional linear version of R-RANSAC. Providing a detailed performance comparison of these two algorithms would certainly be required to fully understand the implications for using the ER-RANSAC algorithm and would be a valuable avenue for future research.

Without the support of numerical data, it appears that ER-RANSAC has a significant increase in computation time compared to R-RANSAC. To reduce this computation time, we realize that there may be alternate ways to extend R-RANSAC to nonlinear systems. In our implementation of ER-RANSAC we used a nine-state constant-acceleration model. One method for reducing the computational overhead could be to use the full nine-state model within the R-RANSAC framework, and use a lower-order model such as a six-state constant-velocity model within RANSAC for model hypothesis and generation. As the models are passed from RANSAC to R-RANSAC, the acceleration terms could then be set to zero. Another method could be to use a hybrid approach, where we use the nonlinear EKF propagation and update steps within R-RANSAC, linear RANSAC methods for model hypothesis and generation, and a nonlinear transformation of the linear states resulting from RANSAC as they get passed back to R-RANSAC. These and other approaches could also be examined to determine the best approach for tracking truly nonlinear systems.

Just as there may be cases where the Kalman filter diverges, while the extended Kalman filter does not, we believe there may be cases where the R-RANSAC algorithm will be unable to track certain nonlinear dynamic models, while the ER-RANSAC algorithm is able to track them. We have not yet explored various types of nonlinear dynamic models that may result in this behavior, however, if these models can be identified, this would provide additional motivation for using the ER-RANSAC algorithm over the linear R-RANSAC algorithm.

For the development of a robust target detection and tracking system, we implemented a tracking simulation which uses multiple radar sensors around the ownship aircraft to increase its surveillance area. One area of future work, would be to implement this setup in hardware to verify that it works. Additionally, we have mentioned that two additional

detection devices could be added to this setup to detect objects above and below the ownship. This is an improvement that would greatly increase the robustness of the DAA system to all types of encounter scenarios.

6.2.3 Complete DAA

There are many avenues of future work for improving the complete DAA system. The first and most obvious improvement would be to extend the use of the radar hardware to an air-based setup. This would allow the ownship the freedom to move within any airspace. Another improvement that needs to be made to the radar is the ability to get elevation measurements in addition to azimuth measurements. This will result in the altitude state information of the aircraft being observable, and will allow us to perform 3D DAA experiments. The last major improvement that could be made with the radar hardware is an increased range detection capability, and as a related improvement, the ability to detect objects with a smaller radar cross section (RCS). This last improvement would hopefully allow a future flight test using small UAS without corner reflectors attached.

The primary capability that was not added to the general DAA simulator described in Appendix C, was the ability to easily perform Monte Carlo simulations. This addition would be a very valuable contribution, allowing one to truly test each of the DAA algorithms in a variety of test conditions.

Many of the autopilot functions have already been implemented in compiled C-MEX s-Functions within Simulink, however, the target detection and tracking, collision detection, and collision avoidance algorithms have not yet been implements in compiled C-MEX s-Functions. By implementing these DAA algorithms in compiled C-MEX s-Functions, the integration of these algorithms in an on-board microprocessors will be one step closer to reality. This will help integrate these DAA algorithms into an air-based DAA solution as opposed to a ground-based solution.

From the complete ground-based radar DAA system, we made the assumption that the intruder aircraft were flying in straight-line paths. This resulted in the use of the linear R-RANSAC algorithm, a straight-line collision detection algorithm, and a collision avoidance algorithm which uses straight-line paths of the intruder aircraft. An improvement that could

be made to this DAA system would be to use the nonlinear ER-RANSAC tracking algorithm developed for tracking maneuvering aircraft in conjunction with a new collision detection and collision avoidance algorithm that takes into account the behavior of maneuvering aircraft.

REFERENCES

[1] Geyer, C., Singh, S., and Chamberlain, L., 2008. Avoiding Collisions Between Aircraft: State of the Art and Requirements for UAVs operating in Civilian Airspace Tech. Rep. CMU-RI-TR-08-03, Robotics Institute, Carnegie Mellon University. vii, 2, 9, 10, 13, 14

[2] RTCA-SC-228, 2015. Draft Detect and Avoid (DAA) Minimum Operational Performance Standards for Verification and Validation. Tech. rep., Radio Telecommunications Corporation of America, Washington D.C. 1

[3] Dalamagkidis, K., Valavanis, K. P., and Piegl, L. A., 2012. *On Integrating Unmanned Aircraft Systems into the National Airspace System.*, 2nd ed., Vol. 52 Springer Science+Business Media B.V. 2

[4] Angelov, P., 2012. *Sense and Avoid in UAS: Research and Applications*. John Wiley & Sons, Ltd. 2

[5] Prats, X., Luisn, D., Jorge, R., Pablo, R., and Enric, P., 2012. "Requirements, Issues, and Challenges for Sense and Avoid in Unmanned Aircraft Systems." *Aircraft,* **49**(3), pp. 677–687. 2

[6] Boskovic, J. D., Jackson, J. A., and Mehra, R. K., 2013. "Sensor and Tracker Requirements Development for Sense and Avoid Systems for Unmanned Aerial Vehicles." In *AIAA Modeling and Simulation Technologies (MST) Conference*. 2

[7] Melnyk, R., Daniel, S., Vitali, V., and Hernando, J., 2014. "Sense and Avoid Requirements for Unmanned Aircraft Systems Using a Target Level of Safety Approach." *Risk Analysis,* **34**(10), pp. 1894–1906. 2

[8] Dalamagkidis, K., Valavanis, K. P., and Piegl, L. a., 2008. "Current Status and Future Perspectives for Unmanned Aircraft System Operations in the US." *Journal of Intelligent and Robotic Systems,* **52**(2), Feb., pp. 313–329. 2

[9] Geyer, C., Singh, S., and Chamberlain, L., 2008. "Avoiding collisions between aircraft: State of the art and requirements for UAVs operating in civilian airspace.". 2

[10] Graham, S., Chen, W., and Luca, J. D., 2011. "Multiple Intruder Autonomous Avoidance Flight Test." *Infotech@Aerospace*(March), pp. 1–22. 2

[11] Chen, R. H., Gevorkian, A., Fung, A., Chen, W., and Raska, V., 2011. "Multi-Sensor Data Integration for Autonomous Sense and Avoid." In *Proceedings of the AIAA Infotech at Aerospace*. 2

[12] Niedfeldt, P. C., 2014. "Recursive-RANSAC : A Novel Algorithm for Tracking Multiple Targets in Clutter." PhD thesis, Brigham Young University. 4, 46, 51, 54, 56, 58, 60, 62, 74, 103

[13] Contarino, V. M., and Scire Consultants, L., 2009. All Weather Sense and Avoid System for UASs Tech. rep., Report to the Office of Naval Research. 8

[14] Sahawneh, L. R., Spencer, J., Beard, R. W., and Warnick, K. F., 2016. "Minimum Required Sensing Range for UAS Sense and Avoid Systems." In *AIAA Infotech@Aerospace*, AIAA. 9, 13, 14

[15] Lee, S. M., Park, C., Johnson, M. A., and Mueller, E. R., 2013. "Investigating Effects of Well Clear Definitions on UAS Sense-And-Avoid Operations." In *Aviation Technology, Integration, and Operations Conference*, AIAA. 10

[16] Consiglio, M., Chamberlain, J., Munoz, C., and Hoffler, K., 2012. "Concept Of Integration For UAS Operations In The NAS." In *28th International Congress of the Aeronautical Sciences (ICAS)*. 10

[17] FAA SAA Workshop, 2009. FAA Workshop on Sense and Avoid (SAA) for Unmanned Aircraft Systems (UAS) Report October. 10

[18] Cook, S. P., Brooks, D., Cole, R., Hackenberg, D., and Raska, V., 2015. "Defining Well Clear for Unmanned Aircraft Systems." *AIAA Infotech @ Aerospace,* **January**, pp. 1–20. 10

[19] Beard, R. W., and McLain, T. W., 2012. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press. 12, 91, 198

[20] Erzberger, H., and Heere, K., 2010. "Algorithm and operational concept for resolving short-range conflicts." *Proceedings of the Institution of Mechanical Engineers Part G-Journal of Aerospace Engineering,* **224**(G2), pp. 225–243. 12

[21] Weisstein, E. W. Cubic Formula From MathWorld–A Wolfram Web Resource. URL: `http://mathworld.wolfram.com/CubicFormula.html` [cited: 2017-01-18]. 31

[22] Hottman, S., Hansen, K., and Berry, M., 2009. Literature review on detect, sense, and avoid technology for unmanned aircraft systems Tech. Rep. September, U.S. Department of Transportation Federal Aviation Administration. 44

[23] Niedfeldt, P. C., and Beard, R. W., 2013. "Recursive RANSAC: Multiple signal estimation with outliers." In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, Vol. 9, IFAC, pp. 430–435. 46

[24] Niedfeldt, P. C., Quist, E. B., and Beard, R. W., 2014. "Characterizing range progression of SAR point scatterers with recursive RANSAC." In *IEEE National Radar Conference - Proceedings*, pp. 712–717. 46

[25] Niedfeldt, P. C., and Beard, R. W., 2014. "Multiple target tracking using recursive RANSAC." In *2014 American Control Conference*, pp. 3393–3398. 46

[26] Ingersoll, K., 2015. "Vision Based Multiple Target Tracking Using Recursive RANSAC." Master's thesis, Brigham Young University. 46

[27] Defranco, P. C., Beard, R. W., Warnick, K. F., and Mclain, T. W., 2015. "Detecting and Tracking Moving Objects from a Small Unmanned Air Vehicle." Master's thesis, Brigham Young University. 46

[28] Mackay, J. K., 2014. "Automated Landing Site Determination for Unmanned Rotocraft Surveillance Applications." Master's thesis, Brigham Young University. 46

[29] Sakamaki, J. Y., Beard, R. W., and Rice, M. D., 2016. "Cooperative estimation for a vision-based target tracking system." Master's thesis, Brigham Young University. 46

[30] Klaus, R., 2013. "Development of a Sense and Avoid System for Small Unmanned Aircraft Systems." Master's thesis, Brigham Young University. 47, 48, 101, 194, 195

[31] Blackman, S., and Popoli, R., 1999. *Design and Analysis of Modern Tracking Systems.* Artech House, Norwood, MA. 50

[32] Bar-Shalom, Y., and Tse, E., 1975. "Tracking in a cluttered environment with probabilistic data association." *Automatica,* **11**, pp. 451–460. 50

[33] Fortmann, T., Bar-Shalom, Y., and Scheffe, M., 1980. "Multi-target tracking using joint probabilistic data association." In *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, Vol. 19, pp. 807–812. 50

[34] Reid, D., 1979. "An algorithm for tracking multiple targets." *IEEE Transactions on Automatic Control,* **24**(6), Dec., pp. 843–854. 51

[35] Mahler, R., 2000. A Theoretical Foundation for the Stein-Winter "Probability Hypothesis Density (PHD)" Multitarget Tracking Approach Tech. rep., Lockheed Martin. 51

[36] Vo, B.-N., and Ma, W.-K., 2006. "The Gaussian Mixture Probability Hypothesis Density Filter." *IEEE Transactions on Signal Processing,* **54**(11), Nov., pp. 4091–4104. 51

[37] Avitzour, D., 1995. "Stochastic simulation Bayesian approach to multitarget tracking." *IEE Proceedings - Radar, Sonar and Navigation,* **142**(2), p. 41. 51

[38] Fischler, M., and Bolles, R., 1981. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM,* **24**(6), pp. 381–395. 55, 80

[39] Chapra, S. C., and Canale, R. P., 2006. *Numerical Methods for Engineers.*, 5th ed. McGraw-Hill. 63

[40] Ferrin, J. L., 2016. "Autonomous Goal-Based Mapping and Navigation Using a Ground Robot." PhD thesis, Brigham Young University. 86

[41] Richards, M., Scheer, J., and Holm, W., 2010. *Principles of Modern Radar: Basic Principles*. SciTech Publishing. 107

[42] Spencer, J. C., and Spencer, J. C., 2015. "A Compact Phased Array Radar for UAS Sense and Avoid." Master's thesis, Brigham Young University. 111

[43] Sahawneh, L. R., 2016. "Airborne Collision Detection and Avoidance for Small UAS Sense and Avoid Systems." PhD thesis, Brigham Young University. 115, 120, 122, 125

[44] Sahawneh, L. R., Argyle, M. E., and Beard, R. W., 2016. "3D Path Planning for Small UAS Operating in Low-Altitude Airspace." In *The 2016 American Control Conference*. 126

[45] McLain, T. W., and Beard, R. W., 2000. "Trajectory Planning For Coordinated Rendezvous Of Unmanned Air Vehicles." In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Vol. 4369, AIAA Reston, VA, pp. 1–8. 126

[46] "Sahawneh, L. R., W., B. R., Avadhanamz, S., and Bai, H.." In *AIAA Guidance, Navigation, and Control (GNC) Conference*. 126

APPENDIX A. TGVV BANK-ANGLE RESPONSE

A.1 Case A: $\phi_{\max}$ is reached

Using Fig. 2.5(a) we see that interval 1 has a positive step in the aileron command and $t_0 = 0$, $\phi(t_0) = 0$, and $\dot{\phi}(t_0) = 0$. This results in the following equations for $\phi_1(t)$ and $\dot{\phi}_1(t)$

$$\begin{aligned} \phi_1(t) &= \tau\dot{\phi}_{\max}\left(e^{-t/\tau} - 1\right) + \dot{\phi}_{\max}t \\ \dot{\phi}_1(t) &= \dot{\phi}_{\max}\left(1 - e^{-t/\tau}\right) \end{aligned} \tag{A.1}$$

Interval 2 has a negative step input with $t_0 = t_1$, $\phi(t_0) = \phi_1(t_1)$, and $\dot{\phi}(t_0) = \dot{\phi}_1(t_1)$. This results in the following equations for $\phi 2(t)$ and $\dot{\phi}_2(t)$

$$\begin{aligned} \phi_2(t) &= -\tau\dot{\phi}_{\max}\left(e^{-(t-t_1)/\tau} - 1\right) - \dot{\phi}_{\max}(t - t_1) + \tau\dot{\phi}_1(t_1)\left(1 - e^{-(t-t_1)/\tau}\right) + \phi_1(t_1) \\ &= \tau\dot{\phi}_{\max}\left(1 - 2e^{-(t-t_1)/\tau} + e^{-t/\tau}\right) - \dot{\phi}_{\max}(t - 2t_1) \\ \dot{\phi}_2(t) &= -\dot{\phi}_{\max}\left(1 - e^{-(t-t_1)/\tau}\right) + \dot{\phi}_1(t_1)e^{-(t-t_1)/\tau} \\ &= \dot{\phi}_{\max}\left(2e^{-(t-t_1)/\tau} - e^{-t/\tau} - 1\right) \end{aligned} \tag{A.2}$$

For interval 3 the aileron command is zero. This means the bank angle only experiences the free response due to the initial conditions, and does not experience any forced response from the aileron input. This free response is taken from the terms on the right side of Eqs. (2.13) and (2.14) as

$$\phi^f(t) = \tau\dot{\phi}(t_0)\left(1 - e^{-(t-t_0)/\tau}\right) + \phi(t_0), \tag{A.3}$$

$$\dot{\phi}^f(t) = \dot{\phi}(t_0)e^{-(t-t_0)/\tau}. \tag{A.4}$$

Using these equations for interval 3 and $t_0 = t_2$, $\phi(t_0) = \phi_2(t_2)$, and $\dot{\phi}(t_0) = \dot{\phi}_2(t_2)$, gives the following response for interval 3

$$\begin{aligned}\phi_3(t) &= \tau\dot{\phi}_2(t_2)\left(1 - e^{-(t-t_2)/\tau}\right) + \phi_2(t_2),\\ \dot{\phi}_3(t) &= \dot{\phi}_2(t_2)e^{-(t-t_2)/\tau}.\end{aligned}$$

From Fig. 2.5(a) values for t_1 and t_2 are chosen that result in $\phi_2(t_2) = \phi_{\max}$ and $\dot{\phi}_2(t_2) = 0$. With these requirements the response for interval 3 becomes

$$\begin{aligned}\phi_3(t) &= \phi_{\max},\\ \dot{\phi}_3(t) &= 0.\end{aligned} \tag{A.5}$$

Interval 4 has a negative step input with $t_0 = t_3$, $\phi(t_0) = \phi_3(t_3)$, and $\dot{\phi}(t_0) = \dot{\phi}_3(t_3)$. This results in the following equations for $\phi_4(t)$ and $\dot{\phi}_4(t)$

$$\begin{aligned}\phi_4(t) &= -\tau\dot{\phi}_{\max}\left(e^{-(t-t_3)/\tau} - 1\right) - \dot{\phi}_{\max}(t - t_3) + \tau\dot{\phi}_3(t_3)\left(1 - e^{-(t-t_3)/\tau}\right) + \phi_3(t_3)\\ &= \tau\dot{\phi}_{\max}\left(1 - e^{-(t-t_3)/\tau}\right) - \dot{\phi}_{\max}(t - t_3) + \phi_{\max}\\ \dot{\phi}_4(t) &= -\dot{\phi}_{\max}\left(1 - e^{-(t-t_3)/\tau}\right) + \dot{\phi}_3(t_3)e^{-(t-t_3)/\tau}\\ &= \dot{\phi}_{\max}\left(e^{-(t-t_3)/\tau} - 1\right)\end{aligned} \tag{A.6}$$

Interval 5 has a positive step input with $t_0 = t_4$, $\phi(t_0) = \phi_4(t_4)$, and $\dot{\phi}(t_0) = \dot{\phi}_4(t_4)$. This results in the following equation for $\phi_5(t)$

$$\begin{aligned}\phi_5(t) &= \tau\dot{\phi}_{\max}\left(e^{-(t-t_4)/\tau} - 1\right) + \dot{\phi}_{\max}(t - t_4) + \tau\dot{\phi}_4(t_4)\left(1 - e^{-(t-t_4)/\tau}\right) + \phi_4(t_4)\\ &= \tau\dot{\phi}_{\max}\left(2e^{-(t-t_4)/\tau} - 1 - e^{-(t-t_3)/\tau}\right) + \dot{\phi}_{\max}(t - 2t_4 + t_3) + \phi_{\max}\end{aligned} \tag{A.7}$$

As stated above, we must choose values for t_1 and t_2 that result in $\phi_2(t_2) = \phi_{\max}$ and $\dot{\phi}_2(t_2) = 0$. To find the values of t_1 and t_2 we start with the expression for $\dot{\phi}_2(t)$ and the requirement that $\dot{\phi}_2(t_2) = 0$ to get

$$0 = \dot{\phi}_{\max}\left(2e^{-(t_2-t_1)/\tau} - e^{-t_2/\tau} - 1\right). \tag{A.8}$$

Next we use the expression for $\phi_2(t)$ and the requirement that $\phi_2(t_2) = \phi_{\max}$ to get

$$\phi_{\max} = \tau\dot{\phi}_{\max}\left(1 - 2e^{-(t_2-t_1)/\tau} + e^{-t_2/\tau}\right) - \dot{\phi}_{\max}(t_2 - 2t_1),$$

where we notice that the first term on the right side of the equation is the negative of Eq. (A.8) multiplied by τ. This cancels the first term resulting in

$$\phi_{\max} = -\dot{\phi}_{\max}(t_2 - 2t_1).$$

Rearranging we get the final expression for t_2 as

$$t_2 = 2t_1 - \frac{\phi_{\max}}{\dot{\phi}_{\max}}. \tag{A.9}$$

To find t_1 we substitute this value of t_2 back into Eq. A.8 and after algebraic manipulation we get

$$0 = \left(e^{-t_1/\tau}\right)^2 - 2e^{-t_1/\tau} + e^{-(\phi_{\max}/\dot{\phi}_{\max})/\tau},$$

which is a quadratic function in $e^{-t_1/\tau}$. Using the quadratic formula we can find the roots to this equation as

$$e^{-t_1/\tau} = 1 \pm \sqrt{1 - e^{-(\phi_{\max}/\dot{\phi}_{\max})/\tau}}.$$

Rearranging this equation we can solve for t_1 as

$$t_1 = -\tau \ln\left[1 \pm \sqrt{1 - e^{-(\phi_{\max}/\dot{\phi}_{\max})/\tau}}\right].$$

For this expression to give a positive value for t_1 we must use the negative sign inside the natural logarithm as

$$t_1 = -\tau \ln\left[1 - \sqrt{1 - e^{-(\phi_{\max}/\dot{\phi}_{\max})/\tau}}\right].$$

While this equation is mathematically correct, if the quantity $(\phi_{\max}/\dot{\phi}_{\max})\tau$ gets too large, numerical roundoff errors can occur while computing the value of t_1. To help reduce the chance of numerical roundoff error we multiply the term inside the natural logarithm by its

conjugate on the numerator and denominator which produces the final expression for t_1 as

$$t_1 = -\tau \ln \left[\frac{e^{-(\phi_{\max}/\dot{\phi}_{\max})/\tau}}{1 + \sqrt{1 - e^{-(\phi_{\max}/\dot{\phi}_{\max})/\tau}}} \right]. \tag{A.10}$$

Now we finish defining the rest of the time variables t_3, t_4, and t_5. The time variable t_3 is defined as

$$t_3 = t_2 + \delta T_3, \tag{A.11}$$

where δT_3 is the total time during interval 3 where the bank angle is at a constant value of $\phi_{\max}$ and can be calculated as

$$\delta T_3 = L_3/v_o, \tag{A.12}$$

where L_3 is the length of the path traveled by the ownship during interval 3 and can be calculated as

$$L_3 = R_{\min}\delta\chi_3, \tag{A.13}$$

where $\delta\chi_3$ is the change in course experienced by the ownship during interval 3 which can be calculated as

$$\delta\chi_3 = \int_{t_2}^{t_3} \frac{g}{v_o} \tan(\phi_3(t))dt = \chi_t - (\delta\chi_1 + \delta\chi_2 + \delta\chi_4|_{t_3=0,t_4=t_1} + \delta\chi_5|_{t_3=0,t_4=t_1,t_5=t_2}), \tag{A.14}$$

where $\delta\chi_1$, $\delta\chi_2$, $\delta\chi_4$, and $\delta\chi_5$ are the change in course experienced by the ownship during intervals 1, 2, 4, and 5, and χ_t is the desired change in course during the complete turning maneuver. These variables can be found using Eq. (2.11) from a coordinated turn as

$$\delta\chi_1 = \int_0^{t_1} \frac{g}{v_o} \tan(\phi_1(t))dt, \tag{A.15}$$

$$\delta\chi_2 = \int_{t_1}^{t_2} \frac{g}{v_o} \tan(\phi_2(t))dt, \tag{A.16}$$

$$\delta\chi_4 = \int_{t_3}^{t_4} \frac{g}{v_o} \tan(\phi_4(t))dt, \tag{A.17}$$

$$\delta\chi_5 = \int_{t_4}^{t_5} \frac{g}{v_o} \tan(\phi_5(t))dt. \tag{A.18}$$

The final expression for t_3 is now found to be

$$t_3 = t_2 + \frac{R_{\min}\delta\chi_3}{v_o}, \tag{A.19}$$

where $\delta\chi_1$, $\delta\chi_2$, $\delta\chi_4$, and $\delta\chi_5$ are defined in Eqs. (A.15), (A.16), (A.17), and (A.18). Now that we have found t_3 we can define t_4 and t_5 as

$$t_4 = t_3 + t_1, \tag{A.20}$$

$$t_5 = t_3 + t_2. \tag{A.21}$$

A.2 Case B: $\phi_{\max}$ is not reached

Similar to Case A, we use Eqs. (2.13) and (2.14) to define the response of the bank angle to step inputs from the ailerons. Using Fig. 2.5(b) we see that interval 1 has a positive step and, therefore, has the same response to Case A as

$$\begin{aligned} \phi_1(t) &= \tau\dot{\phi}_{\max}\left(e^{-t/\tau} - 1\right) + \dot{\phi}_{\max}t, \\ \dot{\phi}_1(t) &= \dot{\phi}_{\max}\left(1 - e^{-t/\tau}\right). \end{aligned} \tag{A.22}$$

The bank-angle response for intervals 2 and 4 can be combined into a single expression because the aileron input is $-\delta_a$ for both intervals. The response for these intervals has the same response as interval 2 in Case A.

$$\begin{aligned} \phi_{2,4}(t) &= \tau\dot{\phi}_{\max}\left(1 - 2e^{-(t-t_1)/\tau} + e^{-t/\tau}\right) - \dot{\phi}_{\max}(t - 2t_1), \\ \dot{\phi}_{2,4}(t) &= \dot{\phi}_{\max}\left(2e^{-(t-t_1)/\tau} - e^{-t/\tau} - 1\right). \end{aligned} \tag{A.23}$$

Interval 5 differs from Case A because the previous interval 4 has been combined with interval 2. For interval 5 of this case we also have a positive step and $t_0 = t_4$, however,

$\phi(t_0) = \phi_{2,4}(t_4)$, and $\dot{\phi}(t_0) = \dot{\phi}_{2,4}(t_4)$. This results in the following equation for $\phi_5(t)$

$$\begin{aligned}\phi_5(t) &= \tau\dot{\phi}_{\max}\left(e^{-(t-t_4)/\tau} - 1\right) + \dot{\phi}_{\max}(t - t_4) + \tau\dot{\phi}_{2,4}(t_4)\left(1 - e^{-(t-t_4)/\tau}\right) + \phi_{2,4}(t_4), \\ &= \tau\dot{\phi}_{\max}\left(2e^{-(t-t_4)/\tau} - 2e^{-(t-t_1)/\tau} + e^{-t/\tau} - 1\right) + \dot{\phi}_{\max}(t + 2t_1 - 2t_4). \end{aligned} \tag{A.24}$$

In deriving t_1 for Case A we used the constraint $\phi_2(t_2) = \phi_{\max}$, however, this constraint is not true for Case B. Finding t_1 for this case requires a more in depth analysis. From the total turning maneuver of the ownship for Case B we know that

$$\chi_t = \delta\chi_1 + \delta\chi_{2,4} + \delta\chi_5,$$

where $\delta\chi_1$ and $\delta\chi_5$ were defined in Eqs. (A.15) and (A.18) in Case A, and $\delta\chi_{2,4}$ is the change in course experienced by the ownship during the combined intervals 2 and 4 as

$$\delta\chi_{2,4} = \int_{t_1}^{t_4} \frac{g}{v_o}\tan(\phi_{2,4}(t))dt. \tag{A.25}$$

Moving χ_t to the right side of the equation results in

$$0 = \delta\chi_1 + \delta\chi_{2,4} + \delta\chi_5 - \chi_t. \tag{A.26}$$

To find t_1 we express each of the components of Eq. (A.26) in terms of the variable t_1 and use the Newton-Raphson method to find the value of t_1 that makes the equation equal zero. The variables that need to be expressed in terms of t_1 are t_2, t_4, and t_5.

A constraint from Case A that we can use for Case B is $\dot{\phi}_2(t_2) = 0$. This constraint resulted in Eq. (A.8) which can be solved for t_2 in terms of t_1 as

$$t_2 = \tau\ln\left[2e^{t_1/\tau} - 1\right]. \tag{A.27}$$

Using Fig. 2.5(b) and Eq. (A.27) we create expressions for t_4 and t_5 in terms of t_1 as

$$t_4 = t_2 + t_1 = t_1 + \tau \ln \left[2e^{t_1/\tau} - 1\right], \tag{A.28}$$

$$t_5 = 2t_2 = 2\tau \ln \left[2e^{t_1/\tau} - 1\right]. \tag{A.29}$$

Using the expression for t_4 we modify Eq. (A.24) to be expressed in terms of t_1 as

$$\phi_5(t) = \tau\dot{\phi}_{\max} \left(e^{-t/\tau}(2e^{t_1/\tau} - 1)^2 - 1 - 2\ln \left[2e^{t_1/\tau} - 1\right]\right) + \dot{\phi}_{\max} t \tag{A.30}$$

APPENDIX B. NONLINEAR OBSERVABILITY

Using radar as the detection device which provides range, azimuth, and elevation measurements, we now show the states used within our specific nonlinear dynamic model are indeed observable. To standardize the notation used within this section we first define the intruder states in numeric form as

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9]^\top. \tag{B.1}$$

Since the ownship states are not being estimated, but are instead inputs to the estimator, we do not give the ownship states numeric subscripts as we did with the intruder states. Instead we maintain the original notation for the ownship states as

$$\mathbf{x}_o = [p_n, p_e, h, \phi, \theta, \psi]^\top, \tag{B.2}$$

where we have dropped the ownship identifier from the subscript of each of the individual state components for simplicity.

Utilizing this notation for the states of the intruder and ownship the observability matrix for this specific problem is seen as

$$\mathcal{O} = \frac{\partial}{\partial \mathbf{x}} \begin{pmatrix} h(\mathbf{x}, \mathbf{x}_o) \\ L_f h(\mathbf{x}, \mathbf{x}_o) \\ L_f^2 h(\mathbf{x}, \mathbf{x}_o) \end{pmatrix} = \frac{\partial}{\partial \mathbf{x}} \begin{pmatrix} h_1(\mathbf{x}, \mathbf{x}_o) \\ h_2(\mathbf{x}, \mathbf{x}_o) \\ h_3(\mathbf{x}, \mathbf{x}_o) \\ L_f h_1(\mathbf{x}, \mathbf{x}_o) \\ L_f h_2(\mathbf{x}, \mathbf{x}_o) \\ L_f h_3(\mathbf{x}, \mathbf{x}_o) \\ L_f^2 h_1(\mathbf{x}, \mathbf{x}_o) \\ L_f^2 h_2(\mathbf{x}, \mathbf{x}_o) \\ L_f^2 h_3(\mathbf{x}, \mathbf{x}_o) \end{pmatrix}$$

$$= \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} & \frac{\partial h_1}{\partial x_4} & \frac{\partial h_1}{\partial x_5} & \frac{\partial h_1}{\partial x_6} & \frac{\partial h_1}{\partial x_7} & \frac{\partial h_1}{\partial x_8} & \frac{\partial h_1}{\partial x_9} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} & \frac{\partial h_2}{\partial x_4} & \frac{\partial h_2}{\partial x_5} & \frac{\partial h_2}{\partial x_6} & \frac{\partial h_2}{\partial x_7} & \frac{\partial h_2}{\partial x_8} & \frac{\partial h_2}{\partial x_9} \\ \frac{\partial h_3}{\partial x_1} & \frac{\partial h_3}{\partial x_2} & \frac{\partial h_3}{\partial x_3} & \frac{\partial h_3}{\partial x_4} & \frac{\partial h_3}{\partial x_5} & \frac{\partial h_3}{\partial x_6} & \frac{\partial h_3}{\partial x_7} & \frac{\partial h_3}{\partial x_8} & \frac{\partial h_3}{\partial x_9} \\ \frac{\partial (L_f h_1)}{\partial x_1} & \frac{\partial (L_f h_1)}{\partial x_2} & \frac{\partial (L_f h_1)}{\partial x_3} & \frac{\partial (L_f h_1)}{\partial x_4} & \frac{\partial (L_f h_1)}{\partial x_5} & \frac{\partial (L_f h_1)}{\partial x_6} & \frac{\partial (L_f h_1)}{\partial x_7} & \frac{\partial (L_f h_1)}{\partial x_8} & \frac{\partial (L_f h_1)}{\partial x_9} \\ \frac{\partial (L_f h_2)}{\partial x_1} & \frac{\partial (L_f h_2)}{\partial x_2} & \frac{\partial (L_f h_2)}{\partial x_3} & \frac{\partial (L_f h_2)}{\partial x_4} & \frac{\partial (L_f h_2)}{\partial x_5} & \frac{\partial (L_f h_2)}{\partial x_6} & \frac{\partial (L_f h_2)}{\partial x_7} & \frac{\partial (L_f h_2)}{\partial x_8} & \frac{\partial (L_f h_2)}{\partial x_9} \\ \frac{\partial (L_f h_3)}{\partial x_1} & \frac{\partial (L_f h_3)}{\partial x_2} & \frac{\partial (L_f h_3)}{\partial x_3} & \frac{\partial (L_f h_3)}{\partial x_4} & \frac{\partial (L_f h_3)}{\partial x_5} & \frac{\partial (L_f h_3)}{\partial x_6} & \frac{\partial (L_f h_3)}{\partial x_7} & \frac{\partial (L_f h_3)}{\partial x_8} & \frac{\partial (L_f h_3)}{\partial x_9} \\ \frac{\partial (L_f^2 h_1)}{\partial x_1} & \frac{\partial (L_f^2 h_1)}{\partial x_2} & \frac{\partial (L_f^2 h_1)}{\partial x_3} & \frac{\partial (L_f^2 h_1)}{\partial x_4} & \frac{\partial (L_f^2 h_1)}{\partial x_5} & \frac{\partial (L_f^2 h_1)}{\partial x_6} & \frac{\partial (L_f^2 h_1)}{\partial x_7} & \frac{\partial (L_f^2 h_1)}{\partial x_8} & \frac{\partial (L_f^2 h_1)}{\partial x_9} \\ \frac{\partial (L_f^2 h_2)}{\partial x_1} & \frac{\partial (L_f^2 h_2)}{\partial x_2} & \frac{\partial (L_f^2 h_2)}{\partial x_3} & \frac{\partial (L_f^2 h_2)}{\partial x_4} & \frac{\partial (L_f^2 h_2)}{\partial x_5} & \frac{\partial (L_f^2 h_2)}{\partial x_6} & \frac{\partial (L_f^2 h_2)}{\partial x_7} & \frac{\partial (L_f^2 h_2)}{\partial x_8} & \frac{\partial (L_f^2 h_2)}{\partial x_9} \\ \frac{\partial (L_f^2 h_3)}{\partial x_1} & \frac{\partial (L_f^2 h_3)}{\partial x_2} & \frac{\partial (L_f^2 h_3)}{\partial x_3} & \frac{\partial (L_f^2 h_3)}{\partial x_4} & \frac{\partial (L_f^2 h_3)}{\partial x_5} & \frac{\partial (L_f^2 h_3)}{\partial x_6} & \frac{\partial (L_f^2 h_3)}{\partial x_7} & \frac{\partial (L_f^2 h_3)}{\partial x_8} & \frac{\partial (L_f^2 h_3)}{\partial x_9} \end{bmatrix}, \quad \text{(B.3)}$$

where we have extended the matrix down by nine rows or equivalently three sets of measurement equations. This will later prove to be a sufficient number of rows to make the rank of the observability matrix equal to the number of states $n = 9$. Evaluating the partial derivatives in the matrix above results in many of the terms being equal to zero. Although we do not show the derivation of these zero terms, we insert them into the observability

matrix at this time as

$$\mathcal{O} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 0 & 0 & 0 & 0 \\ b_{11} & b_{12} & b_{13} & c_{11} & c_{12} & c_{13} & 0 & 0 & 0 \\ b_{21} & b_{22} & b_{23} & c_{21} & c_{22} & c_{23} & 0 & 0 & 0 \\ b_{31} & b_{32} & b_{33} & c_{31} & c_{32} & c_{33} & 0 & 0 & 0 \\ d_{11} & d_{12} & d_{13} & e_{11} & e_{12} & e_{13} & f_{11} & f_{12} & f_{13} \\ d_{21} & d_{22} & d_{23} & e_{21} & e_{22} & e_{23} & f_{21} & f_{22} & f_{23} \\ d_{31} & d_{32} & d_{33} & e_{31} & e_{32} & e_{33} & f_{31} & f_{32} & f_{33} \end{bmatrix} = \begin{bmatrix} A & \mathbf{0} & \mathbf{0} \\ B & C & \mathbf{0} \\ D & E & F \end{bmatrix}, \tag{B.4}$$

where we have grouped multiple elements together to form $\mathbb{R}^{m \times m}$ block matrices, A, B, C, D, E, F, and $\mathbf{0}$.

To determine if the intruder states are observable, we will now evaluate the determinant of the observability matrix and verify that it does not equal zero. Using the block matrix representation shown above, this determinant is calculated as

$$\det|\mathcal{O}| = \det|A| \det|C| \det|F|, \tag{B.5}$$

where we notice that the matrices B, D, and E matrices do not appear in the solution. Since we do not need the B, D, and E matrices, we will not derive expressions for them, and will only derive expressions for the A, C, and F matrices.

To simplify the notation of the derivation process we make the following definitions

$$L_1 = (x_1 - p_n)^2 + (x_2 - p_e)^2 + (x_3 - h)^2, \tag{B.6}$$

$$L_2 = (x_1 - p_n)^2 + (x_2 - p_e)^2, \tag{B.7}$$

$$\overline{x}_1 = x_1 - p_n, \tag{B.8}$$

$$\overline{x}_2 = x_2 - p_e, \tag{B.9}$$

$$\overline{x}_3 = x_3 - h \tag{B.10}$$

B.1 Matrix A

After making the necessary calculations, the A matrix is shown as

$$A = \begin{bmatrix} \frac{\overline{x}_1}{\sqrt{L_1}} & \frac{\overline{x}_2}{\sqrt{L_1}} & \frac{\overline{x}_3}{\sqrt{L_1}} \\ \frac{-\overline{x}_2}{L_2} & \frac{\overline{x}_1}{L_2} & 0 \\ \frac{-\overline{x}_1\overline{x}_3}{L_1\sqrt{L_2}} & \frac{-\overline{x}_2\overline{x}_3}{L_1\sqrt{L_2}} & \frac{\overline{x}_1^2+\overline{x}_2^2}{L_1\sqrt{L_2}}, \end{bmatrix}, \tag{B.11}$$

where the determinant of this matrix is shown as

$$\det|A| = \frac{1}{\sqrt{L_1 L_2}} = \frac{1}{\sqrt{\left[(x_1-p_n)^2+(x_2-p_e)^2+(x_3-h)^2\right]\left[(x_1-p_n)^2+(x_2-p_e)^2\right]}}. \tag{B.12}$$

B.2 Matrix C

To calculate the C matrix we first define

$$L_f h = \frac{\partial h}{\partial x} f(x) = \begin{bmatrix} \frac{\overline{x}_1 x_4 \cos x_5 \cos x_6 + \overline{x}_2 x_4 \sin x_5 \cos x_6 + \overline{x}_3 x_4 \sin x_6}{\sqrt{L_1}} \\ \frac{-\overline{x}_2 x_4 \cos x_5 \cos x_6 + \overline{x}_1 x_4 \sin x_5 \cos x_6}{L_2} \\ \frac{-\overline{x}_1\overline{x}_3 x_4 \cos x_5 \cos x_6 - \overline{x}_2\overline{x}_3 x_4 \sin x_5 \cos x_6 + \left(\overline{x}_1^2+\overline{x}_2^2\right) x_4 \sin x_6}{L_1\sqrt{L_2}} \end{bmatrix}. \tag{B.13}$$

To get the C matrix we only need to take the partial derivatives with respect to the states x_4, x_5, and x_6. These partials are calculated as

$$\frac{\partial}{\partial x_4}(L_f h_1) = \frac{\overline{x}_1 \cos x_5 \cos x_6 + \overline{x}_2 \sin x_5 \cos x_6 + \overline{x}_3 \sin x_6}{\sqrt{L_1}} = \frac{N_1}{\sqrt{L_1}}, \tag{B.14}$$

$$\frac{\partial}{\partial x_5}(L_f h_1) = \frac{-\overline{x}_1 x_4 \sin x_5 \cos x_6 + \overline{x}_2 x_4 \cos x_5 \cos x_6}{\sqrt{L_1}} = \frac{N_2}{\sqrt{L_1}}, \tag{B.15}$$

$$\frac{\partial}{\partial x_6}(L_f h_1) = \frac{-\overline{x}_1 x_4 \cos x_5 \sin x_6 - \overline{x}_2 x_4 \sin x_5 \sin x_6 + \overline{x}_3 x_4 \cos x_6}{\sqrt{L_1}} = \frac{N_3}{\sqrt{L_1}}, \tag{B.16}$$

$$\frac{\partial}{\partial x_4}(L_f h_2) = \frac{-\overline{x}_2 \cos x_5 \cos x_6 + \overline{x}_1 \sin x_5 \cos x_6}{L_2} = \frac{N_4}{L_2}, \tag{B.17}$$

$$\frac{\partial}{\partial x_5}(L_f h_2) = \frac{\overline{x}_2 x_4 \sin x_5 \cos x_6 + \overline{x}_1 x_4 \cos x_5 \cos x_6}{L_2} = \frac{N_5}{L_2}, \tag{B.18}$$

$$\frac{\partial}{\partial x_6}(L_f h_2) = \frac{\overline{x}_2 x_4 \cos x_5 \sin x_6 - \overline{x}_1 x_4 \sin x_5 \sin x_6}{L_2} = \frac{N_6}{L_2}, \tag{B.19}$$

$$\frac{\partial}{\partial x_4}(L_f h_3) = \frac{-\overline{x}_1 \overline{x}_3 \cos x_5 \cos x_6 - \overline{x}_2 \overline{x}_3 \sin x_5 \cos x_6 + (\overline{x}_1^2 + \overline{x}_2^2) \sin x_6}{L_1 \sqrt{L_2}} = \frac{N_7}{L_1 \sqrt{L_2}}, \tag{B.20}$$

$$\frac{\partial}{\partial x_5}(L_f h_3) = \frac{\overline{x}_1 \overline{x}_3 x_4 \sin x_5 \cos x_6 - \overline{x}_2 \overline{x}_3 x_4 \cos x_5 \cos x_6}{L_1 \sqrt{L_2}} = \frac{N_8}{L_1 \sqrt{L_2}}, \tag{B.21}$$

$$\frac{\partial}{\partial x_6}(L_f h_3) = \frac{\overline{x}_1 \overline{x}_3 x_4 \cos x_5 \sin x_6 + \overline{x}_2 \overline{x}_3 x_4 \sin x_5 \sin x_6 + (\overline{x}_1^2 + \overline{x}_2^2)\, x_4 \cos x_6}{L_1 \sqrt{L_2}} = \frac{N_9}{L_1 \sqrt{L_2}}. \tag{B.22}$$

Using these definitions the C matrix is now shown as

$$C = \begin{bmatrix} \frac{N_1}{\sqrt{L_1}} & \frac{N_2}{\sqrt{L_1}} & \frac{N_3}{\sqrt{L_1}} \\ \frac{N_4}{L_2} & \frac{N_5}{L_2} & \frac{N_6}{L_2} \\ \frac{N_7}{L_1\sqrt{L_2}} & \frac{N_8}{L_1\sqrt{L_2}} & \frac{N_9}{L_1\sqrt{L_2}}, \end{bmatrix}, \tag{B.23}$$

where the determinant of this matrix is shown as

$$\det|C| = \frac{x_4^2 \cos x_6}{\sqrt{L_1 L_2}} = \frac{x_4^2 \cos x_6}{\sqrt{\left[(x_1 - p_n)^2 + (x_2 - p_e)^2 + (x_3 - h)^2\right]\left[(x_1 - p_n)^2 + (x_2 - p_e)^2\right]}}. \tag{B.24}$$

B.3 Matrix F

To calculate the F matrix we first define

$$L_f^2 h = \frac{\partial (L_f h)}{\partial x} f(x) = \begin{bmatrix} g_1(x_1, ..., x_6) + \frac{N_1}{\sqrt{L_1}} x_7 + \frac{N_2}{\sqrt{L_1}} x_8 + \frac{N_3}{\sqrt{L_1}} x_9 \\ g_2(x_1, ..., x_6) + \frac{N_4}{L_2} x_7 + \frac{N_5}{L_2} x_8 + \frac{N_6}{L_2} x_9 \\ g_3(x_1, ..., x_6) + \frac{N_7}{L_1\sqrt{L_2}} x_7 + \frac{N_8}{L_1\sqrt{L_2}} x_8 + \frac{N_9}{L_1\sqrt{L_2}} x_9 \end{bmatrix}, \tag{B.25}$$

where $g_i(x_1, ..., x_6)$ denotes a generic function of the states x_1 through x_6. To get the F matrix we only need to take the partial derivatives with respect to the states x_7, x_8, and x_9 which means the generic functions $g_i(x_1, ..., x_6)$ will have no effect. The partials for the F matrix are now calculated as

$$\frac{\partial}{\partial x_7}(L_f^2 h_1) = \frac{N_1}{\sqrt{L_1}}, \tag{B.26}$$

$$\frac{\partial}{\partial x_8}(L_f^2 h_1) = \frac{N_2}{\sqrt{L_1}}, \tag{B.27}$$

$$\frac{\partial}{\partial x_9}(L_f^2 h_1) = \frac{N_3}{\sqrt{L_1}}, \tag{B.28}$$

$$\frac{\partial}{\partial x_7}(L_f^2 h_2) = \frac{N_4}{L_2}, \tag{B.29}$$

$$\frac{\partial}{\partial x_8}(L_f^2 h_2) = \frac{N_5}{L_2}, \tag{B.30}$$

$$\frac{\partial}{\partial x_9}(L_f^2 h_2) = \frac{N_6}{L_2}, \tag{B.31}$$

$$\frac{\partial}{\partial x_7}(L_f^2 h_3) = \frac{N_7}{L_1\sqrt{L_2}}, \tag{B.32}$$

$$\frac{\partial}{\partial x_8}(L_f^2 h_3) = \frac{N_8}{L_1\sqrt{L_2}}, \tag{B.33}$$

$$\frac{\partial}{\partial x_9}(L_f^2 h_3) = \frac{N_9}{L_1\sqrt{L_2}}. \tag{B.34}$$

Using these definitions the F matrix is now shown as

$$F = \begin{bmatrix} \frac{N_1}{\sqrt{L_1}} & \frac{N_2}{\sqrt{L_1}} & \frac{N_3}{\sqrt{L_1}} \\ \frac{N_4}{L_2} & \frac{N_5}{L_2} & \frac{N_6}{L_2} \\ \frac{N_7}{L_1\sqrt{L_2}} & \frac{N_8}{L_1\sqrt{L_2}} & \frac{N_9}{L_1\sqrt{L_2}} \end{bmatrix}. \tag{B.35}$$

This matrix is exactly the same as matrix C, so the determinant of F is shown as

$$\det|F| = \frac{x_4^2 \cos x_6}{\sqrt{L_1 L_2}} = \frac{x_4^2 \cos x_6}{\sqrt{\left[(x_1 - p_n)^2 + (x_2 - p_e)^2 + (x_3 - h)^2\right]\left[(x_1 - p_n)^2 + (x_2 - p_e)^2\right]}}. \tag{B.36}$$

The final step is to calculate the determinant of the observability matrix which was previously shown to be $\det|\mathcal{O}| = \det|A| \det|C| \det|F|$. Using the expressions we have derived for the determinant of the A, C, and F matrices, the determinant of the observability matrix is shown as

$$\det|\mathcal{O}| = \frac{x_4^4 \cos^2 x_6}{\left[(x_1 - p_n)^2 + (x_2 - p_e)^2 + (x_3 - h)^2\right]^{3/2}\left[(x_1 - p_n)^2 + (x_2 - p_e)^2\right]^{3/2}}. \tag{B.37}$$

There are two cases when this determinant equals zero. The first case is when $x_4 = 0$. This means that the ground speed V_g of the intruder is equal to zero. If the intruder has zero velocity then there is no course angle or flight path angle which means these two states are unobservable and therefore the determinant of the observability matrix will be equal to zero. The second case is when $x_6 = m\frac{\pi}{2}, m = 1, 3, 5, ...$ This means that the flight path angle γ is either 90 degrees up or 90 degrees down which means the aircraft is traveling straight up or straight down. If this is the case then there is no course angle which means it is unobservable and therefore the determinant of the observability matrix will be equal to zero.

APPENDIX C. GENERAL PURPOSE DAA SIMULATOR

The DAA problem contains several subcomponents that require significant development processes. In the development of these types of systems, a simulation environment is typically used to test the validity of each of the individual subcomponents. The use of a simulation environment is particularly useful for DAA systems, as the types of encounter scenarios required by this system include multiple aircraft directed at one another to provide potential collision conflicts. By first testing these DAA systems in simulation, we help prevent potential accidents that may lead to the injury or destruction of people, animal life, hardware, or personal property. In this appendix we describe the development of a general purpose DAA simulator that includes components originally developed in Ref. [30].

C.1 Desired Capabilities

At the most basic level, a DAA simulator needs to contain a model of the ownship aircraft and at least one intruder aircraft. It also needs to include some method of detecting and tracking the intruder aircraft, an algorithm to determine when potential collision threats are expected to occur and when the ownship should perform an evasive maneuver, and an algorithm to plan new flight paths to avoid the potential collisions. Finally, the simulator needs some way of collecting and analyzing important data gained from the simulation.

While these capabilities are the minimum requirements necessary to implement a basic DAA simulator, additional capabilities are both necessary and helpful in the development of a robust DAA systems. A few of these additional capabilities include the ability to display the simulation as it is running, record movies of the simulation, change the playback speed of the simulation or any other parameters, increase the number of intruder aircraft, select between alternate DAA function implementations for comparison purposes, easily add new DAA function implementations, implement various DAA functions using compiled

languages to increase the speed of the simulation or using interpreted languages for faster initial development time, run Monte Carlo simulations, and many others. In adding each of these capabilities, another convenient feature is the ability to turn each option on and off, or change their value, within a centralized configuration function easily accessible to the user.

Given the fact that such a simulator would potentially be created and used among multiple researching students and professors, and shared with external interested parties, one final attribute for the simulator is needed. This final attribute includes the use of version control to prevent the loss of valuable working implementations, and the storage of this version-controlled simulator on a remote repository that can easily be accesses by all those who need access to this simulation. By using a remote repository to save a version-controlled DAA simulator, future research efforts can easily be appended, resulting in a single up-to-date version of a DAA simulator.

By adding the optional capabilities listed, the DAA simulator will become a more useful tool in the development of a DAA solution. Also, by making this simulator configurable, the research goals of various users can more easily be achieved as it allows for the personalization of the DAA simulator. Finally, the ultimate goal of the DAA simulator is to provide a stepping stone in the implementation of the DAA subsystems with actual hardware. By creating a method for integrating functions written in compiled languages into the simulator, we are facilitating the portability of these functions to microprocessor hardware, which would certainly be required by small UAS with limited payload capabilities.

C.2 Previous Simulator

A general purpose DAA simulator was previously developed at BYU by Robert Klaus using Simulink/Matlab [30]. This simulator met all of the basic requirements defined in the previous section, and provided several of the other optional capabilities including the ability to display the simulation as it is running, record movies of the simulation, change the playback speed of the simulation, and the use of compiled C-MEX s-functions to increase the speed of the simulation.

This simulator modeled a single ownship and intruder UAS using high fidelity dynamic models. This simulator also used a basic radar model with a limited field of view of

approximated 135 degrees in azimuth and 15 degrees in elevation. This radar provides measurements of range, range rate, and bearing. The radar module used within this simulator was only set up to accept the states from a single intruder aircraft, and to output a single set of radar measurements corresponding to this same intruder aircraft. The tracking algorithm used within this simulator was a basic EKF for tracking a single intruder aircraft, and as a result the output of the tracking module was only designed to carry the estimated state information of a single intruder aircraft. Finally, this simulator did not contain a convenient way to use modular function blocks, but was set up with a single block type for each function module.

C.3 Improved Simulator

The previous DAA simulator provided an excellent starting point in the development of a more generic and modular simulator. The following two sections will describe the improvements we have made in the new simulator, followed by graphical depictions of how this simulator is setup within Simulink.

C.3.1 Capabilities Achieved

The first and most notable improvement that has been made to the DAA simulator is the addition of an increased number of intruders. This has been done using a single intruder module, where the contents of the module have been parameterized to simulate a user specified number of aircraft. Additionally, all of the buses that route information related to the intruders have been parameterized. One of these buses is used to send the state information of each of the aircraft to the tracking module contained within the simulator. This tracking module has also been parameterized to accept an increased number of intruder aircraft.

The second improvement has to do with making the simulator more modular. Within Simulink there are four basic methods for implementing a user-defined function that we are interested in using. The first method uses an interpreted Matlab function, the second method uses a level-1 Matlab s-Function, the third method uses a level-2 Matlab s-Function,

and the fourth method uses a compiled level-2 C-MEX s-function. We have added the capability to specify the type of user defined function block used for each module within the simulation. After the user specifies the desired function types for each function, the model is automatically occupied with each of these function-block types. In addition to specifying the the type of function used in the simulation, the user is also able to select from a variety of specific function implementations. These implementations could, for example, be written using different algorithms, or using code from various contributors. Since the user is now capable of selecting a variety of specific function implementations, we have standardized a method for integrating new user-defined functions. This standardized approach allows the user to quickly implement new functions they are writing within the simulation.

For the previous two contributions mentioned, we have describe how the input from the user can be used to select appropriate options to be used within the current simulation run. These options, and many other options specified by the user have been collected into a single Matlab setup script. This setup script provides a readily accessible location, where the user does not have to dig through significant amounts of code to change basic operations of the simulation. Additional details on how to use this simulator are contained within a README.txt file. This file contains all the necessary information needed to get the simulator up and running for the first time and for adding user-defined functions. Finally, this new DAA simulator is now version controlled and stored on a remote repository that can be easily accesses by all those who need this simulation.

C.3.2 Organizational Structure

We now provide a high-level overview of the structure of this new DAA simulator. At the highest level there are four main blocks as seen in Figure C.1. The large blue block contains the ownship aircraft dynamics and control, and the necessary DAA algorithms. The large red block contains the parameterized intruder aircraft dynamics and control. The other two smaller blocks include a green block that generates environmental wind, gravity, and air density values, and a gray block that contains all of the software necessary to visualize the results of the simulation.

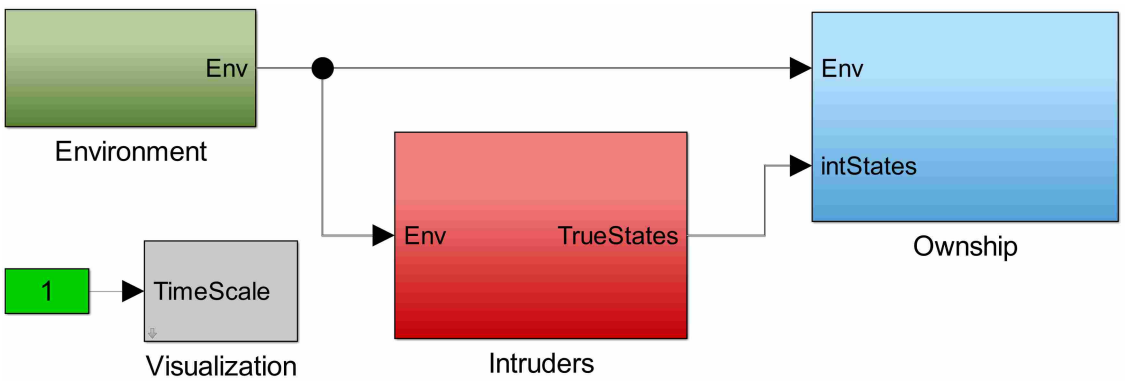


Figure C.1: Top level of general DAA simulator.

The contents of the ownship and intruder blocks are seen in Figure C.2, where we notice that the control and dynamics of the aircraft are modeled using the autopilot structure developed in Ref. [19]. This autopilot structure includes a path planner, path manager, path follower, autopilot, unmanned aircraft dynamic model, and a state estimator. The last MAV block in Figure C.2 include two subcomponents including the autopilot block and the unmanned aircraft block shown in Figure C.3.

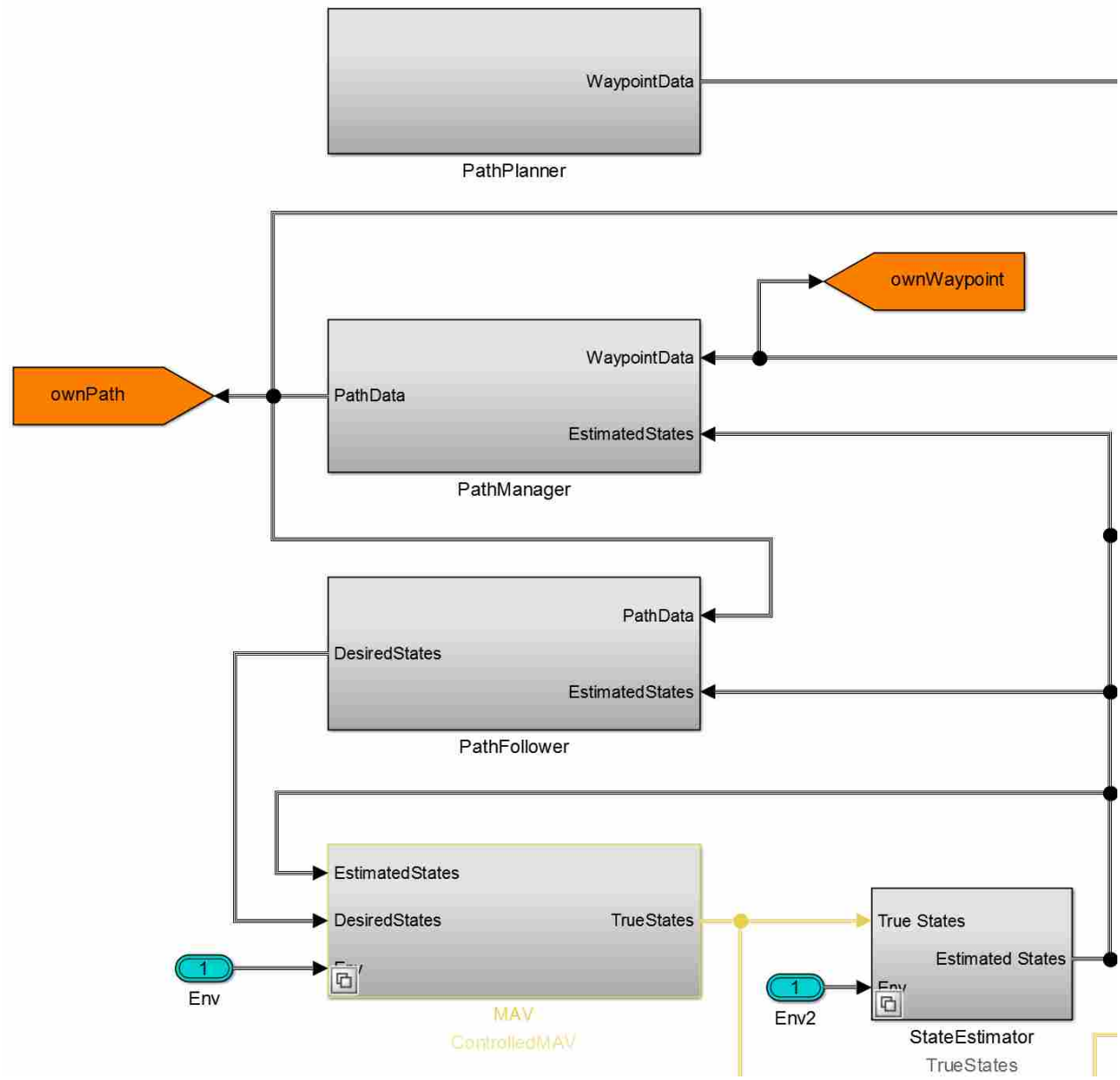


Figure C.2: Autopilot structure used for the ownship and intruder aircraft.

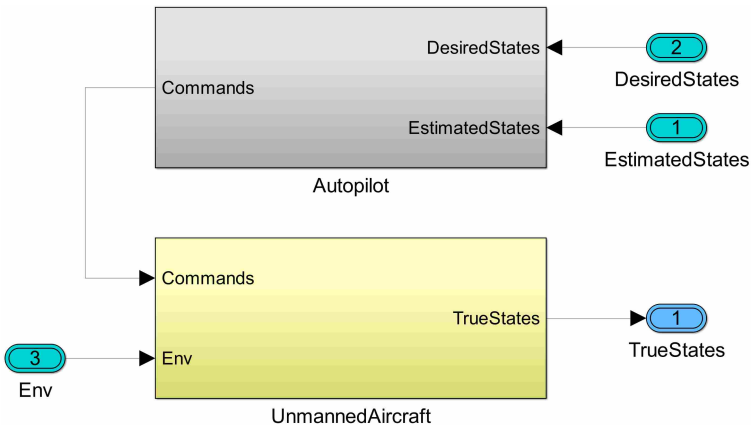


Figure C.3: Contents of MAV block.

For the ownship block, two additional subcomponents have been added as seen in Figure C.4. One of these subcomponents is shown by the red block, which is the target detection and tracking block. This block also has two subcomponents within it shown in Figure C.5. One is for the sensor model that is responsible for detecting the intruder aircraft, and the other is for the tracking algorithm that is responsible for estimating the states of the intruder aircraft. The second main subcomponent that has been added to the ownship block is shown by the green block. This block is the collision avoidance planner block, which contains a function that implements both the collision detection and collision avoidance algorithms.

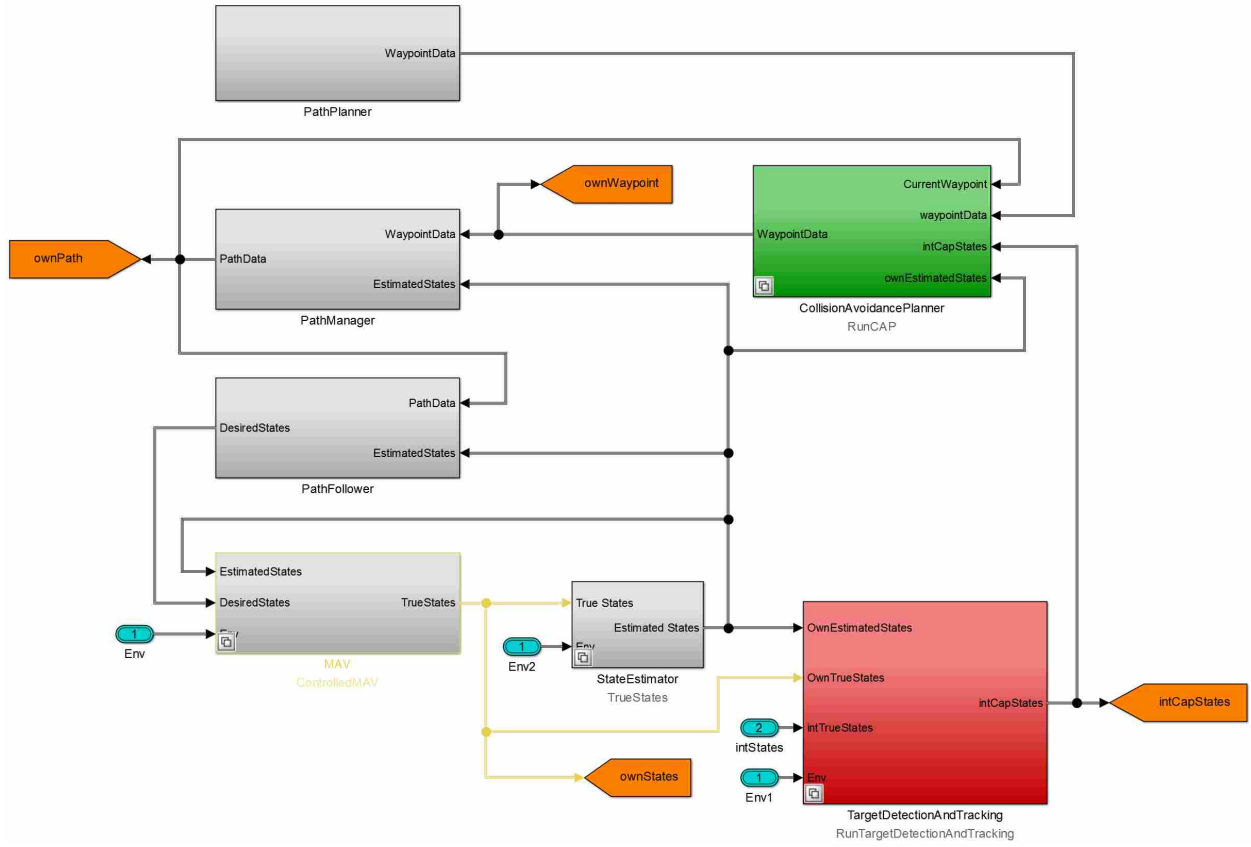


Figure C.4: All blocks used for the ownship including the target detection and tracking block and the collision avoidance planner block.

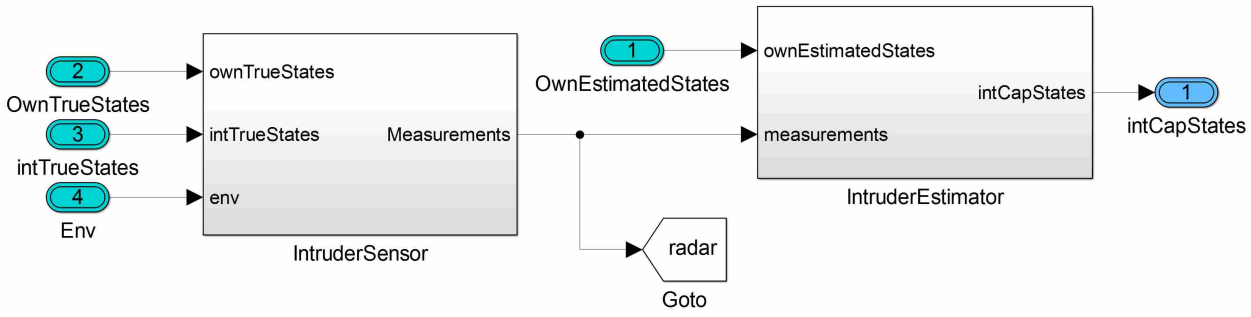


Figure C.5: Contents of TargetDetectionAndTracking block.

Within each of the subcomponents in this simulator, we have defined a generic configurable subsystem. An example of the contents of one of these subcomponents is seen in Figure C.6, where the configurable subsystem is shown by the large gray block.

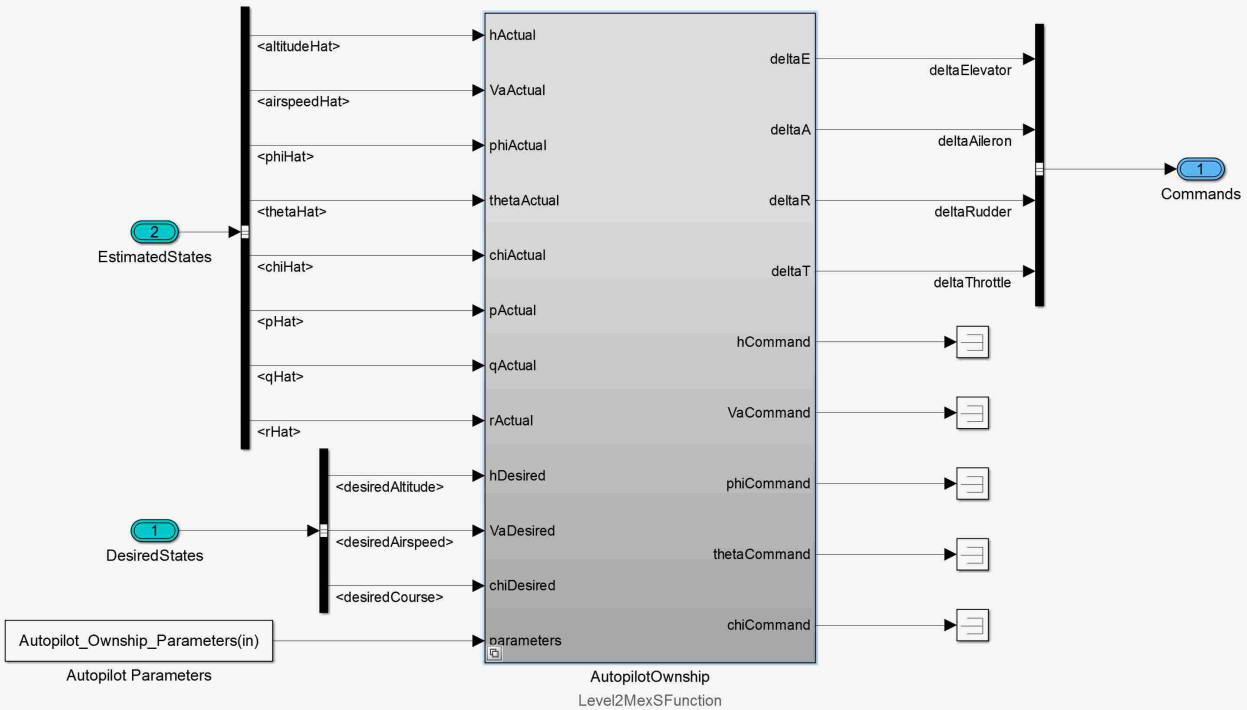


Figure C.6: Location of configurable subsystem within main Simulink block.

Each of the configurable subsystems must be stored in a user defined Simulink library model, as shown in Figure C.7. The first four blocks in this library contain an interpreted Matlab function, a level-1 Matlab s-Function, a level-2 Matlab s-Function, and a compiled level-2 C-MEX s-function, respectively. The fifth block is the configurable subsystem block, which selects between each of the previous four blocks based on user input.

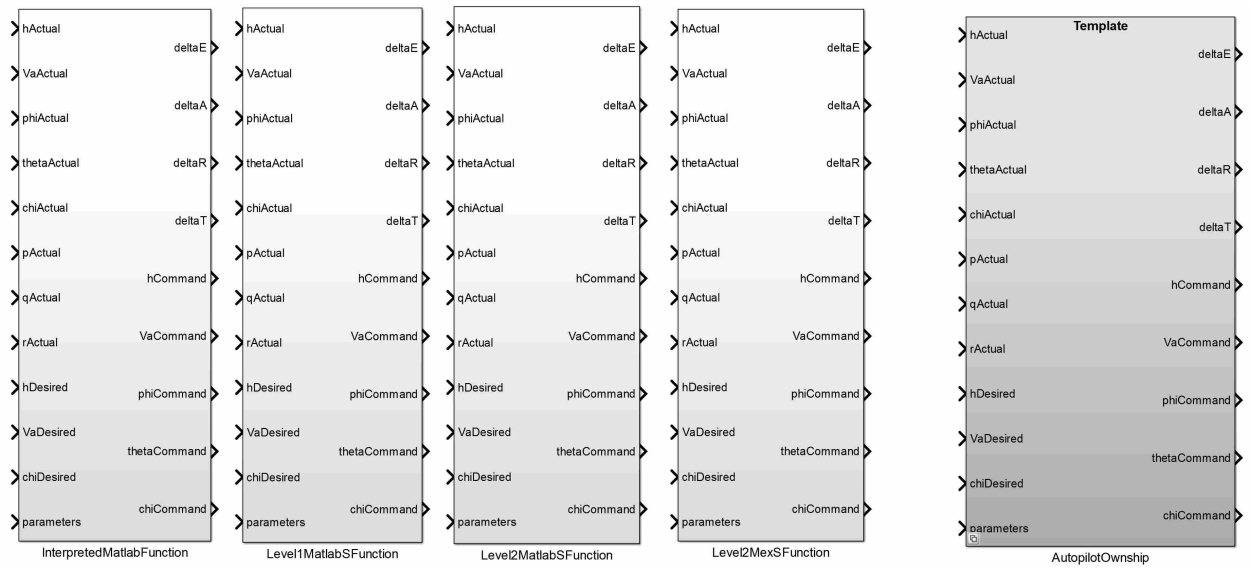


Figure C.7: Configurable subsystem defined in a Simulink library, used to choose between 4 function types.

This general purpose DAA simulator was used in the development of various simulation results shown throughout this thesis including the ER-RANSAC tracking results, and the complete ground-based radar DAA system. This simulator was a vital tool in moving from the simulated ground-based radar DAA system to the actual DAA flight test using a real radar hardware.