University of Cincinnati

Date: 3/17/2016

I, Nicholas P Hanlon, hereby submit this original work as part of the requirements for the degree of Doctor of Philosophy in Aerospace Engineering.

It is entitled:
Simulation Research Framework with Embedded Intelligent Algorithms for Analysis of Multi-Target, Multi-Sensor, High-Cluttered Environments

Student’s name: Nicholas P Hanlon

This work and its defense approved by:

Committee chair: Kelly Cohen, Ph.D.
Committee member: Sundararaman Anand, Ph.D.
Committee member: Manish Kumar, Ph.D.
Committee member: Bruce Walker, Sc.D.
Simulation Research Framework with Embedded Intelligent Algorithms for Analysis of Multi-Target, Multi-Sensor, High-Cluttered Environments

Nicholas P. Hanlon
College of Engineering and Applied Science
University of Cincinnati

A dissertation submitted for the partial fulfillment of the degree of Doctor of Philosophy in Aerospace Engineering & Engineering Mechanics

March 2016

Committee Chair: Dr. Kelly Cohen
Abstract

The National Air Space (NAS) can be easily described as a complex aviation system-of-systems that seamlessly works in harmony to provide safe transit for all aircraft within its domain. The number of aircraft within the NAS is growing and according the FAA, “on any given day, more than 85,000 flights are in the skies in the United States...This translates into roughly 5,000 planes in the skies above the United States at any given moment. More than 15,000 federal air traffic controllers in airport traffic control towers, terminal radar approach control facilities and air route traffic control centers guide pilots through the system”. The FAA is currently rolling out the Next Generation Air Transportation System (NextGen) to handle projected growth while leveraging satellite-based navigation for improved tracking. A key component to instantiating NextGen lies in the equipage of Automatic Dependent Surveillance-Broadcast (ADS-B), a performance based surveillance technology that uses GPS navigation for more precise positioning than radars providing increased situational awareness to air traffic controllers. Furthermore, the FAA is integrating UAS into the NAS, further congesting the airways and information load on air traffic controllers. The expected increase in aircraft density due to NextGen implementation and UAS integration will require innovative algorithms to cope with the increase data flow and to support air traffic controllers in their decision-making.

This research presents a few innovative algorithms to support increased aircraft density and UAS integration into the NAS. First, it is imperative that individual tracks are correlated prior to fusing to ensure a proper picture of the environment is correct. However, current approaches do not scale well as the number of targets and sensors are increased. This work presents a fuzzy clustering design to hierarchically break the problem down into smaller subspaces prior to correlation. This approach provides nearly identical performance metrics at orders of magnitude faster in execution. Second, a fuzzy inference
system is presented that alleviates air traffic controllers from information overload by utilizing flight plan data and radar/GPS correlation values to highlight aircraft that deviate from their intended routes. Third, a genetic algorithm optimizes sensor placement that is robust and capable of handling unexpected routes in the environment. Fourth, a fuzzy CUSUM algorithm more accurately detects and corrects aircraft mode changes. Finally, all the work is packaged in a holistic simulation research framework that provides evaluation and analysis of various multi-sensor, multi-target scenarios.
I dedicate this work to my wife, Jaime, who provided unconditional love and support throughout my educational endeavors and pursuits. This monumental achievement would not have been possible without your endless support and encouragement. Furthermore, to my children, Natalie, Penelope, and Chelsey, who inspired me to never quit.
First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Kelly Cohen. Words are not enough to say thank you for your never-ending encouragement, guidance, mentorship, and patience during my academic and professional career. My sincere thanks goes to my committee members, Dr. Manish Kumar, Dr. Bruce Walker, and Dr. Sundararaman Anand, for your invaluable input and recommendations to my research. Furthermore, thank you Dr. Cohen, Dr. Kumar, and Dr. Walker, not only for you support of my research but for fostering my interest in control theory and intelligent controls from the classroom that provided the foundation for this dissertation. I would also like to thank Dr. Elad Kivelevitch for his assistance and advice in my research. Finally, a special thanks to my family, parents, and parents-in-law for their support and words of encouragement.
## Contents

**Glossary** vii  
**List of Figures** ix  
**List of Tables** xii

### 1 Introduction 1  
1.1 Background ................................................. 1  
1.2 Goals ......................................................... 5  
1.3 Objectives ..................................................... 5  
1.4 Dissertation Structure ....................................... 6

### 2 Literature Survey 7  
2.1 Air Traffic Management Systems ............................... 8  
2.2 Sensor Fusion .................................................. 9  
2.3 Track-to-Track Cross-Covariance Term .......................... 10  
2.4 Track-to-Track Correlation ..................................... 10  
2.5 Fuzzy Logic ................................................... 11  
2.6 Clustering ....................................................... 12  
2.7 Mode Detection and Estimation ................................. 13  
2.8 Genetic Algorithms ........................................... 14

### 3 ARGOS 16  
3.1 Simulation Research Framework ............................... 16  
3.2 Analyst ........................................................ 17  
3.2.1 Scenario Definition ........................................ 17  
3.2.2 Simulation .................................................. 21  
3.2.3 Sensor Placement ............................................ 24  
3.2.4 Air Traffic Control ......................................... 24  
3.2.5 Analysis ..................................................... 27  
3.2.6 Documentation ............................................... 30  
3.3 Developer ....................................................... 31  
3.3.1 Model-View-Controller ..................................... 31  
3.3.2 Sequence Diagrams ......................................... 35
## 4 Problem Formulation

4.1 Targets ........................................................................ 38
  4.1.1 State ........................................................................ 38
  4.1.2 Trajectory ............................................................... 39
  4.1.3 Trajectory Change Points ........................................ 40
  4.1.4 Conformance .......................................................... 42

4.2 Tracking Filter ............................................................. 44

4.3 Sensor Models ............................................................. 44
  4.3.1 Radar Measurements .............................................. 44
  4.3.2 GPS Measurements ................................................. 46
  4.3.3 Sensor Failure ........................................................ 47

4.4 Track Fusion ............................................................... 47
  4.4.1 2 Track Fusion ....................................................... 47
  4.4.2 N Track Fusion ....................................................... 48
  4.4.3 Cross-Covariance Approximation ............................... 49

4.5 Likelihood Function ..................................................... 51
  4.5.1 2 Target Likelihood ............................................... 52
  4.5.2 N Target Likelihood ............................................... 53

4.6 Assignment Problem .................................................. 53
  4.6.1 2-D Assignment Problem ........................................ 53
  4.6.2 N-D Assignment Problem ....................................... 54

4.7 Sequential m-Best 2-D Assignment Algorithm ................. 55

4.8 Fuzzy Clustering ........................................................ 58

## 5 Algorithm Development

5.1 Air Traffic Fuzzy Inference System ............................... 60
  5.1.1 Cascaded Fuzzy Inference System ............................ 60
  5.1.2 Intent Conformance FIS ......................................... 61
  5.1.3 ATC Warning Level FIS .......................................... 64

5.2 Fuzzy Clustering Assisted Track-to-Track Correlation ........ 66

5.3 Fuzzy Mode Change Detection ...................................... 69
  5.3.1 Maximum Posterior Probability ............................... 69
  5.3.2 CUSUM .............................................................. 70
  5.3.3 Fuzzy CUSUM ....................................................... 71

5.4 Sensor Placement Optimization .................................... 74
  5.4.1 Genome and Chromosome Description ...................... 74
  5.4.2 Mutation ............................................................ 74
  5.4.3 Objective Function ............................................... 75
  5.4.4 Training ............................................................ 76

## 6 Results

6.1 ARGOS ........................................................................ 77
  6.1.1 Sensor Placement Performance ................................ 77
  6.1.2 Sensor Tracker Performance .................................... 79
Glossary

**Targets**

- $\rho$  
  1 Parameter - Approximate Cross-Covariance Coefficient
- $\rho_p$  
  3 Parameter - Approximate Cross-Covariance Coefficient - Position
- $\rho_\omega$  
  3 Parameter - Approximate Cross-Covariance Coefficient - Turn Rate
- $\rho_v$  
  3 Parameter - Approximate Cross-Covariance Coefficient - Velocity
- $\hat{P}_{ij}$  
  Cross-Covariance (Approximate) between State Estimates $\hat{x}_i$ and $\hat{x}_j$
- $P_{ij}$  
  Cross-Covariance (Exact) between State Estimates $\hat{x}_i$ and $\hat{x}_j$
- $P_i$  
  Error Covariance of Target $i$
- $P_{f,2}$  
  Fused Covariance of 2 Tracks
- $P_{f,N}$  
  Fused Covariance of N Tracks
- $\hat{x}_{f,2}$  
  Fused State Estimate of 2 Tracks
- $\hat{x}_{f,N}$  
  Fused State Estimate of N Tracks
- $\hat{x}_{GPS,i}$  
  GPS-based State Estimate of Target $i$
- $N_t$  
  Number of Targets
- $p(T_t)$  
  Probability that Target $t$ is included in the Training Dataset
- $\hat{x}_i$  
  State Estimate of Target $i$
- $x_i$  
  True State of Target $i$

**Sensors**

- $x_m$  
  GPS-based Measured Position in x-axis
- $y_m$  
  GPS-based Measured Position in y-axis
- $z_{GPS}$  
  GPS-based Measurement
- $N_s$  
  Number of Sensors
CONTENTS

\( p(F_s) \) Probability of Sensor Failure

\( \theta_m \) Radar-based Measured Azimuth

\( r_m \) Radar-based Measured Range

\( z_R \) Radar-based Measurement

**Genetic Algorithm**

\( p_m(n) \) Probability of Mutation at the \( n^{th} \) Bit

**Correlation**

\( \mathcal{J}_{g,k} \) \( k^{th} \) Tuple Solution within the the \( g^{th} \) Assignment Problem Iteration

\( \mathcal{J}_{g}^p \) \( p^{th} \) Ranked Set of Tuples \( \mathcal{J}_{g,k} \) within the the \( g^{th} \) Assignment Problem Iteration

\( \Lambda(\mathcal{J}) \) Likelihood Function

**Miscellaneous**

\( T_{i}^{j} \) Track File of \( i^{th} \) Target by the \( j^{th} \) Sensor

\( \mathcal{T} \) Vector of Track Files
List of Figures

1.1 U.S. National Airspace ........................................... 2
2.1 Crossover and Mutation Operators ................................. 15
3.1 ARGOS High Level View ............................................ 18
3.2 ARGOS Scenario Definition User Interface ...................... 19
3.3 Sensor Setup User Interface ..................................... 19
3.4 Airport Setup User Interface ................................... 20
3.5 Target Setup User Interface .................................... 20
3.6 Target Position/Velocity MSE .................................... 22
3.7 Sensor Position/Velocity MSE ................................... 23
3.8 IMM Results ...................................................... 23
3.9 Sensor Placement via Genetic Algorithm ......................... 24
3.10 ATC Fuzzy Inference System Input Values ..................... 26
3.11 ATC Warning Level ............................................. 26
3.12 List of Data Sets for Comparative Analysis ..................... 27
3.13 2-Dataset Comparison ......................................... 29
3.14 2-Dataset Plot Comparison ................................... 29
3.15 N-Dataset Comparison ........................................ 30
3.16 Standard Model-View-Controller ................................. 31
3.17 Modified Model-View-Controller ................................. 32
3.18 ARGOS Model-View-Controller ................................. 33
3.19 Graphical User Interface Compilation ....................... 35
3.20 Simulate Sequence Diagram ................................ 36
3.21 Sensor Placement via GA Sequence Diagram .................. 37
4.1 Target Trajectory with Way-points ................................ 41
4.2 Standard Turn Types ............................................ 41
4.3 Conforming Target ............................................... 42
4.4 Non-Conforming Targets ....................................... 43
4.5 Flight Comparison of Southwest FL2362 ......................... 43
4.6 Sensor Measurement Uncertainty ................................ 46
4.7 Sequential mBest 2-D Assignment Algorithm Flow ............. 56
5.1 Cascaded ATC Warning Level FIS ................................ 61
# LIST OF FIGURES

5.2 TCP and TCP+1 Angles ............................................. 62
5.3 Intent Conformance FIS Input 1 ................................. 62
5.4 Intent Conformance FIS Input 2 ................................. 63
5.5 Intent Conformance FIS Output .................................. 63
5.6 Intent Conformance FIS Surface ................................. 64
5.7 ATC Warning Level FIS Input 1 ................................. 65
5.8 ATC Warning Level FIS Input 2 ................................. 65
5.9 ATC Warning Level FIS Output .................................. 65
5.10 ATC Warning Level FIS Surface ............................... 66
5.11 State Estimate Distribution ..................................... 67
5.12 3 Cluster .......................................................... 67
5.13 Initial Clustering ................................................... 68
5.14 Augmented Clusters ............................................... 68
5.15 Clustering and Assignment Flow Example ..................... 69
5.16 Uncertainty in Maneuver Timing ................................. 72
5.17 Fuzzy CUSUM Input Membership Functions .................... 73
5.18 Fuzzy CUSUM Output Membership Functions ................... 73
5.19 Probability of Mutation .......................................... 75

6.1 Sensor Placement Optimization Process Flow ................... 78
6.2 Sensor Placement Optimization Analysis ......................... 79
6.3 Use Case Tracker Map ............................................. 80
6.4 Simulation Results of Varying Trackers .......................... 81
6.5 Varying Trackers Data Comparison ............................... 81
6.6 IMM Tracker Performance ......................................... 82
6.7 Tracker Comparison ............................................... 82
6.8 Scalability .......................................................... 84
6.9 Analysis of MSE Improvement vs Number of Sensors .......... 85
6.10 Diminished Performance Gain with Increased Number of Sensors 85
6.11 Comparative Sensor Failure Analysis ............................ 86
6.12 Conforming ATC Plot .............................................. 87
6.13 Conforming ATC FIS Inputs ....................................... 88
6.14 Conforming ATC Warning Level .................................. 88
6.15 Delayed Conformance ............................................. 89
6.16 Non-Conforming (Diverted Route) ATC Plot .................... 90
6.17 Non-Conforming (Diverted Route) ATC FIS Inputs ............ 90
6.18 Non-Conforming (Diverted Route) ATC Warning Level ....... 91
6.19 Non-Conforming (Spurious Intent) ATC Plot ................... 92
6.20 Non-Conforming (Spurious Intent) ATC FIS Inputs .......... 92
6.21 Non-Conforming (Spurious Intent) ATC Warning Level ....... 93
6.22 Combined Scenarios ATC Plot ................................... 94
6.23 Combined Scenarios ATC Warning Level ......................... 95
6.24 Case A Performance Results ..................................... 96
6.25 Case B Performance Results ..................................... 96
<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.26</td>
<td>Case C Performance Results</td>
<td>96</td>
</tr>
<tr>
<td>6.27</td>
<td>Case D Performance Results</td>
<td>97</td>
</tr>
<tr>
<td>6.28</td>
<td>Target Trajectory with 3 Way-Points</td>
<td>98</td>
</tr>
<tr>
<td>6.29</td>
<td>CV Posterior Model Probability</td>
<td>99</td>
</tr>
<tr>
<td>6.30</td>
<td>S Time Series</td>
<td>100</td>
</tr>
<tr>
<td>6.31</td>
<td>S Time Series False Positive Analysis</td>
<td>100</td>
</tr>
<tr>
<td>6.32</td>
<td>Mode Change Detection</td>
<td>100</td>
</tr>
<tr>
<td>6.33</td>
<td>Mode Change Time Prediction</td>
<td>101</td>
</tr>
<tr>
<td>6.34</td>
<td>Way-Point 1 Mode Time Detection</td>
<td>102</td>
</tr>
<tr>
<td>6.35</td>
<td>Way-Point 1 Mode Time Correction</td>
<td>103</td>
</tr>
<tr>
<td>6.36</td>
<td>Way-Point 2 Mode Time Detection</td>
<td>103</td>
</tr>
<tr>
<td>6.37</td>
<td>Way-Point 2 Mode Time Correction</td>
<td>104</td>
</tr>
<tr>
<td>6.38</td>
<td>Way-Point 3 Mode Time Detection</td>
<td>104</td>
</tr>
<tr>
<td>6.39</td>
<td>Way-Point 3 Mode Time Correction</td>
<td>105</td>
</tr>
<tr>
<td>6.40</td>
<td>Sensor Placement Locations based on Three Different Scenario Configurations</td>
<td>107</td>
</tr>
<tr>
<td>6.41</td>
<td>Sensor Placement Locations for Minimize Average Target MSE</td>
<td>108</td>
</tr>
<tr>
<td>6.42</td>
<td>Sensor Placement Locations for Minimize Maximum Target MSE</td>
<td>109</td>
</tr>
<tr>
<td>7.1</td>
<td>ARGOS Version 2.0</td>
<td>113</td>
</tr>
</tbody>
</table>
List of Tables

3.1 ATC Warning Level Categories .................................................. 25
4.1 2-D Assignment Problem (1st iteration) ......................................... 57
4.2 2-D Assignment Problem (2nd iteration) ......................................... 58
5.1 Intent Conformance Rule Base .................................................... 63
5.2 ATC Warning Level Rule Base ................................................... 64
5.3 Example Partition Matrix .......................................................... 67
6.1 Use Case Sensor Placement Optimization ....................................... 78
6.2 Tracker Performance Comparison ............................................... 81
6.3 Comparison of Scenarios with Sensor Failure .................................. 86
6.4 Conforming ATC - Time in ATC Warning Level Categories .................. 87
6.5 Non-Conforming (Diverted Route) ATC - Time in ATC Warning Level Categories .......................................................... 89
6.6 Non-Conforming (Spurious Intent) ATC - Time in ATC Warning Level Categories .......................................................... 91
6.7 Combined ATC - Percentage Breakdown of ATC Warning Level Categories .......................................................... 94
6.8 Fuzzy Clustering Assisted Track-to-Track Correlation Test Cases .......... 95
6.9 Algorithm Comparison .............................................................. 97
6.10 Ratio of Optimal Number of Clusters to the True Number of Targets .... 97
6.11 Mode Time Detection - Single Run .............................................. 99
6.12 Mode Time Detection Correction - Single Run ............................... 101
6.13 Mode Detection Discrepancy $CV - CT$ ......................................... 105
6.14 Mode Detection Discrepancy $CT - CV$ ......................................... 106
6.15 Mode Correction Discrepancy $CV - CT$ ......................................... 106
6.16 Mode Correction Discrepancy $CV - CT$ ......................................... 106
6.17 False Positive Detection Rate ..................................................... 106
6.18 Probability of Target Run per Case ............................................. 107
6.19 MSE per Target for each Test Case - Minimize Average MSE .............. 108
6.20 MSE per Target for each Test Case - Minimize Maximum MSE ............ 109
7.1 ARGOS Version Development Plan .............................................. 117
“On any given day, more than 85,000 flights are in the skies in the United States”

FAA

1

Introduction

1.1 Background

The National Air Space (NAS) can be easily described as a complex aviation system-of-systems that seamlessly works in harmony to provide safe transit for all aircraft within its domain. The number of aircraft within the NAS is growing and according the FAA, “On any given day, more than 85,000 flights are in the skies in the United States...This translates into roughly 5,000 planes in the skies above the United States at any given moment. More than 15,000 federal air traffic controllers in airport traffic control towers, terminal radar approach control facilities and air route traffic control centers guide pilots through
1. INTRODUCTION

the system(1)”. Figure 1.1 provides a glimpse of the air traffic within the NAS. Radar technology has remained as the backbone to support the 21 Air Route Traffic Control Centers (ARTCCs) scattered through the United States. This underlying network infrastructure inadvertently forces aircraft into inefficient flight paths, subsequently leading to longer routes and increased fuel consumption.

Figure 1.1: U.S. National Airspace

The third generational air traffic control system, Next Generation Air Transportation System (NextGen), is being rolled out in stages from 2012 through 2025. NextGen is a paradigm shift in air navigation by switching from the established ground-based systems (i.e., radar systems and beacons) to satellite-based systems. Peter Appel, administrator of Research and Innovative Technology Administration of DOT, stated “NextGen GPS technology will be used to shorten routes, save time and fuel, reduce traffic delays, increase capacity, and permit controllers to monitor and manage aircraft with greater safety margins (2)”. A key component to instantiating NextGen lies in the equipage of Automatic Dependent Surveillance-Broadcast (ADS-B), a performance based surveillance technology
that uses GPS navigation for more precise positioning than radars. Thus, air traffic controls and other aircraft have increased situational awareness allowing for closer separation. The FAA ruling (3) requires equipage for aircraft operating in airspace classes A, B, and C, certain Class E airspace, and other specified airspace.

The UAV world has seen tremendous growth in the past decade due to emerging technologies and economies of scale. Beyond military use, UAV research and testing is being evaluated for many different areas, including local government, emergency services, and commercial. UAVs have been researched to provide enhanced situational awareness during wild land fires (4, 5, 6). Amazon has been researching/testing delivery packages within 30 minutes, aptly named Amazon Prime Air (7). Both Google (8) and Facebook (9) are researching the use of UAVs to provide internet access to areas of the world that lack internet connectivity. UAVs can also provide an inexpensive method for real estate to capture aerial images of property, as opposed to financing helicopters or cranes (10).

On February 15, 2013, the Subcommittee on Oversight held a hearing (11) to discuss UAS integration to ensure safety, stating in their records, “The Teal Group, an aerospace and defense industry market intelligence firm, forecasts worldwide annual spending on UAS research, development, testing, and evaluation (RDT&E) activities and procurement rising from $6.6 billion in 2013 to $11.4 billion in 2022. Total worldwide spending for the period is forecast to amount to $89.1 billion. Throughout the forecast period, Teal expects the U.S. share of RDT&E to account for 62 percent of worldwide spending, while U.S. procurement will amount to 55 percent of worldwide spending”. In response, the U.S. Department of Transportation’s Federal Aviation Administration (FAA) released (12) its roadmap (13) on November 7, 2013 for UAS integration into the National Airspace System (NAS).

The challenge is fully equipping aircraft with ADS-B technology, and in particular, the UAV market since it incurs both additional cost and weight. There will be issues as the NextGen system along with UAS integration will further congest the NAS. Pilots
and air traffic controllers risk becoming overly dependent on the ADS-B system, potentially ignoring other aircraft. Similarly, a system failure or intentional act may render the ADS-B nonfunctional. The FAA roadmap explicitly references the threats to national security, stating in section 1.4.5, “Integrating public and civil UAS into the NAS carries certain national security implication, including security vetting for certification and training of UAS-related personnel, addressing cyber and communications vulnerabilities, and maintaining/enhance air defense and air domain awareness capabilities in an increasingly complex and crowded airspace.” A research team at the University of Texas demonstrated the capability of electronically hijacking a UAV’s GPS system and spoofing its GPS receiver “...even as it steers a new navigational course induced by the outside hacker” (14). Researchers have explored the vulnerabilities and threats of ADS-B technology and its integration with the NextGen system (15, 16). The GAO stated that the Department of Homeland Security has “not properly examined nor identified specific steps to mitigate potential security threats posed by routine UAS access to the national airspace (11).” Malaysia Airlines Flight 370 disappeared on March 8, 2014 after the aircraft’s transponder stopped transmitting flight details and officials relied on radar pings to assist to narrowing the search and rescue operations. As of today, it is not known if the disappearance of the aircraft was due to nefarious reasons or system malfunction, but the loss of aircraft communication is real. Thus for the indefinite future, air traffic control (ATC) must continue to rely on ground-based radar systems as a supporting factor in air traffic management.

The FAA does provide Radar Traffic Information Service (according to the FAA Aeronautical Information Manual (AIM), Chapter 4, Section 4-1-15) for the purpose of "assisting and advising a pilot that a particular radar target’s position and track indicates it may intersect or pass in such proximity to that pilot’s intended flight path that it warrants attention (17)." The caveat to this service lies in the human interface, as explicitly stated in the provision section that traffic volume, workload, and communications
frequency congestion may prevent the controller from providing timely instruction.

1.2 Goals

This goals of this dissertation are two-fold:

- Develop a stable, flexible software tool for two distinct users: (a) analysts to test various algorithms and analyze their performance (b) developers to easily modify and extend the software tool functionality.

- Develop efficient and effective innovative algorithms that support enhancing situational awareness for air traffic controllers.

1.3 Objectives

The following objectives provide more concrete details into how the goals are met:

- Construct the software research tool using the software architectural pattern model-view-controller to support modular development and future tool expansion of other algorithms.

- Develop a cascaded fuzzy inference system using target trajectory flight path and radar/GPS chi-square correlation value to numerically prioritize targets for air traffic controller to monitor.

- Develop a hierarchical approach to track-to-track correlation that decomposes the problem space into smaller sub-problems to leverage fuzzy clustering properties, providing similar performance in correlation accuracy at 1-2 orders of magnitude faster in execution speed.

- Develop a fuzzy inference system to improve recognition of timing target mode transition.
1. INTRODUCTION

- Develop a robust sensor placement optimization based on genetic algorithms

1.4 Dissertation Structure

The layout of this dissertation is as follows. Chapter 2 provides a literature survey of the previous and state-of-the-art approaches to track-to-track correlation. Chapter 3 presents ARGOS, the simulation research tool that supports algorithm analysis. Chapter 4 describes the underlying models within ARGOS for target trajectories, target tracking track fusion, and track-to-track correlation. Chapter 5 lists four new intelligent algorithms using fuzzy logic and genetic algorithms to extend previous work. Chapter 6 contains results of ARGOS analysis and the new algorithmic development. Chapter 7 concludes the thesis with observations and recommendations for future work.
"The person who says it cannot be done should not interrupt the person who is doing it."

Chinese Proverb

Multi-sensor multi-target environments is a research area of great interest to both commercial and military applications. Tutorials and overviews outlining high-level concepts of various components such as centralized/decentralized sensor systems, target tracking, data fusion, decision-making, etc. can be found at (18, 19, 20, 21, 22, 23). The following sections provide a more in depth review of the state-of-the-art topics that relate to this dissertation.
2. LITERATURE SURVEY

2.1 Air Traffic Management Systems

Air traffic management (ATM) is the overarching theme of interconnected systems, consisting of systems such as Air Traffic Control, Air Traffic Services, Air Traffic Flow and Capacity Management, that aids aircraft in all aspects of travel from terminal to terminal. The significant rise in air traffic density forces ATM to leverage powerful and intelligent systems, relieving human operators from the monotonous, mundane tasks to focus on urgent issues that need to addressed. Many products have been developed by commercial entities in an attempt to support the ATM growth and transition to the NextGen system. Raytheon developed the AutoTrac Air Traffic Management System (24) that uses accurate Kalman Filter multi-sensor fusion tracker and displays the data in a graphical interface for increased human operator productivity. Telephonics developed AEROTRAC (25), to provide surveillance capacity, tracking accuracy, and flight information processing for the proposed NextGen system. Similarly, Boeing has produced its own suite of ATM products, in particular, the Advanced Air Traffic Management (26) system, to support the transition to the NextGen system. The limitation in each is the reliance on successful implementation of the NextGen system and ADS-B technology on all aircraft (commercial, UAS, etc). In addition, the underlying foundation is not built to evaluate and analyze new innovative algorithms for research purposes.

Research tools have been previously developed in the past to explore data association. FUSEDAT (27) is a software package for fusion, data association, and tracking with multiple sensors. This software focuses primarily on the measure-to-track association problem and uses a centralized data fusion architecture. MATSurv (Multisensor Air Traffic Surveillance) (28) was developed to perform track-to-track association based on actual data from two FAA radars with multi-modality sensor systems. This work was limited to two sensors simplifying the assignment algorithm calculations.
2.2 Sensor Fusion

Incorporating multiple sensors into an environment provides several benefits, such as: (a) adds redundancy in case of system failures, and (b) minimizes tracking errors by fusing the estimates together. Fusing track estimates together can be accomplished by several different approaches (e.g., track-to-track, measurement-to-track, Maximizing a Posterior, Sequential Kalman Filter, etc.) and choice of methodologies is subjective to the objectives and constraints applicable to the underlying architecture (centralized versus decentralized). This work employs the track-to-track structure with a fusion center that fuses the set of tracks into an overall target estimate.

Given a set of $N$ tracks, an estimate is achieved by fusing the tracks together based on the maximum likelihood criterion (29, 30). The formulation is detailed in section 4.4.2. It is important to note that the drawback is the multiple matrix inversions required in the formulation. For a small number of tracks, this is not a problem, but becomes computationally expensive as the number of tracks increase.

Fong (31, 32) proposed the simplified maximum likelihood algorithm for track-to-track fusion by composing a block diagonal matrix of covariance matrices where each block is the covariance matrix of the two-sensor fusion, completed for all combinations of sensor tracks. Although matrix inversion is not avoided, the approach reduces the computational load since the $N$ sensor fusion is inverting a block diagonal matrix. Fong demonstrated the simplified technique to have similar results in numerical performance compared to (29). Nevertheless, the gain in computation time by only having to invert a block diagonal matrix is lost due to the need to compute the covariance matrices from all combinations of two sensor fusions, each inherently requiring matrix inversions in itself.
2. LITERATURE SURVEY

2.3 Track-to-Track Cross-Covariance Term

In multi-sensor problems, it is imperative to fuse individual sensor tracks together to create a common picture of the environment for the decision makers. Irrelevant to the problem space (military, FAA, etc.), having an accurate picture of the number of targets improves the decision makers ability to make decisions. Prior to fusing multiple tracks, the question must be answered whether tracks from disparate sensors represent the same truth object. Initial work (33) assumed that the estimation errors of track files were independent when testing tracks for a common origin. This assumption has since been disproved; the estimation errors are correlated since the process noise from both models is incorporated into the estimate (34). The measurement noise from the sensors does not play any significance in the correlation test (35).

2.4 Track-to-Track Correlation

Unfortunately, researchers have not used consistent terminology regarding association and correlation in the domain of target tracking. For purposes of this work, track association is referred to as the association of measurements to a track, whereas track correlation is the association of multiple tracks that represent the same truth object.

The problem of track-to-track correlation has been studied extensively over the past few decades; a few references can be found at (36, 37, 38, 39, 40, 41), a concept originally proposed by Singer in (33). When the number of sensors exceeds two, the S-D assignment formulation for the track-to-track correlation problem becomes NP-Hard even under the assumption of unity detection probability and no spurious measurements(42).

Furthermore, literature has comparatively explored the difference between a single time test versus a sliding window test. The single time test simply runs the test statistic on the most current estimates. The sliding window test uses multiple estimates to test if the null hypothesis is correct. The sliding window hypothesis was tested by You (43)
to show the strength of multiple estimates. However, Bar-Shalom promptly pointed out flaws in the derivation. Bar-Shalom (44, 45) says that (43) assumed the estimation errors in the time series are white noise, however, in fact, the estimation errors are correlated, and one must account auto-correlation in the sliding window approach. Interestingly, and counter-intuitively (45) concludes, is the power of the test statistic for the single time test is stronger than sliding window, for when process noise is low. However, (45) states that as process noise increases, the sliding window approach appears more desirable.

2.5 Fuzzy Logic

Zadeh(51) originally published his fuzzy set theory in 1965. Conventional control theory is based on mathematical models of differential equations that result in a discrete, binary value of 0 or 1 (true or false). However, these models lack the human intuition that processes are not always 0 or 1. Rather than the binary logic of conventional controllers, fuzzy logic controllers are multi-valued based on degree of membership which better represent the state of the process. Fuzzy logic sets become an ideal tool for complex systems for generating the control rules (46). Lewis (47) states that fuzzy control is the union of three different disciplines: conventional control theory, artificial intelligence, fuzzy set theory (in particular, approximate reasoning and linguistic variables). Although fuzzy logic does not require a mathematical model of the plant, the detailed structure and evaluation criteria remains the same as conventional control theory. Engineers must still understand the system behaviors and collect the expert knowledge to build the linguistic rules of the system. Fuzzy controllers are composed of four main elements (48):

1. Rule Base: The rule base is a series of If-Then-Else statements in linguistic nature that describes the experts knowledge of the system.

2. Inference Mechanism: The inference mechanism serves two main purposes:

   (a) Matching: Determine to what extent that each rule is applicable in the system.
2. LITERATURE SURVEY

There is a tendency to characterize fuzzy logic as probability, yet from a time point-of-view, these two concepts are different. Probability measures the likelihood that an event will occur in the future as opposed to fuzzy logic which measures the ambiguity of past events.

(b) Inference Step: Conclude which actions to take based on the rules determined to apply at the given time.

3. Fuzzification Interface: Fuzzification converts the crisp, numeric input values into fuzzy sets that the inference mechanism can then apply the rules.

4. Defuzzification Interface: Defuzzification is the result of evaluating the rule base to create a fuzzy set that is converted into crisp output values.

2.6 Clustering

Recognizing the impracticality of MHT (and others) in large scale problems, authors have proposed decomposing the problem space into clusters, reducing the dimensionality of the problem by deceasing the number of candidate associations. Partitioning the problem space introduces the potential for sub-optimal results but the benefit of real-time execution supersedes the cost. The ideal cluster contains only those track files that belong to the same truth target. However, that requires truth knowledge of the number of targets in the environment and well spaced targets. In a similar problem, measurement-to-track, where one associates sensor measurements to a track file, Chummumu(49) and Konstantinova(50) proposed a clustering approach to minimize the solution space for faster processing time. Chummumu recognized the issue of overlapping track files and extended the work by expanding the cluster size to superclusters, which encapsulates other clusters to evaluate additional sensor measurements. Both works achieved faster execution times from slight improvements to an order of magnitude.
Mode Detection and Estimation

This concept is extended to fuzzy clustering such that data can belong to multiple clusters with a degree of membership to the cluster. A summary of the key variables of the fuzzy c-means algorithm for this work is provided here but interested readers are directed to (52, 53) for a more extensive overview of the mathematical formulation. The partition matrix is the calculated degree of membership for each observation to each cluster. At this time, no work has been discovered that leverages partition matrix to improve execution time in track-to-track correlation.

2.7 Mode Detection and Estimation

Radar tracking has been a heavily researched topic over the past few decades. The original Kalman Filter provided a huge leap forward in estimation theory by demonstrating the ability to more precisely estimate a target state in the presence of process and sensor noise. This technology has evolved to more complex track architectures such as the extended Kalman Filter, unscented Kalman Filter, Multiple Hypothesis Tracker, and Interacting Multiple Model (IMM) Tracker. The IMM tracker utilizes multiple motion (or behavior) models with the ability to switch from one behavior model to another through probabilistic means. Its popularity is its low computational cost, nearly linear to the number of models, with nearly the same performance as an algorithm with quadratic complexity (23). The IMM estimator is an ideal state estimator for ATC, compared to a single model Kalman Filter, as it produces smaller root mean square predictions errors and associates more measurements to existing tracks (54).

Jilkov (55) outlined the early designs to mode detection. The simplest design is to determine aircraft flight mode by evaluating the posterior probability of the IMM. Therefore, the projected mode of the aircraft is the mode with the largest posterior probability. However, this is prone to random fluctuations, measurement noise, process noise, and false alarms. It is observed that during steady modes of the system, one mode dominates
2. LITERATURE SURVEY

other modes until a jump in the system occurs and another mode takes on the dominant position. A natural extension to the simple design described above is check if the current dominant mode has fallen below a specified dominance threshold. Once this condition is met, a mode change is acknowledged and the new dominant mode is computed by finding the mode that exceeds the dominance threshold. Jilkov states two flaws of these heuristic approaches: (a) it does not consider the statistical properties of the posterior probability time series, and (b) it provides no estimation of when the change occurred. Therefore, Jilkov presented the cumulative sum (CUSUM)-type statistical test to address the two flaws with improved mode detection along with mode change occurrence through post-processing. Lowe (56) leveraged the design by Jilkov and described the approach for purposes of predicting pilot intent and aircraft trajectory in uncontrolled airspace.

2.8 Genetic Algorithms

The genetic algorithm (GA) is a search heuristic used for solving combinatorial optimization problems where the underlying formulation is based on biological evolution and natural selection. Genetic algorithms are well suited for non-standard optimization problems where the “objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear (57).” But the simplicity comes at a cost that the algorithm tends to be computationally expensive (58). Knowledge of the problem space, proper balance of exploitation versus exploration, appropriate parameter settings of the genetic operators, and a suitable fitness function can improve converge time.

The algorithm consists of four distinct stages: initialization, selection, reproduction, and termination. The initialization stage creates a pool of initial solutions, or chromosomes. The size of the pool is dependent of the problem at hand but usually has a range from hundreds to thousands of chromosomes.

Following each generation, a selected number of chromosomes are chosen to breed
the next generation of chromosomes. The selection process assigns each chromosome a fitness value which dictates its probability of being selected for the next generation. Fitter chromosomes have a higher probability than less fit chromosomes. Although the fitter chromosomes are more likely to be selected to breed, a few less fit chromosomes may be selected to breed in the next generation. This feature ensures diversity among the pool and prevents the chromosomes from converging too quickly on a local minimum rather than the global minimum.

Once the chromosomes have been selected, they undergo the genetic operators of crossover and mutation in the reproduction stage. In crossover, two random chromosomes are selected as parents to breed new offspring that carry some traits of its parents. The crossover process will randomly select a position $k$ in the chromosome and exchange the gene information between the parents from the position $k$ to the end of the chromosome to form two new chromosomes. In mutation, since each bit in the chromosome is either a 0 or 1, mutation is the process of randomly flipping a bit from 0 to 1 or vice versa. The common approach is to assign each bit a certain chance of mutation. Figure 2.1 depicts both the crossover and mutation processes.

![Figure 2.1: Crossover and Mutation Operators](image)

The selection and reproduction stages are repeated until a satisfying termination point has been reached. This may be indicated by a viable solution, a fixed number of iterations, the level of improvement is no longer significant, or any combination of terminating criteria.
3.1 Simulation Research Framework

Chapter 2 listed examples of modeling and simulation frameworks that range from academic to industry. Acquiring industry-based modeling and simulation software packages for research purpose are difficult to obtain; they may not be available for academic use or the financial cost is too large. Other academic tools are specific designs with only a few primary objectives to explore. The goal of this work is to construct a modeling and simulation framework that can address fundamental research questions in sensor management in a holistic manner with flexibility for various plug-n-play algorithms. The simulation
framework is built for purposes of two users: (a) analyst and (b) developer. An analyst must be able to execute various simulations and analyze the effects of different configurations without a need to understand the underlying mathematical models. A developer must be able to easily add, modify, and manipulate the underlying models to allow for flexibility in the simulation.

3.2 Analyst

The front end view of ARGOS is designed to provide intuitive navigation and ease of use of the tool. The next few sections provides additional details of the various user interfaces (UI).

3.2.1 Scenario Definition

The fundamental configuration within ARGOS is defined by a scenario, constructed within the scenario definition UI. A scenario comprises of region size, simulation length, airport configuration (optional), sensor configuration, and target configuration. The sensor configuration data defines the sensor position, measurement uncertainty, model fidelity, and tracking parameters. The target configuration data defines the target’s initial position, velocity, and conformation to the broadcasted state. All subsequent operations within ARGOS (i.e., simulation, air traffic controller warning system, sensor placement optimization, track-to-track correlation, etc.) are based on the scenario parameters. Users may then save the scenario for later use. Figure 3.2 provides an example of the scenario definition UI with all relevant data already populated.
3. ARGOS

Figure 3.1: ARGOS High Level View

- Scalability
- RMS error
- Process noise
- Uncertainty
- Sensor
- Sensitivity
- Robustness
- Correct Association
- Probability of
- Performance Metrics

ARGOS

- Fuzzy Inference System
- Fuzzy C-Means Algorithm
- Sequential M-Most Assignment
- Generalized Likelihood
- S-D Assignment Algorithm
- Sensor Fusion

IMM
- Single Mode KF
- Trackers

Uncertainty
- Measurement
- Sensor Failure
- Probability of
- Number of Sensors

Probabilistic
- Conforming
- Conforming/Non-
- Density Level
- Maneuvering
- Managed
- Unmanaged
- Number of Targets

OUTPUTS

INPUTS
3.2 Analyst

Figure 3.2: ARGOS Scenario Definition User Interface

Figure 3.3: Sensor Setup User Interface
3. ARGOS

Figure 3.4: Airport Setup User Interface

Figure 3.5: Target Setup User Interface
3.2 Analyst

3.2.2 Simulation

The Simulation UI serves as a visual validation of the scenario, providing a real-time series progress of the target trajectories and state estimation over the specified simulation length. At the conclusion of the simulation, a variety of data points and graphs provide insight into the outcome of the simulation, listed below. The data points are averaged in the event that the number of runs is greater than one.

- **Graphical**
  - Overall Target Position & Velocity Mean Square Error (fig. 3.6)
    By default, the target position and velocity mean square error (MSE) are displayed at each time step at the conclusion of the simulation. The data represents the MSE between the fused state estimate and true target state.
  - Sensor Position & Velocity Mean Square Error (fig. 3.7)
    The user may view the positional and velocity MSE for each sensor by selecting the ‘Sensor Position/Velocity MSE’ option under Graph Options.
  - IMM Results (fig. 3.8)
    Sensors that track with an IMM tracker can view IMM performance. This option depicts the mode probability and turn rate parameters at each time step.

- **Numerical**
  - Overall Target Position & Velocity Mean Square Error
    Whereas the graphical depicted the MSE at each time step, the numerical table shows the overall MSE for each target.
  - Sensor Position & Velocity Mean Square Error
    Similar to the graphical result, this option displays the overall numerical MSE for each individual sensor over the entire simulation.
3. ARGOS

The numerical table provides a quick reference to the performance of each sensor and the enhanced target MSE by fusing the estimates together. For ease of use, the user may select any cell in the numerical table to filter the results on the screen. For example, selecting the Fused cell for Target ID 3 will display only that target in the environment and the graphical positional/velocity MSE for that target (fig. 3.6). Selecting any sensor cell for Target ID 3 will likewise, display only that target in the environment but show the graphical sensor positional/velocity MSE for that target (fig. 3.7).

The user can select different plot options to view the true target track, fused track, sensor tracks, and time steps. In addition, user may zoom, pan, and use the data cursor mode on any graph to gather further insight on the data. Finally, the user can save the results of the simulation for further analysis in the analysis section of the tool.

![Figure 3.6: Target Position/Velocity MSE](image-url)
3.2 Analyst

Figure 3.7: Sensor Position/Velocity MSE

Figure 3.8: IMM Results
3. ARGOS

3.2.3 Sensor Placement

The Sensor Placement Optimization UI provides the user the ability to optimize sensor positions. More importantly, to achieve the holistic approach, the user can update the scenario definition with the optimized sensor positions for further analysis. The supplied algorithm is a genetic algorithm optimization that provides instant updates on optimization progress and performance.

The user may also choose a limited set of sensors to optimize from the given list by checking the optimize toggle buttons next to each sensor. This is applicable when a predefined set of sensors are fixed in the environment. For additional flexibility in optimization and analysis, the user has the option of four objective functions (more details on the objective function are found in section 5.4.3).

![Sensor Placement via Genetic Algorithm](image)

Figure 3.9: Sensor Placement via Genetic Algorithm

3.2.4 Air Traffic Control

The Air Traffic Control UID is a numerical, graphical, and audio indication of aircraft adherence to flight routes and ABS-B broadcasted data points. Aircraft adherence is broken down into three categories, shown in table 3.1.
Table 3.1: ATC Warning Level Categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Default Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watch</td>
<td>[0, 0.33]</td>
<td>Aircraft state indicates normal operation with minimal perturbation from intended route.</td>
</tr>
<tr>
<td>Advisory</td>
<td>(0.3, 0.67]</td>
<td>Aircraft state indicates abnormal operation requiring occasional monitoring.</td>
</tr>
<tr>
<td>Warning</td>
<td>(0.67, 1.0]</td>
<td>Aircraft state indicates adverse operation requiring immediate attention and monitoring.</td>
</tr>
</tbody>
</table>

A fuzzy inference system using the angle discrepancy between the aircraft’s velocity vector and trajectory change points (TCP), as well as the $\chi^2$ distribution value between the sensor fused estimate and GPS-based tracking estimate, calculates a level of compliance in its flight route. The numerical output is placed in colored coded blocks for fast recognition of advisory and warning levels. The graphical plot displays colored circles along the trajectory to illustrate adherence. Finally, an audible tone is played based on the highest warning category that contains an aircraft (watch - no tone, advisory - low pitch tone, warning - high pitch tone). Finally, the threshold between each category is modifiable by the analyst for fine-tuning when aircraft are placed in different categories. Similarly, the analyst may view or edit the different fuzzy inference systems that compose the cascaded fuzzy inference system. A droplist shows applicable inference systems to use for the ATC simulation.

After the simulation is complete, the analyst can analyze the historical data points of the simulation, including the three input values into the ATC fuzzy inference system (fig. 3.10) and the warning level at each time step for each aircraft (fig. 3.11).
3. ARGOS

Figure 3.10: ATC Fuzzy Inference System Input Values

Figure 3.11: ATC Warning Level
3.2.5 Analysis

The strength of ARGOS is the analysis of various simulation runs for comparison purposes. The main analysis page lists all user saved data sets (file name, file created date, file meta data such as the number of sensors and targets, and a description of the data provided when the data was initially saved). The level of analysis is based on the number of files selected for comparison. It is imperative to note that the analysis tool will display results for any selected files. This is purposefully built into the system for flexibility but puts the onus on the analyst to understand if the results make sense. For example, comparing two scenarios with a different number of sensors is feasible when evaluating the incremental gain placing additional sensors in the environment. However, the data is misleading if the analyst is evaluating overall tracker performance with varying IMM parameters. In both cases, a View Plots push button displays the visual representation of the environment for verification (see fig. 3.14).

Figure 3.12: List of Data Sets for Comparative Analysis
3. ARGOS

3.2.5.1 2-Dataset Comparison

Figure 3.13 is an example of the 2-dataset analysis screen, which contains five distinct blocks of information. Where applicable, a \textit{delta} column either indicates parameter inequality or calculates the numerical difference in parameter values.

1. Environment Parameters Table
   This block lists high-level parameters for quick comparison, which includes the number of runs, number of sensors, number of targets, sensor fusion approximation and coefficient values, and execution time.

2. Airport Parameters Table
   Compares airport locations and perimeter settings.

3. Sensor Parameters Table
   Compares relevant radar parameters, such as, sensor location, sensor measurement uncertainty, probability of failure, tracker type, and tracker settings.

4. Target Fused Positional MSE Graph
   The MSE graph is the same graphical representation of the data from the simulation screen (fig. 3.6), displaying the MSE for the two selected datasets. The tabs are colored coded to indicate which target tracks have the best overall MSE. If the target ID is black, then the target from the first dataset has the overall best MSE. If the target ID is red, then vice versa.

5. Target Parameters Table
   The target parameter table displays the numerical overall MSE for each target, and the percentage improvement from the first dataset to the second dataset (which corresponds to the coloring of the tabs mentioned above). Similar to the cell selection functionality in the simulation UI, the user can select a target and the graph is updated to display the selected target.
3.2 Analyst

Figure 3.13: 2-Dataset Comparison

Figure 3.14: 2-Dataset Plot Comparison
3. ARGOS

3.2.5.2 N-Dataset Comparison

Comparing more than two datasets is beneficial and informative, but lacks the one-to-one comparison and delta values. The general display of numerical and graphical results are the same as the 2-Dataset view but contains the values for all user selected datasets. Additionally, the execution time is shown in a bar graph.

![Figure 3.15: N-Dataset Comparison](image)

3.2.6 Documentation

The underlying mathematical formulations and algorithms are accessible to interested users by selecting Documentation and selecting the topic from the menu toolbar. ARGOS opens a PDF document based on the user’s specified PDF viewer installed on their local computer.
3.3 Developer

ARGOS is not designed simply for analyzing different sensor management configurations but with the purpose of flexibility and sustainability to easily incorporate new state-of-the-art algorithms. A modified version of the Model-View-Controller (MVC) architecture is employed to achieve these goals.

3.3.1 Model-View-Controller

The Model-View-Controller is an architectural design pattern that breaks the application development into three different (nearly mutually exclusive) modular components (shown in fig. 3.16):

- **Model** - encapsulates the underlying data (system state) and defines the logic to manipulate the data (system state)
- **View** - output visualization of the data for the user (UI)
- **Controller** - intermediary communication line between models and views

![Figure 3.16: Standard Model-View-Controller](image)

The decomposition of the design allows each component to be updated without affecting each other. For instance, an analyst is interested in comparing the tracking performance of two heterogeneous sensors. A developer does not need to make changes to the model as the data is already available but simply update the view code to extract the data and display. Therefore, the risk of incidental model changes is removed for increased stability in the application. From another perspective, an analyst is interested in the correlation
accuracy of a measurement-to-track fusion approach as opposed to the embedded track-to-track fusion. A new model is added that performs the measure-to-track fusion with no edits to the view. Finally, with proper interfaces in place, a plug-and-play structure allows models of varying fidelity to easily be replaced. For example, the Sensor-Tracking Interface requires the sensor to pass along range and azimuth. A low fidelity model may measure true range and azimuth, then apply white Gaussian noise whereas a high fidelity model rely more on the range range equation, system losses, pre- and post-processing, etc. to produce the same parameters. The key is the proper interfaces between the various components for the developers.
Figure 3.18: ARGOS Model-View-Controller
3. ARGOS

3.3.1.1 Model

The model contains the underlying state of the system and the logic to manipulate the state. ARGOS contains numerous models to support the research objectives. A short list of examples are listed below. More details of these models are presented throughout the dissertation where appropriate.

- Target State Dynamics (Section 4.1)
- Target Tracking (Section 4.2)
- Sensor Models (Section 4.3)
- Sensor Placement Optimization (Section 5.4)
- Target State Fusion Center
  - 2 Track Fusion (Section 4.4.1)
  - N Track Fusion (Section 4.4.2)
  - Cross-Covariance Approximation (Section 4.4.3)
- Likelihood Function
  - 2 Track Likelihood (Section 4.5.1)
  - N Track Likelihood (Section 4.5.2)
- Assignment Algorithms
  - Sequential m-Best S-D Assignment Algorithm (Section 4.7)
  - Fuzzy Clustering Assisted Track-to-Track Correlation (Section 5.2)
- ATC Fuzzy Inference System (Section 5.1)
- Fuzzy Mode Detection (Section 5.3)
- Genetic Algorithm based Sensor Placement Optimization (Section 5.4)

3.3.1.2 View

View is the visual representation of the model presented to the user and the point of contact that the user interacts with the model. Figure 3.19 displays snippets of a few ARGOS views.
3.3 Developer

3.3.1.3 Controller

The controller provides the link between the view and the model. The controller determines the appropriate models to call when the user performs an action on the view. Similarly, the controller passes data from the model to the view to update the graphical display for the user. In the sequence diagrams below, it can be seen that the controller manages the flow of events during simulation and optimization algorithm, calling the appropriate models when necessary.

3.3.2 Sequence Diagrams

Sequence diagrams provide a graphical flow of object interaction in a time sequence. Two sequence diagrams (Simulation Execution and Sensor Placement Optimization via a Genetic Algorithm) are provided for illustrative purposes, shown in fig. 3.20 and fig. 3.21 respectively.
Figure 3.20: Simulate Sequence Diagram
Figure 3.21: Sensor Placement via GA Sequence Diagram
“Success is no accident. It is hard work, perseverance, learning, studying, sacrifice, and most of all, love of what you are doing”

Pele

“A person who never made a mistake never tried anything new”

Albert Einstein

4

Problem Formulation

4.1 Targets

4.1.1 State

The target (a.k.a. aircraft) state vector is represented in the 2-dimensional Euclidean space \((x \in \mathbb{R}^n)\). Thus, a target can be characterized by the following vector for any time step \(k\),

\[
x_k = \begin{bmatrix} x & \dot{x} & y & \dot{y} & \omega \end{bmatrix}
\]  (4.1)
4.1 Targets

where $x$ and $\dot{x}$ are the position and velocity components along the x-axis, $y$ and $\dot{y}$ are the position and velocity components along the y-axis, and $\omega$ is the turn rate.

4.1.2 Trajectory

The trajectory $\mathcal{T}_t$ for target $t$ is the collection of states that dictate the target’s position during the length of the simulation.

$$\mathcal{T}_t = [x^T_0 \ x^T_1 \ \ldots \ x^T_f]$$ (4.2)

Subsequent states are a function of the target’s previous state and intended turn rate (as described in the following section).

$$x_{k+1} = A \ast x_k$$ (4.3)

where

$$A = \begin{bmatrix}
1 & \sin(\omega \Delta t) / \omega & 0 & \cos(\omega \Delta t) - 1 / \omega & 0 \\
0 & \cos(\omega \Delta t) & 0 & -\sin(\omega \Delta t) & 0 \\
0 & -\cos(\omega \Delta t) / \omega & 1 & \sin(\omega \Delta t) / \omega & 0 \\
0 & \sin(\omega \Delta t) & 0 & \cos(\omega \Delta t) & 0 \\
0 & 0 & 0 & 0 & \omega
\end{bmatrix}$$ (4.4)

which dictates a nearly-coordinated turn model (CT). In the case that $\omega = 0$ (linear trajectory), then eq. (4.4) is simplified to

$$A = \begin{bmatrix}
1 & \Delta t & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & \Delta t & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}$$ (4.5)
4. PROBLEM FORMULATION

which dictates a constant velocity model (CV).

4.1.3 Trajectory Change Points

Trajectory Change Points (TCP), or sometimes referred as way-points, are geographical positions within the environment that indicate a change in aircraft heading. Each target has at minimal two way-points where $T_0$ is the target starting location and $T_f$ is the target ending location. Intermediate way-points are embedded between $T_0$ and $T_f$.

Whereas way-points are static, TCPs are relative to the target’s current state. For example, if the target is en route to way-point 3, then way-point 3 is considered the next TCP and way-point 4 is considered TCP+1. Starting at way-point 0, targets travel in linear motion ($\omega = 0$) towards the next TCP, then undergo a nearly-coordinated turn ($\omega \neq 0$) until their velocity vector is aligned with the next TCP+1, and then return to a linear motion ($\omega = 0$). This process is repeated until the target reaches its final way-point. Figure 4.1 shows an example of a target trajectory with six way-points.

There are three standard turn types, shown in fig. 4.2, utilized within Center-TRACON Automation, an ATC automation tool developed by NASA Ames Research Center. This work employs the inside turn type to maneuver the target from the current TCP to TCP+1.
4.1 Targets

Figure 4.1: Target Trajectory with Way-points

Figure 4.2: Standard Turn Types
4. PROBLEM FORMULATION

4.1.4 Conformance

Based on the assumption that all aircraft (manned and unmanned) are equipped with ADS-B, two distinct types of aircraft exist, those that conform to a specified flight plan (conforming) and those that do not (non-conforming). Aircraft that divert from predefined flight plans may result from harmless incidents such as poor weather conditions, emergency needs, and equipment malfunction to some form of malicious or spurious intent.

![Figure 4.3: Conforming Target](image)

4.1.4.1 Example

FlightAware (59) provides live flight tracking using the filed FAA flight route and ADS-B data. Figure 4.5a is a snippet extract from (59) that depicts Southwest Flight 2362 departing from KSAN enroute to KLAS on March 8, 2016, along with the ARGOS based depiction of the diverted route. The dotted blue line is the filed flight route and the solid green line is the route based on ADS-B data. The lower left section of the image indicates that the pilot diverted the planned route to then return later to the planned route. Figure 4.4a shows the scenario recreated in ARGOS.
4.1 Targets

(a) Weather Diverted
(b) Spurious Intent

Figure 4.4: Non-Conforming Targets

(a) Flight Aware
(b) ARGOS

Figure 4.5: Flight Comparison of Southwest FL2362
4. PROBLEM FORMULATION

4.2 Tracking Filter

The performance of the tracking filter is dependent on the underlying model used in anticipation of target motion. A single model Kalman Filter based on the assumption of a target traveling at a constant velocity (CV) is included in the implementation. The constant velocity model performs well during that prescribed motion but struggles during any target maneuver. The Interacting Multiple Models (IMM) tracker improves upon tracking capability of maneuvering targets by using a collection of Kalman Filters, in this case two Kalman Filters, one each for the CV and CT modes. Similar to the Kalman Filter above, the IMM predicts and updates its state estimate at each measurement, as described in (60). The key component of the IMM used later on in the Fuzzy Mode Detection is the mode posterior probability

4.3 Sensor Models

Target measurements, \( z_k \), are made based on two types of sensing devices, ground-based radars and satellite-based GPS. Although the sensing capabilities are vastly different, the sole difference in this work is the measurement units (polar versus Cartesian) as discussed below.

4.3.1 Radar Measurements

Radar systems use electromagnetic wave propagation to detect and measure objects within a particular space. The radar provides nonlinear measurements \( z_R = [r_m \theta_m]^T \) of bearing (azimuth) and range, computed as:

\[
\begin{bmatrix}
  r_m \\
  \theta_m
\end{bmatrix} = \begin{bmatrix}
  \sqrt{x^2 + y^2} \\
  \tan^{-1}(\frac{y}{x})
\end{bmatrix} + \eta_R
\]  

(4.6)
where \( \eta_R = [N(0, \sigma_r^2) \ N(0, \sigma_\theta^2)]^T \) is white Gaussian noise to represent sensor measurement uncertainty (i.e., variance in range \( \sigma_r^2 \) and variance in azimuth \( \sigma_\theta^2 \)), which may be due to random errors, systematic errors, or spurious errors (20).

Simply converting polar measurements to Cartesian coordinates introduces issues when tracking moving targets (61): (a) the conversion process is biased, and (b) the correlation between the co-variance estimate and measurement error is biased. To cope with the issues, the polar measurements are converted to Cartesian coordinates via the Modified Unbiased Converted Measurement (MUCM) approach prior to state estimation.

\[
\begin{bmatrix}
x \\
y
\end{bmatrix} = \begin{bmatrix}
e^{-\frac{\sigma_\theta^2}{2}} r_m \cos \theta_m \\
e^{-\frac{\sigma_\theta^2}{2}} r_m \sin \theta_m
\end{bmatrix}
\] (4.7)

where \( \sigma_\theta \) is the sensor variance in azimuth. Similarly, the sensor noise co-variance matrix \( R \) must be augmented to deal with the issues listed above:

\[
R = \begin{bmatrix}
R_{11} & R_{12} \\
R_{21} & R_{22}
\end{bmatrix}
\]

such that

\[
R_{11} = \frac{1}{2} \left( r_m^2 + \sigma_r^2 \right) \left[ 1 + \cos(2\theta_m) e^{-2\sigma_\theta^2} \right] - e^{-\sigma_\theta^2} r_m^2 \cos^2 \theta_m
\] (4.8)

\[
R_{12} = R_{21} = \frac{1}{2} \left( r_m^2 + \sigma_r^2 \right) \left[ \sin(2\theta_m) e^{-2\sigma_\theta^2} \right] - e^{-\sigma_\theta^2} r_m^2 \cos \theta_m \sin \theta_m
\] (4.9)

\[
R_{22} = \frac{1}{2} \left( r_m^2 + \sigma_r^2 \right) \left[ 1 - \cos(2\theta_m) e^{-2\sigma_\theta^2} \right] - e^{-\sigma_\theta^2} r_m^2 \sin^2 \theta_m
\] (4.10)

where \( \sigma_r \) is the sensor variance in range.

Another limiting factor is that measurement uncertainty is universal irrespective of sensor location which presents challenges when optimizing sensor placement. Measurement...
uncertainty is based on the size of the resolution cell where the target is located; the farther out the target, the larger the resolution cell and subsequently, the uncertainty in its position. In addition, radars have minimum and maximum ranges at which a target can be detected. To account for these factors, the sensor measurement is written as a function of target range, as shown in fig. 4.6.

4.3.2 GPS Measurements

As opposed to the nonlinear measurements provided by radar, GPS provides positional latitude/longitude measurements, which are represented as \( \mathbf{z}_{\text{GPS}} = [x_m \ y_m]^T \), computed as:

\[
\begin{bmatrix}
  x_m \\
  y_m
\end{bmatrix} =
\begin{bmatrix}
  x \\
  y
\end{bmatrix} + \eta_{\text{GPS}}
\]  

(4.11)

where \( \eta_{\text{GPS}} = [N(0, \sigma_x^2) \ N(0, \sigma_y^2)]^T \) is white Gaussian noise to represent sensor measurement uncertainty. Additionally, measurement uncertainty is consistent and not a function of range as in Radar-based measurements.
4.3.3 Sensor Failure

Sensor failure, either due to malicious intent or system degradation, is an inherent property that can disrupt the operator’s perception of the environment. Injecting multiple sensors into the environment adds redundancy and robustness in the event of a sensor becoming non-operational. Users specify the probability of failure $p(F_s)$ for each radar (GPS sensors are excluded from failure at this point). Sensor operation at each time step $k$ is based on the roll of the die. Once the a sensor fails, it remains non-operational for the remainder of the simulation run.

4.4 Track Fusion

It is assumed that state data is synchronized when transmitted to the fusion center. Asynchronous track-to-track fusion has been studied previously with applicable approaches found in (62, 63, 64).

4.4.1 2 Track Fusion

Given 2 state estimates $\hat{x}_1, \hat{x}_2$ (and accompanying covariance matrices $P_1, P_2$) from 2 sensors $S_1, S_2$, and assuming all 2 tracks represent the same truth object, then a fused estimate $x_{f,N}$ and fused covariance $P_{f,N}$ of the true target state can be formulated (35, 65, 66) as

$$\hat{x}_{f,2} = \hat{x}_1 + (P_1 - P_{12}) \mathbf{T}^{-1} (\hat{x}_2 - \hat{x}_1)$$  \hspace{1cm} (4.12)

$$P_{f,2} = P_1 + (P_1 - P_{12}) \mathbf{T}^{-1} (P_2 - P_{21})$$  \hspace{1cm} (4.13)
where

\[
T = \begin{cases} 
P_1 + P_2 - P_{12} - P_{21} & \text{assuming correlated estimation errors where} \\
 & P_{ij} \text{ is the cross-covariance and } P_{ji} = P_{ij}^T \\
P_1 + P_2 & \text{assuming uncorrelated estimation errors}
\end{cases}
\] (4.14)

### 4.4.2 N Track Fusion

The 2 track fusion approach above is generalized to \(N\) tracks. Given \(N\) state estimates \(\hat{x}_1, \hat{x}_2, ..., \hat{x}_N\) (and accompanying covariance matrices \(P_1, P_2, ..., P_N\)) from \(N\) sensors \(S_1, S_2, ..., S_N\), and assuming all \(N\) tracks represent the same truth object, then a fused estimate \(\hat{x}_{f,N}\) and fused covariance \(P_{f,N}\) of the true target state can be formulated based on a Maximum Likelihood (30, 65) criterion:

\[
\hat{x}_{f,N} = P_{f,N} I_N^T N^{-1} x_N
\] (4.15)

where

\[
P_{f,N} = (I_N^T N^{-1} I_N)^{-1}
\] (4.16)

such that \(x_N\) is the vertical concatenation of \(N\) state estimates

\[
x_N = \left[\begin{array}{cccc}
x_1^T & x_2^T & \cdots & x_N^T
\end{array}\right]^T
\] (4.17)

\(P_N\) is the corresponding covariance matrix to \(x_N\)

\[
P_N = \begin{bmatrix}
P_{11} & P_{12} & \cdots & P_{1N} \\
P_{21} & P_{22} & \cdots & P_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
P_{N1} & P_{N2} & \cdots & P_{NN}
\end{bmatrix}
\] (4.18)
4.4 Track Fusion

and $I_N$ is the horizontal concatenation of $M \times M$ identity matrices ($M$ is the dimensionality of the state space)

$$I_N = \begin{bmatrix} I_{M \times M} & I_{M \times M} & \ldots & I_{M \times M} \end{bmatrix}^T$$ (4.19)

4.4.3 Cross-Covariance Approximation

Early works in track-to-track fusion assumed that if two (or more) targets represented the same truth object, the independent track files were uncorrelated. However, Bar-Shalom(35) mathematically proved that ”while the measurement noises of the ... sensors can be safely assumed as independent there is still the same process noise in the dynamic model”. The correlated noise is accounted for by the off-diagonal terms in $P_{ij}$ where $i \neq j$ in eq. (4.18). Kaplan(30) states that while calculating the exact cross-covariance is feasible when using a linear state estimator such as a Kalman Filter, actual cross-covariance calculations are infeasible for IMM trackers for maneuvering targets due to the varying model weights for each sensor. Oftentimes though, associated tracks are assumed uncorrelated(30) and the cross-covariance terms are set to zero (i.e., $P_{i,j} = 0$ where $i \neq j$). In this case, then eq. (4.15) and eq. (4.16) can be simplified to

$$\hat{x}_{f,N} = P_{f,N} \sum_{i=1}^{N} P_{ii}^{-1} \hat{x}_i$$ (4.20)

and

$$P_{f,N} = \left( \sum_{i=1}^{N} P_{ii}^{-1} \right)^{-1}$$ (4.21)

Techniques for approximating the cross-covariance(30, 68, 69) have also been studied since processing and communication requirements restrict usage of the exact calculation. (30, 40) describe one approach to approximating the cross-covariance term between two target estimates:

$$\hat{P}_{ij} = \rho * A_i A_j^T$$ (4.22)
4. PROBLEM FORMULATION

where $A_t$ is the square root of the covariance matrix for estimate $\hat{x}_t$,

$$A_tA_t^T = P_t \quad (4.23)$$

and $\rho$ is the cross-covariance coefficient term that can be empirically calculated through Monte Carlo simulations. This approach is referred as the one-parameter approximation approach given the single parameter $\rho$. The approximated cross-covariance term is guaranteed to be positive semi-definite for $\rho \in [0, 1)$ (30).

The one-parameter approach is extended to the two-parameter approach (30) by decoupling the position and velocity components. The approximated cross-covariance is computed as:

$$
\hat{P}_{ij} = \begin{bmatrix}
\rho_p A_{i,p} A_{j,p}^T & \sqrt{P_{p}P_{v}} \ast A_{i,p} A_{j,v}^T \\
\sqrt{P_{p}P_{v}} \ast A_{i,v} A_{j,p}^T & \rho_v A_{i,v} A_{j,v}^T
\end{bmatrix} \quad (4.24)
$$

where $A_{t,p}$ is the square root of the covariance matrix for estimate $\hat{x}_t$ with respect to position, and $A_{t,v}$ is the square root of the covariance matrix for estimate $\hat{x}_t$ with respect to velocity. The two-parameter approximation technique assumes a state space of four dimensions, $[x \ x \ y \ y]^T$, but this work also incorporates turn rate into the state space, therefore, the two-parameter approach is extended to account for the turn rate $\omega$ when an IMM tracker is employed. The three-parameter approximation is calculated as:

$$
\hat{P}_{ij} = \begin{bmatrix}
\hat{P}_{ij,pv} & \hat{P}_{ji,\omega} \\
\hat{P}_{ij,\omega} & \ast
\end{bmatrix} \quad (4.25)
$$

where $\hat{P}_{ij,pv}$ is eq. (4.24), and $\hat{P}_{ij,\omega}$ is the weighted piecewise mean of the $\omega$ terms, that is,
4.5 Likelihood Function

\[
\hat{P}_{ij,\omega} = \rho_{\omega} \hat{P}_{i,\omega} \hat{P}_{j,\omega} = \rho_{\omega} \\
\begin{bmatrix}
\hat{P}_{i,x}\omega \\
\hat{P}_{i,y}\omega \\
\hat{P}_{i,\omega}\omega \\
\end{bmatrix}
\begin{bmatrix}
\hat{P}_{j,x}\omega \\
\hat{P}_{j,y}\omega \\
\hat{P}_{j,\omega}\omega \\
\end{bmatrix}
\]

(4.26)

and \( \hat{P}_{ji,pu} = \hat{P}^T_{ij,pu} \).

4.5 Likelihood Function

Let \( N_s \) represent the number of sensors and \( N_t \) represent the number of targets in the simulation. Each sensor in the decentralized architecture makes independent measurements and processes those measurements by means of a filter (e.g., Kalman Filter or IMM) to generate state estimates \( \hat{x}_i \) and error covariances \( P_i \) for \( i = 1, \ldots, N_t \). Let \( T_j^i \) represent the track file that contains the state estimate and covariance for the \( i \)th target by the \( j \)th sensor. A tuple

\[
\mathcal{T} = \left\{ T_j^i \mid i = 1, \ldots, N_s \right\}
\]

(4.27)

is the grouping of track files \( T_j^i \) such that it contains no more than one track file per sensor.

The fundamental challenge in track-to-track correlation is determining whether separate reported tracks from disparate sensors represent the same truth object. A common approach is to evaluate the likelihood that the track estimates are statistically the same. For a simple two track file scenario, a hypothesis test is formed between targets \( i \) and \( j \):

\[
H_0 : \ x_i = x_j \\
H_1 : \ x_i \neq x_j
\]

(4.28)

If the null hypothesis \( H_0 \) is accepted, then the two tracks represent the same truth object and are fused. In the multi-target, multi-sensor case, a global association hypothesis
4. PROBLEM FORMULATION

The set \( \mathcal{H} \) is created containing tuples \( \mathcal{T} \). The objective is to find the global hypothesis most likely among the set of all global hypotheses. The mathematical derivation has been presented by various authors found in \( (30, 36, 42) \) and briefly summarized here. The likelihood of a tuple (assuming \( P_d = 1 \)) is defined as:

\[
\Lambda(\mathcal{T}) = p(\vec{T}_1, \vec{T}_2, \ldots, \vec{T}_N|\mathcal{T})
\]  
(4.29)

Thus, the most likely global hypothesis is found by the Maximum Likelihood method:

\[
\hat{\mathcal{H}} = \arg \max_{\mathcal{T} \in \mathcal{H}} \prod_{T \in \mathcal{T}} \Lambda(\mathcal{T})
\]  
(4.30)

4.5.1 2 Target Likelihood

Kaplan (30) states though that since the true target state is unknown, an exact expression for the likelihood is unavailable. This is not a problem in a two sensor scenario as the true target state is replaced by the difference between the two tracks: \( \Delta_{ij} = \hat{x}_i - \hat{x}_j \). If the two tracks belong to the same truth object, then \( \Delta_{ij} \) adheres to a zero-mean normal distribution with variance \( P \). The likelihood of two targets is formulated as:

\[
\Lambda(\mathcal{T}) = p(\hat{x}_1, \hat{x}_2|x) = \frac{1}{|2\pi(P_{ii} + P_{jj} - P_{ij} - P_{ij}^T)|^{1/2}} \exp \left( -\frac{1}{2} (\hat{x}_i - \hat{x}_j)^T (P_{ii} + P_{jj} - P_{ij} - P_{ij}^T)^{-1} (\hat{x}_i - \hat{x}_j) \right) 
\]  
(4.31)

If associated tracks are assumed uncorrelated, then eq. (4.31) is simplified to:

\[
\Lambda(\mathcal{T}) = p(\hat{x}_1, \hat{x}_2|x) = \frac{1}{|2\pi(P_{ii} + P_{jj})|^{1/2}} \exp \left( -\frac{1}{2} (\hat{x}_i - \hat{x}_j)^T (P_{ii} + P_{jj})^{-1} (\hat{x}_i - \hat{x}_j) \right) 
\]  
(4.32)
4.5.2 N Target Likelihood

When $N_s > 2$, the unknown target state cannot be removed as in the two sensor case, therefore, a generalized likelihood formulation is employed:

$$\Lambda(T) = p(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N|x) = \prod_{i=1}^{N} \frac{1}{|2\pi P_N|^{1/2}} \exp \left( -\frac{1}{2} (\bar{x}_N - I_N \hat{x}_{f,N})^T P_N^{-1} (\bar{x}_N - I_N \hat{x}_{f,N}) \right)$$

(4.33)

where $P_N$, $\bar{x}_N$, and $x_{f,N}$ were defined earlier. Again, if we assume no correlation between targets, then eq. (4.33) can be written as

$$\Lambda(T) = p(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N|x) = \prod_{i=1}^{N} \frac{1}{|2\pi P_{ii}|^{1/2}} \exp \left( -\frac{1}{2} (x_i - \hat{x}_{f,N})^T P_{ii}^{-1} (x_i - \hat{x}_{f,N}) \right)$$

(4.34)

4.6 Assignment Problem

4.6.1 2-D Assignment Problem

We begin by considering the simplified two sensor case for the track-to-track association problem where sensor 1 has $N_1$ track files and sensor 2 has $N_2$ track files. Assuming “that the track association events among different track pairs are independent, then the 2-D assignment formulation finds the most likely (joint) track-to-track association hypothesis by solving the following constrained optimization(36)“

$$\min_{\chi_{ij}} \sum_{i=0}^{N_1} \sum_{j=0}^{N_2} c_{ij} \chi_{ij}$$

(4.35)
4. PROBLEM FORMULATION

subject to

\[ \sum_{i=0}^{N_1} \chi_{ij} = 1, \quad j = 1, \ldots, N_2 \]  
\[ \sum_{j=0}^{N_2} \chi_{ij} = 1, \quad i = 1, \ldots, N_1 \]  

(4.36)

\[ \chi_{ij} \in \{0, 1\}, \quad i = 0, 1, \ldots, N_1, \quad j = 0, 1, \ldots, N_2 \]  

(4.37)

where \( \chi_{ij} \) is defined by the binary assignment

\[ \chi_{ij} = \begin{cases} 1 & \text{track } i \text{ from sensor 1 and track } j \text{ from sensor 2 represent the same target} \\ 0 & \text{otherwise} \end{cases} \]  

(4.38)

and \( c_{ij} \) is the negative log-likelihood of the two tracks defined in eq. (4.31) and eq. (4.32)

\[ c_{ij} = -\ln \Lambda (T) \]  

(4.39)

4.6.2 N-D Assignment Problem

Efficient algorithms, such as JVC, Munkres (Hungarian), Auction, etc., have been developed and shown to solve the 2-D Assignment problem. The two sensor case is extended to a multidimensional assignment problem for \( N_s > 2 \) scenarios, namely the S-D assignment problem, where the objective is to minimize the following constrained optimization

\[ \min_{\chi_{i_1,i_2,\ldots,i_s}} \sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} \cdots \sum_{i_s=0}^{N_s} c_{i_1,i_2,\ldots,i_s} \chi_{i_1,i_2,\ldots,i_s} \]  

(4.40)
subject to
\[
\begin{align*}
\sum_{i_2=0}^{N_1} \sum_{i_s=0}^{N_s} \chi_{j_{i_2}...i_s} &= 1, \quad j = 1, ..., N_1 \\
\sum_{i_1=0}^{N_1} \sum_{i_3=0}^{N_3} \sum_{i_s=0}^{N_s} \chi_{i_1j_{i_3}...i_s} &= 1, \quad j = 1, ..., N_2 \\
&\vdots \\
\sum_{i_1=0}^{N_1} \sum_{i_{s-1}=0}^{N_{s-1}} \chi_{i_1i_{i_2}...i_{s-1}} &= 1, \quad j = 1, ..., N_s
\end{align*}
\]
(4.41)

\[\chi_{i_1i_2...i_s} \in \{0, 1\}, \quad i_1 = 0, 1, ..., N_1, \quad i_2 = 0, 1, ..., N_2, \quad \ldots \quad i_s = 0, 1, ..., N_s\] (4.42)

where \(\chi_{i_1i_2...i_s}\) is defined by the binary assignment

\[\chi_{i_1i_2...i_s} = \begin{cases} 
1 & \text{tracks } i_1i_2...i_s \text{ represent the same target} \\
0 & \text{otherwise}
\end{cases}\] (4.43)

Similar to the 2-D problem formulation, \(c_{i_1i_2...i_s}\) is the negative log-likelihood of the \(S\) tracks defined in eq. (4.33) and eq. (4.34)

\[c_{i_1i_2...i_s} = -\ln \Lambda (\mathcal{F})\] (4.44)

This constrained optimization problem is NP-hard (36), but algorithms exists to find suboptimal solutions, such as Lagrangian Relaxation and Sequential m-Best S-D Assignment Algorithm.

### 4.7 Sequential m-Best 2-D Assignment Algorithm

The sequential m-Best 2-D assignment algorithm is a heuristic approach to solving the S-D assignment problem. The approach solves a series of generalized 2-D assignment problems,
4. PROBLEM FORMULATION

wherein each iteration of the algorithm introduces the tracks from the next sensor and associates those tracks to the previous results. To avoid converging to a local minimum, the m-best solutions (based on Murty’s algorithm(70)) from each iteration is kept and used in the next iteration. Popp(42) states that “[d]etermining m-best solutions (as opposed to only the best one) becomes especially important for assignment-based approaches to data association since the hard irrevocable decisions that such approaches make can be mitigated via the m-best assignment formalism.” Figure 4.7 depicts the sequential flow of 2-D assignment operations to arrive at the best assignment to the S-D problem. The figure sets $m = 3$ as to illustrate the number of solutions maintained and fed into the subsequent iteration.

![Figure 4.7: Sequential mBest 2-D Assignment Algorithm Flow](image)

The initial iteration associates tracks from the first two sensors based on the 2-D formulation described above. The general 2-D assignment problem generates a solution

$$
\mathcal{J}_2^p = \{ \mathcal{J}_{2,1}, \mathcal{J}_{2,2}, ..., \mathcal{J}_{2,k} \}; \quad p = 1, ..., m
$$

(4.45)
Table 4.1: 2-D Assignment Problem (1st iteration)

<table>
<thead>
<tr>
<th></th>
<th>$T_2^1$</th>
<th>$T_2^2$</th>
<th>$T_2^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1^1$</td>
<td>21</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>$T_2^2$</td>
<td>5</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>$T_3^3$</td>
<td>10</td>
<td>8</td>
<td>11</td>
</tr>
</tbody>
</table>

where $T_{2,i}$ is the $i^{th}$ tuple and $k$ is the number of tuple solutions. Table 4.1 provides an example of the initial 2-D assignment for sensors 1 and 2, each containing three track files labeled 1, 2, and 3. The numerical value in the table is the negative log-likelihood calculation using appropriate eq. (4.31) or eq. (4.32).

The best solution to the 2-D assignment problem is

$$ T_2^1 = \{T_2^1, T_2^2, T_2^3\} $$  \hspace{1cm} (4.46)

where

$$ T_{2,1} = \{T_1^1, T_3^1\}, \ T_{2,2} = \{T_2^1, T_1^1\}, \ T_{2,3} = \{T_3^1, T_2^1\} $$  \hspace{1cm} (4.47)

The additional $m - 1$ best ranked solutions are also found using Murty’s algorithm for a total of $m$ solutions.

The second iteration incorporates the next sensor in the list and executes the 2-D assignment problem using the track files from the sensor against the results from the previous iteration to generate the solution

$$ T_3^p = \{T_3, T_3, ..., T_{3,k}\}; \ p = 1, ..., m $$  \hspace{1cm} (4.48)

Table 4.2 provides an example of the second 2-D assignment using $T_2^1$ from the initial iteration and including sensor 3 track files. Likewise, the numerical value in the table is the negative log-likelihood calculation now using appropriate eq. (4.33) or eq. (4.34).
4. PROBLEM FORMULATION

| Table 4.2: 2-D Assignment Problem (2nd iteration) |
|-----------------|-----------------|-----------------|
| \{T_1^1, T_3^1\} | 7    | 16  | 10 |
| \{T_1^2, T_3^2\} | 18  | 25  | 8  |
| \{T_1^3, T_3^3\} | 19  | 4   | 17 |

The best solution to the 2-D assignment problem is

\[ T_3^1 = \{T_3,1, T_3,2, T_3,3\} \] (4.49)

where

\[ T_3,1 = \{T_2,1, T_3^1\} = \{T_1^1, T_3^2, T_1^3\}, \]
\[ T_3,2 = \{T_2,2, T_3^2\} = \{T_1^2, T_3^3, T_1^3\}, \]
\[ T_3,3 = \{T_2,3, T_3^3\} = \{T_1^3, T_2^3, T_2^2\} \] (4.50)

The same process is repeated for \( T_3^2, \ldots, T_3^n \). This produces \( m \) results for each the previous \( m \) solutions, creating a total of \( m^2 \) solutions. The list is then trimmed down to \( m \) solutions for the next iteration. The process is repeated until all \( N_s \) sensors are subsequently added and afterwards, the best overall solution is kept.

4.8 Fuzzy Clustering

In fuzzy clustering, the degree of membership of any data point \( x \) on the set \( A \) is defined as:

\[
\mu_A(x) = \begin{cases} 
1 & \text{if } x \text{ is a full member of } A \\
\in (0, 1) & \text{if } x \text{ is a partial member of } A \\
0 & \text{if } x \text{ is not a member of } A 
\end{cases}
\] (4.51)
4.8 Fuzzy Clustering

The data (or observations) is collected into a $n \times N$ matrix

$$Z = \begin{pmatrix}
  z_{11} & z_{12} & \cdots & z_{1N} \\
  z_{21} & z_{22} & \cdots & z_{2N} \\
  \vdots & \vdots & \ddots & \vdots \\
  z_{n1} & z_{n2} & \cdots & z_{nN}
\end{pmatrix}$$  \hspace{1cm} (4.52)

where $N$ is the total number of observations and $n$ is the number of variables in the observation:

$$z_{ik} = \begin{bmatrix} z_{1k} & z_{2k} & \cdots & z_{nk} \end{bmatrix}^T$$  \hspace{1cm} (4.53)

A partition matrix $U = [\mu_{ik}]_{c \times N}$ then defines the degree of membership $\mu_{ik}$ for each observation to each cluster such that

$$\mu_{ik} = \frac{1}{c} \sum_{j=1}^{c} \left( \frac{d_{ik}A}{d_{jk}A} \right)^{2/m-1}$$  \hspace{1cm} (4.54)

where $c$ is the number of clusters, $d$ is the similarity matrix, $i$ is the $i^{th}$ data point, $m$ is a weighting constant, and $A$ is the norm-inducing matrix.
5.1 Air Traffic Fuzzy Inference System

The expected growth in air traffic density due to NextGen and UAV integration will only further stress air traffic controllers. The ATC Fuzzy Inference System assists the controllers by highlighting aircraft that require additional monitoring.

5.1.1 Cascaded Fuzzy Inference System

The Air Traffic Control fuzzy inference system uses three inputs to compute the ATC Warning Level value for each target. Assuming a designer evaluates all three inputs
simultaneously, $3^3 = 27$ rules are required to evaluate all combinations. However, decomposing the inputs into logical groupings can reduce the number of rules. In this thesis, the intent conformance is computed by only evaluating inputs based on TCP and TCP+1 way-points. Then the intent conformance output is fed directly into the second fuzzy inference system along with the two track chi-square $\chi^2$ value to determine ATC warning levels. Figure 5.1 shows the cascaded ATC fuzzy inference system, broken down to illustrate the two fuzzy inference systems. This decomposition scales well to large problems spaces as demonstrated in (71).
5. ALGORITHM DEVELOPMENT

Figure 5.2: TCP and TCP+1 Angles

Membership functions are used for more granularity in controlling the output surface.

Figure 5.3: Intent Conformance FIS Input 1

Figure 5.6 shows the input-output surface relationship. The asymmetrical surface between the two intent input values shows preference towards targets adhering to the next way-point with a degraded value for aircraft headed to TCP+1.
5.1 Air Traffic Fuzzy Inference System

![Figure 5.4: Intent Conformance FIS Input 2](image)

![Figure 5.5: Intent Conformance FIS Output](image)

<table>
<thead>
<tr>
<th>TCP near</th>
<th>TCP middle</th>
<th>TCP far</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>high</td>
<td>med-high</td>
</tr>
<tr>
<td>med-high</td>
<td>med</td>
<td>med-low</td>
</tr>
<tr>
<td>med-low</td>
<td>low</td>
<td>low</td>
</tr>
</tbody>
</table>

Table 5.1: Intent Conformance Rule Base
5. ALGORITHM DEVELOPMENT

5.1.3 ATC Warning Level FIS

The second fuzzy inference system uses the intent conformance variable along with the chi-square $\chi^2$ distribution value between the fused sensor estimate and GPS tracking estimate. The $\chi^2$ value between the fused state estimate $\hat{x}_{f,i}$ and GPS-based state estimate $\hat{x}_{GPS,i}$ is calculated as:

$$
\chi^2(T_{f,i}, T_{GPS,i}) = [\hat{x}_{f,i} - \hat{x}_{GPS,i}]^T \ast (P_{f,i} + P_{GPS,i})^{-1} \ast [\hat{x}_{f,i} - \hat{x}_{GPS,i}] \quad (5.1)
$$

<table>
<thead>
<tr>
<th>Table 5.2: ATC Warning Level Rule Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intent high</td>
</tr>
<tr>
<td>Intent medium</td>
</tr>
<tr>
<td>Intent low</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
5.1 Air Traffic Fuzzy Inference System

Figure 5.7: ATC Warning Level FIS Input 1

Figure 5.8: ATC Warning Level FIS Input 2

Figure 5.9: ATC Warning Level FIS Output
5. ALGORITHM DEVELOPMENT

5.2 Fuzzy Clustering Assisted Track-to-Track Correlation

The fuzzy clustering based track-to-track correlation reduces the size of the solution space by removing illogical solutions. The algorithm first decomposes the problem space by the k-means clustering algorithm to determine which track files belong in which clusters. Second, the fuzzy clustering algorithm is executed to compute the partition matrix. Third, the clusters are adjusted using the partition matrix. Fourth, an S-D assignment algorithm is used to solve the reduced problem for each cluster (this work uses the Sequential m-Best S-D Assignment Algorithm as described earlier). Finally, the track-to-track correlations are appended together to arrive at the final result. The fuzzy c-means algorithm uses the track file state estimate for clustering analysis. Consider the example shown in fig. 5.11 below.

This is a rather simple example where clustering minimizes the problem space where each cluster contains only the track file state estimates of the same truth target. However, knowledge of the true number of targets is not always available. For example, consider the
5.2 Fuzzy Clustering Assisted Track-to-Track Correlation

![Figure 5.11: State Estimate Distribution](image)

Table 5.3: Example Partition Matrix

<table>
<thead>
<tr>
<th></th>
<th>$T_1^1$</th>
<th>$T_1^2$</th>
<th>$T_1^3$</th>
<th>$T_2^1$</th>
<th>$T_2^2$</th>
<th>$T_2^3$</th>
<th>$T_3^1$</th>
<th>$T_3^2$</th>
<th>$T_3^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>0.91</td>
<td>0.95</td>
<td>0.93</td>
<td>0.42</td>
<td>0.45</td>
<td>0.52</td>
<td>0.04</td>
<td>0.05</td>
<td>0.02</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>0.09</td>
<td>0.05</td>
<td>0.07</td>
<td>0.58</td>
<td>0.55</td>
<td>0.48</td>
<td>0.96</td>
<td>0.95</td>
<td>0.98</td>
</tr>
</tbody>
</table>

same data with only two clusters, shown in fig. 5.12. This decomposition separates the track files corresponding to target two ($T_2^*$). As a result, target two ($T_2^*$) is separated into two separate tracks by the fusion center where one track is based solely on $\{T_2^3\}$ and the second track is based on the fused results of the tuple $\{T_1^2, T_2^2\}$. To compensate, a greedy heuristic approach searches the partition matrix $U$ for the state estimate that has the largest membership to the cluster that is not already included in that cluster. It is known that each sensor provides a state estimate to the observation matrix $Z$, so the number of observations in a cluster will have a modulus value of zero when dividing the cluster size by the number of sensors

$$\text{mod}(\frac{\text{cluster size}}{N_S}) = 0$$  \hfill (5.2)

Table 5.3 shows an example of the partition matrix with membership values. Cluster 1 evaluates the partition matrix and adds $T_2^2$ to its modified cluster named Cluster 1*. Since
5. ALGORITHM DEVELOPMENT

the modulus is not zero \((mod(\frac{5}{3} \neq 0))\), the process is repeated and \(T^1_2\) is subsequently added to Cluster 1*. Likewise, the same procedure is repeated for Cluster 2 until equation (5.2) is satisfied. The final cluster configuration is shown in fig. 5.14.

![Initial Clustering](image1)

(a) Cluster 1 expanding data points based on partition matrix \(U\)

![Augmented Clusters](image2)

(b) Cluster 2 expanding data points based on partition matrix \(U\)

Figure 5.13: Initial Clustering

Figure 5.14: Augmented Clusters

Similar to the problem described earlier regarding target two \((T^*_2)\), each cluster contains a copy of the target, therefore two tracks are created using the same tuple \(\{T^1_2, T^2_2, T^3_2\}\). This issue is accounted for in post-processing at the fusion center where duplicate tuples are discarded. Figure 5.15 provides an overview of the processing flow for this example.

Populating the tuple with the correct track files is not sufficient to say that they belong to the same truth object. Assuming the track files represent the same truth object, their test statistic adheres to a chi-square \(\chi^2\) distribution\((35, 36, 40, 44, 65, 72)\). If the test statistic falls below a threshold, then the tracks are fused. In this work, the clustering
5.3 Fuzzy Mode Change Detection

5.3.1 Maximum Posterior Probability

Jilkov (55) outlined the early designs to mode detection. The simplest design is to determine aircraft flight mode by evaluating the posterior probability of the IMM

$$\mu_{k|k}^j = P \left( m_k^j | z^k \right)$$

(5.3)

where $z^k$ is the measurement at time step $k$ and $m_k^j$ is mode $j$ at time step $k$. Therefore, the projected mode of the aircraft is the mode with the largest posterior probability,

$$\hat{m}(k) = \arg \max_j \mu_{k|k}^j$$

(5.4)

However, this is prone to random fluctuations, measurement noise, process noise, and false alarms. It is observed that during steady modes of the system, one mode dominates other modes until a jump in the system occurs and another mode takes on the dominant

Figure 5.15: Clustering and Assignment Flow Example
position. A natural extension to the simple design described above is check if the current dominant mode has fallen below a specified dominance threshold $t_d$.

$$\max_{m^j \in M} \mu^j_{k|k} < t_d$$  \hspace{1cm} (5.5)$$

Once this condition is met, a mode change is acknowledged and the new dominant mode is computed by finding the mode that exceeds the dominance threshold.

$$\max_{m^j \in M} \mu^j_{k|k} \geq t_d$$  \hspace{1cm} (5.6)$$

Jilkov states two flaws of these heuristic approaches: (a) it does not consider the statistical properties of the posterior probability time series, and (b) it provides no estimation of when the change occurred.

### 5.3.2 CUSUM

Jilkov presented the cumulative sum (CUSUM)-type statistical test to address the two flaws with improved mode detection along with mode change occurrence through post-processing. Lowe (56) leveraged the design by Jilkov and described the approach for purposes of predicting pilot intent and aircraft trajectory in uncontrolled airspace.

$$H_0 : \quad m = m_{j_s}, \{\forall k_s, k_s + 1, \ldots, k\}$$

$$H_1 : \quad m \neq m_{j_s}, \{\forall k_s, k_s + 1, \ldots, k\}$$  \hspace{1cm} (5.7)$$

where the null hypothesis assumes the current mode is equal to the previous steady state mode $m_{j_s}$. $k_c \in (k_s,k]$ is the unknown mode time change. The CUSUM statistics are recursively computed as

$$S(k_s) = 0$$

$$S(k) = S(k - 1) + \mu_{j_s}(k) - \lambda_{j_s}$$  \hspace{1cm} (5.8)$$
5.3 Fuzzy Mode Change Detection

from time $k_s$ for $k \geq k_s$. $\mu_j(k)$ is the steady state mode at time $k$ and $\lambda_{js}$ is the sensitivity parameter to mode changes. Increasing the sensitivity too much may cause false positives whereas decreasing the sensitivity may cause missed mode detections and incorrect time projections. The null hypothesis is rejected when

$$S(k) - \max_{k_s \leq \gamma < k} S(\gamma) < -\tau_{js}$$

(5.9)

where $\tau_{js}$ is the null hypothesis rejection threshold. The threshold controls the minimum number of data points required to recognize a mode change. The time of mode change occurs at

$$k_c = \arg \max_{k_s \leq \gamma < k} S(\gamma)$$

(5.10)

(55) proposed a sensitivity parameter of $\lambda_{js} = 0.7$ and a threshold level of $\tau_{js} = 3 * \lambda_{js}$.

5.3.3 Fuzzy CUSUM

Aircraft equipped with ABS-B broadcast flight reports containing metadata about aircraft dynamics and intentions (73), such as position coordinates, North/East velocity, barometric altitude, airspeed, and trajectory change point. The trajectory change point provides a geographic location that the aircraft will execute a maneuver to change its course, thus undergoing some form of mode change during that maneuver. Course heading changes at the TCP are not instantaneous though. Although the type of turn (see fig. 4.2) may be deterministic (or pre-planned), execution of the coordinated turn is subject to human intervention. That is, a pilot may begin an inside turn a little early or late but statistically it should adhere to a normal distribution around the TCP, as shown in fig. 5.16.

A fuzzy inference system is integrated into the work to dynamically adjust the sensitivity parameter using ADS-B data reports to more quickly detect mode changes and more accurately predict the time that the mode change occurred.

The input to the controller is the distance to the broadcasted trajectory change point
and the output is the sensitivity parameter value. The output range of sensitivity parameters were based on the work from (56). The theory in the fuzzy logic controller is that when the aircraft is in a steady state mode, the sensitivity parameter is decreased so random fluctuations and noise in the system does not unintentionally cause a false positive. As the aircraft is near a trajectory change point, the sensitivity is increased to rapidly detect the mode change. Examples of rules in the fuzzy inference system include:

If distance is veryClose, then sensitivity is veryHigh

If distance is veryFar, then sensitivity is veryLow

The purpose is to not only capture the mode change quickly, but to allow the algorithm to accurately predict backwards the time of change by finding the maximum value of $S(k)$. The goal of the fuzzy-based CUSUM is to maximize the value of $S(k)$ at the anticipated trajectory change point and then decrease its value such that the null hypothesis rejection threshold is met quickly.
5.3 Fuzzy Mode Change Detection

Figure 5.17: Fuzzy CUSUM Input Membership Functions

Figure 5.18: Fuzzy CUSUM Output Membership Functions
5. ALGORITHM DEVELOPMENT

5.4 Sensor Placement Optimization

5.4.1 Genome and Chromosome Description

A genome is the concatenation of two $n$-digit binary values that completely defines all possible locations in the Cartesian space. The number of digits $n$ is dictated by the size of the environment under observation. A chromosome is a chain of genomes that define the location of all $N_S$ sensors within the environment. For instance, given a $1000 - x - 1000km$ environment, two 10-digit binary values are required where the first 10-digits indicate the x-position and the second 10-digits indicate the y-position, accordingly. Therefore, a sensor $S_i$ genome of 10110010110010011010 is sufficient to designate the sensor location at (715,154). The genomes are concatenated together to form a chromosome (i.e., candidate solution). For example, a three sensor scenario is characterized by the following chromosome:

10110010110010011010 10110010110010011010 10110010110010011010

5.4.2 Mutation

Mutation is the change of a single bit in a chromosome so that its value reverts from 0 to 1 or vice versa. It is important to note that in this application of genetic algorithm, each genome represents a physical location in the environment. Arbitrary mutations may cause large or small adjusts with equal probability, that is, a mutation in the first section of the $n$-digit value will cause a large shift and a mutation in the latter section will cause a small shift. The probability of mutation $p_m$ then becomes a function of $n$ and factor $\kappa$

$$p_m(n, \gamma) = \frac{n^\kappa}{\text{geneSize}^\kappa} \ast \nu; \quad (5.11)$$

where $\kappa$ is a scaling term to adjust the intermediate n-digit values and $\nu$ is the maximum value for mutation probability at the $n^{th}$ digit. Figure 5.19 displays the probability of
mutation for $\kappa \in [1, 7]$ and $\nu = 0.2$. Small $\kappa$ values provide a linear rise in mutation probability along the genome whereas large $\kappa$ values are exponential. This formulation increases the chances of finer adjustments around a location but still allows for the potential for large shifts in position.

![Probability of Mutation](image)

**Figure 5.19: Probability of Mutation**

### 5.4.3 Objective Function

The objective is to place the static sensors in positions that provide optimal state estimation over the time series $t_0, \ldots, t_{final}$. The ideal solution are sensor positions that minimize the MSE, but truth knowledge is hardly available in the real world. One potential solution is to evaluate the covariance matrix, which provides insight into the change in state estimation. This is accomplished by incorporating the determinant of the covariance matrix into the objective function. Additionally, the MSE using the same sensor configuration may vary due to the uncertain nature of the sensor measurements. Therefore, a Monte Carlo for each sensor configuration is executed and the mean of the objective function for each target is computed.

- Truth Data
5. ALGORITHM DEVELOPMENT

- Minimize Average MSE

\[
f = \frac{1}{n} \sum_{i=t_0}^{t_f} (\hat{x}_{f,i} - x_i)^2, \quad \forall j = 1, \ldots, N_t
\]  

(5.12)

- Minimize Maximum MSE

\[
f = \min_j \max_{j} \frac{1}{n} \sum_{i=t_0}^{t_f} (\hat{x}_{f,i} - x_i)^2, \quad \forall j = 1, \ldots, N_t
\]  

(5.13)

- Sensor Based Data

- Minimize Average Determinant of Error Covariance Matrix

\[
f = \frac{1}{n} \sum_{i=t_0}^{t_f} \det P_{f,i}, \quad \forall j = 1, \ldots, N_t
\]  

(5.14)

- Minimize Maximum Determinant of Error Covariance Matrix

\[
f = \min_j \max_{j} \frac{1}{n} \sum_{i=t_0}^{t_f} \det P_{f,i}, \quad \forall j = 1, \ldots, N_t
\]  

(5.15)

5.4.4 Training

Caution must be exercised not to train the genetic algorithm to excel at one particular scenario while not remaining robust to other scenario formulations. A training scenario is defined as a subset of targets \((t \in T)\) based on the original scenario formulation. Prior to execution, targets are evaluated against their probability of run \(p(T_t)\) to determine if they are included in the training data set. At minimum, at least one target must be present in the training run. If no targets were pre-selected for the training run, then one target is selected at random.
6 Results

6.1 ARGOS

The best methodology to present ARGOS results is by listing representative use cases of the tool.

6.1.1 Sensor Placement Performance

Figure 6.1 provides an example of analyzing sensor placement algorithms. Once a scenario is defined, the user runs the simulation and saves the simulation data. Next, the user optimizes the sensor positions in the Sensor Placement UI. Once complete, the user
6. RESULTS

Table 6.1: Use Case Sensor Placement Optimization

<table>
<thead>
<tr>
<th>Target ID</th>
<th>Pre Optimization</th>
<th>Post Optimization</th>
<th>Percentage Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15.88 m</td>
<td>11.28 m</td>
<td>28.93%</td>
</tr>
<tr>
<td>2</td>
<td>21.61 m</td>
<td>12.91 m</td>
<td>40.26%</td>
</tr>
<tr>
<td>3</td>
<td>16.29 m</td>
<td>11.84 m</td>
<td>27.36%</td>
</tr>
<tr>
<td>4</td>
<td>22.94 m</td>
<td>13.85 m</td>
<td>39.63%</td>
</tr>
</tbody>
</table>

can save the sensor position data for the scenario. The user can subsequently run the simulation again with the new sensor locations. Finally, the user can analyze the performance by analyzing the two datasets, prior to and after sensor placement optimization. Figure 6.2 displays the analysis between the two scenarios with changes in the sensor locations marked in the sensor table. The numerical differences in target MSE are shown in table 6.1.

Figure 6.1: Sensor Placement Optimization Process Flow
6.1 ARGOS

Figure 6.2: Sensor Placement Optimization Analysis

6.1.2 Sensor Tracker Performance

Target trackers have strengths and weaknesses. Single mode CV Kalman Filters are ideal for non-maneuvering target trajectories, however struggle to provide strong state estimation (i.e., minimal mean square error) once the target performs a maneuver. Interacting Multiple Models (IMMs) utilize multiple trackers with weighted models, greatly improving the ability by embedding various models to account for different target trajectories. Yet, the strength of the IMM also can account for degraded performance if the target trajectory does not exhibit many different maneuvers and/or the IMM transition matrix settings. The tool is useful for comparing and contrasting different tracker configurations.

To analyze the performance of various tracker settings, the best scenario definition is to place all the sensors at the same geographical location, and then customize each sensor. There are two ways to analyze the performance of the sensors trackers: (a) based on the simulation, and (b) compared against different scenarios (i.e., saved datasets). Take for example the scenario shown in fig. 6.3.
6. RESULTS

This example demonstrates two sensors, a single mode CV tracker (Sensor ID-1) and a two mode (CV and CT) IMM tracker (Sensor ID-2). The results are based on 100 runs of the scenario and images are extracted directly from the ARGOS tool.

Comparison of the sensors within the same scenario is accomplished by running the simulation and analyzing the simulation results, as shown in fig. 6.4. This allows direct comparison of the sensors tracking performance (see fig. 6.5), as well as evaluation of IMM performance if applicable (see fig. 6.6). Sensor ID-2 shows improved tracking capability during the trajectory maneuver between time steps 15 and 35.

The use case example is extended to evaluate all three combinations of tracker settings (note that four combinations exist but since the sensors are geographically located at the same location, it is not necessary to evaluate both cases where one sensor is a single mode CV tracker and the second is a two-mode IMM tracker). After executing the three distinct scenarios and saving the data, then the three datasets are analyzed, as shown in fig. 6.7.
Figure 6.4: Simulation Results of Varying Trackers

Figure 6.5: Varying Trackers Data Comparison

Table 6.2: Tracker Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>2 KFs</th>
<th>2 IMM</th>
<th>1 KF &amp; 1 IMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Target MSE</td>
<td>23.25 m</td>
<td>12.87 m</td>
<td>18.68 m</td>
</tr>
<tr>
<td>Execution Time</td>
<td>0.8059 s</td>
<td>1.1191 s</td>
<td>1.0263 s</td>
</tr>
</tbody>
</table>
6. RESULTS

Figure 6.6: IMM Tracker Performance

Figure 6.7: Tracker Comparison
The dataset Use Case Tracker - 2 IMMs performs the best in terms of overall MSE. This can also be seen in the line chart that the IMMs perform much better during the two coordinate turns. However, Use Case Tracker - 2 IMMs is the worst in terms of execution time, due to the extra computation required of executing multiple models and fusing the results together for a single estimate. The overall target MSE and execution time for the three datasets are shown in table 6.2.

6.1.3 Scalability Analysis

Evaluation of the scalability of a scenario is achieved in one of two methods based on the objective of the analysis. If the user is interested in the execution time between only select datasets, then the user can do so as described in the Analysis section description. Both the 2-Dataset and N-Dataset comparison UIs provide execution times. The user may also evaluate scalability at a global level by clicking the View Scalability button on the Analysis UI. Each successful execution of the simulation stores the number of sensors, number of targets, and execution time (excluding time required for graphical updates to the UI). Furthermore, if the simulation executes 100 runs, then 100 time points are collected and stored. This analysis is more generic as individual settings may produce varying execution times, such as running a simulation for the same number of targets and sensors, but different cross-covariance approximation methods. Disregarding the cross-covariance terms in sensor fusion improves execution time by simplifying the inverse calculation of the covariance matrix. Finer details are captured in the 2- and N-Dataset comparison screens. Figure 6.8 depicts the image that is displayed.
6. RESULTS

6.1.4 Fusion Analysis

Adding sensors to the environment and fusing their estimates inherently improves the overall estimation of the target. However, the incremental gain in lowering the target’s overall MSE is diminished as the number of sensors increases. Figure 6.9 shows the ARGOS analysis UI comparing seven different scenarios, ranging from a single sensor to seven sensors in the environment. Figure 6.10 shows MSE and execution times of the data, extracted from fig. 6.9 for additional clarity.
6.1 ARGOS

Figure 6.9: Analysis of MSE Improvement vs Number of Sensors

Figure 6.10: Diminished Performance Gain with Increased Number of Sensors
6. RESULTS

6.1.5 Robustness Analysis

Using the same scenarios in section 6.1.4, the probability of sensor failure is set to 5% \( p(F_{s_1}) = 0.05 \). At each time step, \( p(F_{s_1}) \) is compared to a randomly generated, uniform distributed number to determine if the sensors fails. Once a sensor meets the criterion, then the sensor is rendered offline for the remainder of the simulation. Figure 6.11 shows the ARGOS 2-Dataset screen comparing the two scenarios: Use Case No Failure (2 Sensors) and Use Case Sensor Failure (2 Sensors).

![Figure 6.11: Comparative Sensor Failure Analysis](image)

Table 6.3: Comparison of Scenarios with Sensor Failure

<table>
<thead>
<tr>
<th>( N_S )</th>
<th>( p(F_{s_1}) = 0 )</th>
<th>( p(F_{s_1}) = 0.05 )</th>
<th>Decremented Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>12.8742</td>
<td>15.7015</td>
<td>-21.96%</td>
</tr>
<tr>
<td>3</td>
<td>11.6725</td>
<td>13.0211</td>
<td>-11.55%</td>
</tr>
<tr>
<td>4</td>
<td>11.1885</td>
<td>12.1988</td>
<td>-9.03%</td>
</tr>
<tr>
<td>5</td>
<td>11.2557</td>
<td>11.8269</td>
<td>-5.07%</td>
</tr>
<tr>
<td>6</td>
<td>10.7891</td>
<td>11.1894</td>
<td>-3.71%</td>
</tr>
<tr>
<td>7</td>
<td>10.7733</td>
<td>10.8669</td>
<td>-0.87%</td>
</tr>
</tbody>
</table>
6.2 Air Traffic Controller Warning System

Three scenarios are presented for conforming, non-conforming (diverted route), and non-conforming (spurious). A fourth case includes a higher density environment with a mixture of the three cases. All scenarios use two homogeneous sensors, correlates the sensor tracks, and fuses the results prior to executing the ATC fuzzy inference system. The first figure in each case is the visual display of target trajectory and ATC Warning Level at each individual time step (indicated by a colored circle). Graphical results are a single run and numerical plots are the average of 25 Monte Carlo simulations.

6.2.1 Conforming

Figure 6.12: Conforming ATC Plot

Table 6.4: Conforming ATC - Time in ATC Warning Level Categories

<table>
<thead>
<tr>
<th>Time Steps</th>
<th>Watch</th>
<th>Advisory</th>
<th>Warning</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage (%)</td>
<td>72</td>
<td>16</td>
<td>0</td>
<td>88</td>
</tr>
<tr>
<td>81.8</td>
<td>18.2</td>
<td>0</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

The conforming scenario represents aircraft that adhere to the specified flight path
6. RESULTS

Figure 6.13: Conforming ATC FIS Inputs

Figure 6.14: Conforming ATC Warning Level
and the sensor fused state estimate correlates well with the GPS based state estimate during the majority of the trajectory, staying in the watch category for 81.8% of the time. The 18.2% advisory category occurs during the two coordinated turns when TCP is incremented to the next way-point and the aircraft only begins its maneuver to realign its velocity vector with TCP. A simple illustrative example is shown in fig. 6.15; the yellow shaded area indicates that the angular displacement of the aircraft at time $k$ through $k+2$ has the strongest presence in the middle membership function. At time $k+3$ the near membership function has the greatest influence, subsequently lowering the ATC warning level in the scenario presented.

![Figure 6.15: Delayed Conformance](image)

### 6.2.2 Non-Conforming; Diverted Route

<table>
<thead>
<tr>
<th>Time Steps</th>
<th>Watch</th>
<th>Advisory</th>
<th>Warning</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage (%)</td>
<td>1.2</td>
<td>70.2</td>
<td>58.6</td>
<td>100</td>
</tr>
</tbody>
</table>

The diverted route scenario is an example of an aircraft re-routing a portion of the flight path (e.g., poor weather conditions, collision avoidance, etc.), without updating flight plan. The large spike in the ATC warning level indicates the diverted flight path.
6. RESULTS

Figure 6.16: Non-Conforming (Diverted Route) ATC Plot

Figure 6.17: Non-Conforming (Diverted Route) ATC FIS Inputs
6.2 Air Traffic Controller Warning System

Figure 6.18: Non-Conforming (Diverted Route) ATC Warning Level

The large angular displacements and $\chi^2$ value raise to the warning level to watch during the first leg of the diverted route. Although $\chi^2$ is still large, the velocity vector realignment towards TCP during the second leg informs ATC that the aircraft demonstrates a course correction back to the intended flight path.

6.2.3 Non-Conforming; Spurious Intent

<table>
<thead>
<tr>
<th>Time Steps</th>
<th>Watch</th>
<th>Advisory</th>
<th>Warning</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage (%)</td>
<td>43</td>
<td>4</td>
<td>23</td>
<td>90</td>
</tr>
<tr>
<td>Percentage (%)</td>
<td>47.8</td>
<td>4.4</td>
<td>47.8</td>
<td>100</td>
</tr>
</tbody>
</table>

The spurious scenario is the purposeful deception to mislead air traffic controllers in the position of the aircraft. The aircraft initially adheres to the intended flight path but deliberately diverts. The ATC warning level FIS immediately captures the deviated flight path raising the warning level from warning to watch. This is also shown in table 6.6 such that the spurious aircraft only spends four time steps in the advisory category due to the
6. RESULTS

Figure 6.19: Non-Conforming (Spurious Intent) ATC Plot

Figure 6.20: Non-Conforming (Spurious Intent) ATC FIS Inputs
Figure 6.21: Non-Conforming (Spurious Intent) ATC Warning Level

rapid change from watch to warning.
6. RESULTS

6.2.4 Combined

![Combined Scenarios ATC Plot](image)

**Figure 6.22: Combined Scenarios ATC Plot**

<table>
<thead>
<tr>
<th></th>
<th>Watch</th>
<th>Advisory</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conforming (%)</td>
<td>92.4</td>
<td>7.6</td>
<td>0</td>
</tr>
<tr>
<td>Non-Conforming (%)</td>
<td>0</td>
<td>57.8</td>
<td>42.2</td>
</tr>
<tr>
<td>Overall (%)</td>
<td>86.3</td>
<td>1.9</td>
<td>2.8</td>
</tr>
</tbody>
</table>

**Table 6.7: Combined ATC - Percentage Breakdown of ATC Warning Level Categories**
6.3 Fuzzy Clustering Assisted Track-to-Track Correlation

Four cases were tested with increased complexity to evaluate scalability and performance metrics across varying number of clusters. The simulation is executed for 50 timesteps for each scenario listed below. The results displayed are the mean value of the 50 timesteps across 10 Monte Carlo runs for each cluster. The $m$ value for the Sequential $m$-Best Assignment algorithm, for both the benchmark and clustering approach, is set to three ($m = 3$). The performance metric is a measure of accuracy of getting the correct track files in the tuple, $T = \{T_1, T_2, \ldots, T_{NS}\}$ where $k$ represents the target.

Table 6.8: Fuzzy Clustering Assisted Track-to-Track Correlation Test Cases

<table>
<thead>
<tr>
<th>Case</th>
<th>$NS$</th>
<th>$NT$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case A</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>Case B</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>Case C</td>
<td>10</td>
<td>25</td>
</tr>
<tr>
<td>Case D</td>
<td>5</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 6.9 shows the performance and execution time of both the benchmark and clus-
6. RESULTS

(a) Execution Time  
(b) Performance

Figure 6.24: Case A Performance Results

(a) Execution Time  
(b) Performance

Figure 6.25: Case B Performance Results

(a) Execution Time  
(b) Performance

Figure 6.26: Case C Performance Results
6.4 Fuzzy Mode Detection

Figure 6.27: Case D Performance Results

Table 6.9: Algorithm Comparison

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Sequential m-Best (Time)</th>
<th>Clustering (Time)</th>
<th>Time Improvement Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario A</td>
<td>1.00000 (0.14572 s)</td>
<td>0.99966 (0.02986 s)</td>
<td>4.880</td>
</tr>
<tr>
<td>Scenario B</td>
<td>1.00000 (2.89200 s)</td>
<td>0.99959 (0.18988 s)</td>
<td>15.231</td>
</tr>
<tr>
<td>Scenario C</td>
<td>0.99967 (12.5769 s)</td>
<td>0.99959 (0.57136 s)</td>
<td>22.012</td>
</tr>
<tr>
<td>Scenario D</td>
<td>1.00000 (11.3989 s)</td>
<td>0.99943 (0.44914 s)</td>
<td>25.380</td>
</tr>
</tbody>
</table>

tering approach. The last column displays the ratio of execution times by employing the clustering algorithm. Table 6.10 displays the number of clusters that provided the maximum accuracy prior to degrading performance as well as ratio of cluster to the number of targets in the environment.

Table 6.10: Ratio of Optimal Number of Clusters to the True Number of Targets

<table>
<thead>
<tr>
<th>Case</th>
<th>Number of Clusters</th>
<th>( N_T )</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case A</td>
<td>8</td>
<td>10</td>
<td>0.80</td>
</tr>
<tr>
<td>Case B</td>
<td>20</td>
<td>25</td>
<td>0.80</td>
</tr>
<tr>
<td>Case C</td>
<td>21</td>
<td>25</td>
<td>0.84</td>
</tr>
<tr>
<td>Case D</td>
<td>39</td>
<td>50</td>
<td>0.78</td>
</tr>
</tbody>
</table>

6.4 Fuzzy Mode Detection

Figure 6.28 depicts the aircraft trajectory using two motion models, constant velocity (CT) and coordinated turn (CT). The aircraft executes three coordinated turns at 3 degrees per second turn rates, with each turn duration varying in length. The pilot is assumed to use
6. RESULTS

inside turns and the timing around the turn location is based on the normal distribution shown in fig. 5.16b where $\sigma_i$ is 2 time steps.

Figure 6.28: Target Trajectory with 3 Way-Points

6.4.1 Single Run Analysis

An IMM tracker provides state estimation of the aircraft along with mode posterior probabilities, shown in fig. 6.29, over the length of the simulation. The red dotted line indicates the true aircraft mode where CV = 0 and CT = 1. Finding the maximum posterior probability for mode detection results in fast recognition times but may result in false positive mode change findings. Figure 6.29 shows the posterior probability of the CT model for the length of the simulation for a single run. At time step 104, the probability of mode
6.4 Fuzzy Mode Detection

CT exceeds 0.5, falsely predicting that the aircraft has entered a coordinate turn. The cumulative sum statistical test delays the prediction until a specified threshold is met (a large enough decrease in the S series value). Figure 6.30 displays the time series for the CUSUM test, with a zoomed in picture shown in fig. 6.31 at the time of the false mode change. Both the CUSUM and Fuzzy CUSUM detect a change (the drop in S between time steps 103 and 106) but the change is not significant enough.

![Figure 6.29: CV Posterior Model Probability](image)

Figure 6.29 graphically shows the time the mode change was detected by the three approaches, with numerical results in table 6.11.

<table>
<thead>
<tr>
<th>Way-Point 1</th>
<th>Way-Point 2</th>
<th>False Positive</th>
<th>Way-Point 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>29-51</td>
<td>81-92</td>
<td>N/A</td>
</tr>
<tr>
<td>Max $\mu_j$</td>
<td>32-53</td>
<td>83-94</td>
<td>104-104</td>
</tr>
<tr>
<td>CUSUM</td>
<td>35-56</td>
<td>86-97</td>
<td>N/A</td>
</tr>
<tr>
<td>Fuzzy CUSUM</td>
<td>33-55</td>
<td>85-95</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Based off the S time series, the CUSUM can predict when the mode changed, unlike the maximum posterior probability approach. Figure 6.31 shows the time corrected mode change, with numerical results in table 6.12.

Figure 6.32 shows the time at which a change in mode is predicted. The maximum
6. RESULTS

Figure 6.30: S Time Series

Figure 6.31: S Time Series False Positive Analysis

Figure 6.32: Mode Change Detection
6.4 Fuzzy Mode Detection

Table 6.12: Mode Time Detection Correction - Single Run

<table>
<thead>
<tr>
<th>Way-Point 1</th>
<th>Way-Point 2</th>
<th>False Positive</th>
<th>Way-Point 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>29-51</td>
<td>81-92</td>
<td>N/A</td>
</tr>
<tr>
<td>Max ( \mu_j )</td>
<td>32-53</td>
<td>83-94</td>
<td>104-104</td>
</tr>
<tr>
<td>CUSUM</td>
<td>30-52</td>
<td>82-93</td>
<td>N/A</td>
</tr>
<tr>
<td>Fuzzy CUSUM</td>
<td>30-52</td>
<td>82-93</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Figure 6.33: Mode Change Time Correction

The posterior probability approach promptly predicts the mode change in the fastest response, but as stated above, is prone to false positives, as shown by the blue spike at time 104. Since the CUSUM relies on the statistical test, there is a delay in mode change detection.

6.4.2 Monte Carlo Analysis - Graphical

Figure 6.34 through fig. 6.39 illustrate the statistics for the time of mode change detection and correction based on 5000 simulation runs. The first column is the transition from CV to CT; the second column is the transition from CT to CV. The first row is the maximum posterior probability, the second row is CUSUM, and the third row is FCUSUM. Each graph displays a histogram of the discrepancy from perfect mode detection and correction. The solid vertical bar represents the mean value. The dash vertical bars are references to the other methodologies for comparative purposes. Negative values are delays in mode detection and correction (that is, mode detection occurred after the transition). The ideal
6. RESULTS

time detection is a solid vertical bar at the zero point along the x-axis with a tight fitting distribution, meaning that the algorithm detected the mode change at the instance that it occurred. Since the maximum posterior probability only relies on the current value of the mode probability, its corrected time is the same as the time of detection.

Both the CUSUM and FCUSUM initially have larger delays in mode change detection since they require a threshold to be exceed from the cumulative time series. However, since the FCUSUM adjusts the sensitivity parameter using knowledge of the expected turn time, it is able to cause a larger change in the S series data, resulting in the threshold to be exceeded sooner.

Figure 6.34: Way-Point 1 Mode Time Detection
6.4 Fuzzy Mode Detection

Figure 6.35: Way-Point 1 Mode Time Correction

Figure 6.36: Way-Point 2 Mode Time Detection
6. RESULTS

Figure 6.37: Way-Point 2 Mode Time Correction

Figure 6.38: Way-Point 3 Mode Time Detection
6.4 Fuzzy Mode Detection

6.4.3 Monte Carlo Analysis - Numerical

Table 6.13 through table 6.16 display the mean discrepancy times for the CUSUM and Fuzzy CUSUM algorithms, along with the FCUSUM improvement. On average, the FCUSUM improved both mode change detection and time correction by 25%.

<table>
<thead>
<tr>
<th>Way-Point</th>
<th>CUSUM</th>
<th>Fuzzy CUSUM</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Way-Point 1</td>
<td>-6.63</td>
<td>-5.09</td>
<td>23%</td>
</tr>
<tr>
<td>Way-Point 2</td>
<td>-6.89</td>
<td>-5.13</td>
<td>26%</td>
</tr>
<tr>
<td>Way-Point 3</td>
<td>-6.56</td>
<td>-5.03</td>
<td>23%</td>
</tr>
</tbody>
</table>

Table 6.17 shows that both the CUSUM and FCUSUM algorithms drastically improvement false positive rates of mode estimation. However, the improved detection and correction of the FCUSUM does have an adverse affect of doubling the false positive rate.
6. RESULTS

Table 6.14: Mode Detection Discrepancy $CT - > CV$

<table>
<thead>
<tr>
<th>Way-Point</th>
<th>Fuzzy CUSUM</th>
<th>CUSUM Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-5.84</td>
<td>-4.27 27%</td>
</tr>
<tr>
<td>2</td>
<td>-6.42</td>
<td>-4.91 24%</td>
</tr>
<tr>
<td>3</td>
<td>-5.93</td>
<td>-4.33 27%</td>
</tr>
</tbody>
</table>

Table 6.15: Mode Correction Discrepancy $CV - > CT$

<table>
<thead>
<tr>
<th>Way-Point</th>
<th>Fuzzy CUSUM</th>
<th>CUSUM Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2.45</td>
<td>-1.85 25%</td>
</tr>
<tr>
<td>2</td>
<td>-1.99</td>
<td>-1.49 25%</td>
</tr>
<tr>
<td>3</td>
<td>-2.38</td>
<td>-1.79 25%</td>
</tr>
</tbody>
</table>

Table 6.16: Mode Correction Discrepancy $CV - > CT$

<table>
<thead>
<tr>
<th>Way-Point</th>
<th>Fuzzy CUSUM</th>
<th>CUSUM Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1.29</td>
<td>-0.99 23%</td>
</tr>
<tr>
<td>2</td>
<td>-2.25</td>
<td>-1.67 26%</td>
</tr>
<tr>
<td>3</td>
<td>-1.31</td>
<td>-0.98 25%</td>
</tr>
</tbody>
</table>

compared to the CUSUM. Nevertheless, the small probability of a mode detection false positive is small enough to be considered insignificant.

Table 6.17: False Positive Detection Rate

<table>
<thead>
<tr>
<th>Max $\mu_j$</th>
<th>CUSUM</th>
<th>Fuzzy CUSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>22%</td>
<td>0.04%</td>
<td>0.08%</td>
</tr>
</tbody>
</table>

6.5 Sensor Placement Optimization

Each target is assigned a probability of run $p(T_i)$ at scenario definition that is utilized within the sensor placement optimization. Figure 6.40 illustrates the targets in the environment and table 6.18 shows the probabilities for each target within each case. The number of training runs for Cases 1 and 2 is only one since all targets are in the optimization scheme. The third case executes the optimization with 10 training datasets to have a
good representation of the distribution of targets. The probability of run for targets 1-4 is set to one so target trajectories are always expected in the GA training. Case 2 considers an additional target $T_5$ to be considered at all times during the training as well. The final case assumes that $T_5$ may not always be present so $p(T_5) = 0.2$. For example, this target may be a private aircraft or UAV hobbyist typically only flying on the weekends.

The GA parameters are the same for each case: population size is 40, number of iterations is 25, and the number of Monte Carlo runs is 25. After execution of each case, the sensor states are saved and a simulation of the scenario is ran with 100 Monte Carlo runs to analyze the performance of the optimized sensor locations.

![Figure 6.40: Sensor Placement Locations based on Three Different Scenario Configurations](image)

**Table 6.18:** Probability of Target Run per Case

<table>
<thead>
<tr>
<th>Case</th>
<th>$p(T_1)$</th>
<th>$p(T_2)$</th>
<th>$p(T_3)$</th>
<th>$p(T_4)$</th>
<th>$p(T_5)$</th>
<th>Training Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.2</td>
<td>10</td>
</tr>
</tbody>
</table>
6. RESULTS

6.5.1 Minimize Average Target MSE

The objective of the first use case is how to optimally place two sensors within the environment based on the selected cost function, minimize average MSE, as described in eq. (5.12). Figure 6.41 depicts the locations of the sensors for each case listed in table 6.18.

![Figure 6.41: Sensor Placement Locations for Minimize Average Target MSE](image)

<table>
<thead>
<tr>
<th>Case</th>
<th>( T_1 )</th>
<th>( T_2 )</th>
<th>( T_3 )</th>
<th>( T_4 )</th>
<th>( T_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.31</td>
<td>13.12</td>
<td>12.09</td>
<td>13.82</td>
<td>32.49</td>
</tr>
<tr>
<td>2</td>
<td>13.16</td>
<td>17.89</td>
<td>12.74</td>
<td>19.16</td>
<td>20.10</td>
</tr>
<tr>
<td>3</td>
<td>10.75</td>
<td>17.66</td>
<td>11.03</td>
<td>18.62</td>
<td>25.02</td>
</tr>
</tbody>
</table>

Case 1 simply disregards any presence of \( T_5 \) and optimizes over targets 1-4. The sensors are placed near targets 1-4 resulting in best overall MSE for those targets but does not perform well for the unknown target \( T_5 \). Case 2 attempts to resolve the issue of unknown targets in the southeast corner. Here the sensors are more neutrally located, lowering the MSE for \( T_5 \) but at the cost of poorer performance for targets 1-4. Case 3 is a hybrid approach that assumes that a target may appear in the southeast corner, resulting in a
location between cases 1 and 2. Targets 1-4 have decremented performance in their MSE but not to the same extend as case 3. Likewise, it is still able to track $T_5$ at a better performance as case 1 but not at the same level of performance as case 3.

### 6.5.2 Minimize Maximum Target MSE

The objective of the second use case is how to optimally place two sensors within the environment based on the selected cost function, minimize maximum MSE, as described in eq. (5.13).

![Figure 6.42: Sensor Placement Locations for Minimize Maximum Target MSE](image)

Table 6.20: MSE per Target for each Test Case - Minimize Maximum MSE

<table>
<thead>
<tr>
<th>Case</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.80</td>
<td>12.20</td>
<td>14.49</td>
<td>13.12</td>
<td>32.31</td>
</tr>
<tr>
<td>2</td>
<td>19.03</td>
<td>17.30</td>
<td>19.44</td>
<td>18.05</td>
<td>16.55</td>
</tr>
</tbody>
</table>

Although the MSE values differ from the previous use case, the overarching results are the same. Case 1 disregards $T_5$ and results in a poor tracking performance. Case 2
6. RESULTS

includes $T_5$ and the MSE for all targets is relatively the same. Case 3 optimized over the training data when $T_5$ appeared 20% of the time, so MSE for $T_{1-4}$ increased (decreased) compared to Case 1 (Case 2) and the MSE for $T_5$ decreased (increased) compared to Case 1 (Case 2).
“If I were giving a young man advice as to how he might succeed in life, I would say to him, pick out a good father and mother, and begin life in Ohio.”

Wilbur Wright

“The best dividends on the labor invested have invariably come from seeking more knowledge rather than more power.”

Orville Wright

Conclusions and Future Work

Given the projected growth of manned aircraft and the integration of unmanned aerial systems into the National Airspace, air traffic controllers and sensor systems will be stressed to maintain optimal situational awareness. This dissertation explored various tools and algorithms to support the anticipated growth and integration of many more targets in the environment. This chapter provides conclusions and future work of each element of the thesis.
7. CONCLUSIONS AND FUTURE WORK

7.1 ARGOS

The ARGOS conclusions and future work is divided into a few different categories. First, the *Application Development* section discusses ARGOS as a research tool and the programmatic changes required for future growth. Second, *Algorithm Development* explores additional algorithms and model behavior for future integration to the tool. Third, *Version Development Plan* outlines the subsequent versions and associated features. Fourth, *Potential Applications* explores uses of the tool to specific real-world examples.

7.1.1 Application Development

The underlying architecture (MVC) is instrumental in transitioning from small scale research work into larger scale research projects. This framework reduces redundant code, adds stability to future releases through regression testing, and compartmentalizes the user view/presentation of the data and the model behavior. A developer can easily change the presentation of the data without affecting the underlying model behavior, and vice-versa with model changes. Although there are multiple ways to programmatically formulate the MVC architecture, an Object-Oriented Programming (OOP) design is the best methodology. MathWorks MATLAB is a well-understood and used application in engineering applications, and has included OOP in their core distribution. Therefore, this dissertation leverages MATLAB entirely to construct ARGOS v1.0.

But the use of OOP becomes computationally intensive for large scale problems as MATLAB is more designed for vector-based operations. The OOP approach includes additional memory overhead which can become cumbersome as the scenario scales in size. Furthermore, this approach requires the core MATLAB installation (and proper release) on the end user’s computer. MATLAB provides proprietary toolboxes to facilitate removing the reliance to have MATLAB on end user computers. MATLAB Compiler SDK provides functionality to construct C/C++, Java, and Python code packages; MATLAB
Coder creates standalone applications. Future releases removes the reliance on MATLAB core installation on end user computers. As ARGOS is expanded for additional functionality and usability, ARGOS v2.0 will also leverage various programming languages that are more suitable for particular tasks while still leveraging MATLAB for various processing in the background as necessary. Figure 7.1 is illustrative of the subsequent ARGOS release developed in Java Netbeans Platform with planet Earth created through NASA’s WorldWind Java SDK (75).

![Figure 7.1: ARGOS Version 2.0](image)

Version 3.0 extends the tool by integrating real-time sensor data directly into ARGOS. This may include ADS-B data broadcasted from an air vehicle to non-cooperative sensing capability such as RADAR and LIDAR sensors. ARGOS would provide a real-time view of the environment, including sensor data, aircraft data, tracking errors, and so forth as already seen within the tool. Furthermore, the data can be captured for analysis within the analysis component of ARGOS. For example, an end user can run simulation data prior to real-time feed data, then subsequently use the tool in the field to collect the live data. With both data sets, the end user compares the prediction to the actual in the
analysis section of ARGOS. Or similarly, an end user can compare different live data feeds for analysis of different scenarios.

On a lesser but equally important note is the development environment, commonly referred to as the IDE (Integrated Development Environment). Any program file can be edited by a text editor application. IDE’s provide enhanced functionality and support to developers for more rapid program development and fewer code issues. MATLAB’s IDE provides a built-in file editor but lacks the advanced features of more mainstream IDEs. For example, Netbeans(ref) and Ecplise(ref) IDEs includes code completion, highlights source code syntactically and semantically, easily refactors code, coding tips, code generators, etc. Although these IDEs are typically designed around a particular programming language (Netbeans - Java, Ecplise - C/C++), they have become flexible for development in other languages as well. In addition, these applications are free for use for private and commercial development purposes.

Although MVC is a commonly understood and popular architecture, alternative architecture patterns are available that extend the MVC design for specific purposes, including but not limited to, model-view-viewmodel, presentation-abstraction-control, model-view-presenter, and hierarchical model-view-controller. It would be worthwhile to explore alternative designs early on as ARGOS grows in size and complexity to ensure that a best fitting architecture pattern is used to support algorithm and application development.

Finally, one particular addition to support ARGOS is parallel processing capabilities to improve execution time. For example, the sensor placement optimization sequentially iterates over each candidate solution in the genetic algorithm algorithm. However, each candidate solution can be processed independently on a separate core, and then use the numerical results to determine candidate selection for the next generation. Also, each Monte Carlo run can be executed independently in a similar fashion. Furthermore, the fuzzy clustering assisted track-to-track correlation algorithm can also benefit from the parallel processing design since each cluster is independent of one another.
7.1.2 Algorithm Development

The underlying architecture (MVC) easily supports implementing other algorithms within the multi-sensor, multi-target environment. For example, algorithms of interest to integrate into ARGOS include path planning, conflict detection and resolution, and task assignment. Path planning is an imperative task for autonomous aircraft so they can travel from one point to the next while coping with potential obstacles. A variety of algorithms have been explored including but not limited to (with references): A* (76, 77), potential fields (78), genetic algorithms (79, 80, 81, 82), and fuzzy-based (83, 84, 85).

Conflict detection and resolution is a heavily researched field, in particular, different methodologies have been proposed to predict and correct potential aircraft collisions, such as probabilistic (74, 86, 87, 88), geometric (89, 90), covariance control (91), and stochastic (92). The expected growth in aircraft density and decrease in required spatial separation during the continuous roll-out of the FAA’s NextGen will require more intelligent conflict detection and resolution algorithms. ARGOS provides a base modeling and simulation framework to test and analyze different algorithmic approaches.

The current ARGOS environment is two-dimensional but a three-dimensional environment is required for more realistic and effectiveness application of different algorithm, in particular, conflict detection and resolution where altitude is another key component in collision detection and avoidance.

ARGOS sensors were described in section 4.3 with simple performance characteristics to demonstrate the effectiveness of the included algorithms in the dissertation work. As ARGOS continues to grow, these assumptions need to be removed for more realistic operation. The sensors must have a probability of detection less than 1.0 (i.e., a measurement is not guaranteed at each time step). Also, the probability of false alarms must be greater than 0.0 (i.e., false measurements that could be interpreted as coming from a target), requiring measurement-to-track data association techniques such as Probabilistic Data Association Filter (PDAF). Also, sensors are assumed to make a track file at first instance.
7. CONCLUSIONS AND FUTURE WORK

a measurement is received. However, a collection of plots or detections are required, typically adhering to the 'M-of-N rule', before a track is initiated. After a track is initiated, track maintenance governs the life of the track and specifies when the track file is deleted. Also, the ARGOS sensors are geographically static, not including the GPS satellites that provide aircraft positional data but those may be considered static as the orbital trajectory is not modeled. Manned and unmanned aircraft can also carry sensor equipment to monitor and make measurements. Future releases of ARGOS will include mobile sensors, which ultimately will require path planning and collision avoidance capabilities.

An assumption used in the above work is synchronous time of target measurements. In reality, this assumption is rarely true as each sensor works independently and makes measurements based on its particular system configuration. So measurements are not necessarily temporally concordant. This introduces challenges when fusing estimates at the fusion center. Integration of out-of-sequence measurements elevates the usefulness of ARGOS for potential field use of the tool. This topic has been researched and interested readers can review such work in (20, 62, 63, 64, 66, 93, 94, 95, 96, 97).

Another assumption was that measurement errors in the ADS-B data were not correlated. The positional error was based on Gaussian noise in the x- and y-direction. However, Mohleji (98) demonstrated that ADS-B position and error data is strongly time correlated and “... the time correlation of the position measurement errors are significantly reduced with the Gauss-Markov process compared with Gauss probability distribution.” Lowe (56) also showed this implementation in his Master’s thesis for predicting pilot intent and aircraft trajectory. ARGOS must be extended for more realistic modeling of ADS-B errors given the prominent presence of the technology in the NextGen system. More accurate models would improve the Fuzzy ATC system described earlier, collision avoidance systems, etc.

Finally, a fundamental problem in combinatorial optimization is the resource allocation problem. This problem often arises in multitarget, multisensor environments when sensors
are overburdened and one must decide which targets to track at what time with the limited set of resources. A variety of intelligent algorithms have been researched to find (near-) optimal solutions to these NP-hard problems. As aircraft density increases, new innovative algorithms are necessary to make real-time decisions with limited resources. ARGOS can provide a test bed to analyze the performance and effectiveness of these algorithms.

7.1.3 Version Development Plan

Table 7.1 lists the version development plan. Version numbering adheres to the major.minor/.maintenance/ format. The table lists only the major.minor version as maintenance tasks are unplanned and only intended for corrections to the current software as needed. The two major version changes are the software architectural change and integration with real-time data, two major milestones in advancing ARGOS. The minor changes are the application and algorithmic additions to the research tool.

Table 7.1: ARGOS Version Development Plan

<table>
<thead>
<tr>
<th>Version</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Dissertation Work</td>
</tr>
<tr>
<td>2.0</td>
<td>Conversion to C++/Java environment with potential hybrid environment with MATLAB</td>
</tr>
<tr>
<td>2.1</td>
<td>Time-Correlated ADS-B positional error</td>
</tr>
<tr>
<td>2.2</td>
<td>Collision detection and avoidance</td>
</tr>
<tr>
<td>2.3</td>
<td>Resource allocation algorithms</td>
</tr>
<tr>
<td>2.4</td>
<td>Path-planning algorithms</td>
</tr>
<tr>
<td>2.5</td>
<td>Track file enhancements (initiation and maintenance)</td>
</tr>
<tr>
<td>2.6</td>
<td>Enhanced analysis functionality</td>
</tr>
<tr>
<td>2.7</td>
<td>Parallel processing capability</td>
</tr>
<tr>
<td>2.8</td>
<td>Mobile sensors</td>
</tr>
<tr>
<td>2.9</td>
<td>Task assignment algorithms</td>
</tr>
<tr>
<td>2.10</td>
<td>Data fusion with asynchronous measurements</td>
</tr>
<tr>
<td>2.11</td>
<td>Higher-fidelity sensor models</td>
</tr>
<tr>
<td>2.12</td>
<td>Higher-fidelity aircraft models</td>
</tr>
<tr>
<td>3.0</td>
<td>Real-time System Operation with Sensor Data and Mapping Integration</td>
</tr>
</tbody>
</table>
ARGOS was primarily developed to support the growth in aircraft density and UAS integration into the National Airspace. In its most simplistic form, it serves as an educational tool to learn about multi-target, multi-sensor environments. However, the underlying structure permits flexibility in future use to leverage its functionality for other purposes by integrating the aforementioned extensions. For example, fire fighters battling wildland fires have limited situational awareness unless crews are sent into the danger zone on the ground or through air vehicles, both risky operations for the crew. Unmanned aerial vehicles are ideal tools to provide aerial surveillance due to their low cost of operation, rapid deployment, and reduce risk to human life. ARGOS can serve as a testing and analysis tool as well as a live command system in ARGOS v3.0. References to previous work in application of UAS, including intelligent control algorithms to minimize fire damage, for fire fighters during wildland fires are found in (99, 100, 101, 102, 103, 104).

The sensor placement optimization tool allows the flexibility of optimizing only a subset of the sensors. So in cases where static sensors are already established, such as in the case of the FAA, the tool could be used to evaluate where to place new sensors based on density prediction. The cost function could be extended to include additional parameters to determine the best sensor parameters to meet specified tracking objectives by the FAA. In other words, a sensor with extremely small measurement uncertainty is ideal but the financial costs outweigh the benefit of improved tracking capabilities.

NASA’s World Wind is not limited to planet Earth but also includes extraterrestrial datasets for the Moon, Mars, Venus, and Jupiter (105, 106). Exploration of extraterrestrial planets such as additional autonomous rovers on Mars using new innovative algorithms can be explored and analyzed.
7.2 Air Traffic Controller Warning System

We present an approach using angular displacement to the TCP and TCP+1 way-points, and the $\chi^2$ value between the fused sensor estimate and GPS state estimate to indicate which targets require additional monitoring, thereby reducing the information overload to the air traffic controller.

The tables of time spent in each warning level category is only beneficial for the conforming scenario. The amount of time spent in the advisory and watch categories for the non-conforming scenarios is dependent solely on when the aircraft departs from the flight path. However, the conforming scenario provides insight into baseline conformance level results, which can be adjusted by modifying the fuzzy inference rules to either raise or lower the warning level.

A Kalman Filter was also explored using only the constant velocity model. During linear trajectories of the flight path, the constant velocity model demonstrated a better mean square error (MSE) on the target tracking, consequently improving the $\chi^2$ value, and thereby lowering ATC warning level. However, the constant velocity model struggled during any maneuver, causing more time for the velocity vector to realign to the TCP along with a higher $\chi^2$ value due to the larger MSE. The decreased performance of the IMM during linear portions of flight are minute compared to the vastly improved tracking during maneuvers, resulting in more stable ATC warning levels.

The scenarios are limited to only flights with pre-planned flight paths. In the event that an aircraft diverts from a flight path, it would be informative to infer the new TCPs. For instance, it an aircraft diverts its route to maintain separation, new TCPs are generated and ATC warnings are reduced to indicate that the aircraft is following a new prescribed flight path. Mueller (73) and Krozel (107, 108) have both researched intent inference and path prediction that would provide increase situational awareness when computing warning levels for air traffic.
7. CONCLUSIONS AND FUTURE WORK

7.3 Fuzzy Clustering Assisted Track-to-Track Correlation

The objective of this work is to cluster the state estimates from sensor track files to reduce the dimensionality of the problem space to achieve real-time execution with near-optimal performance compared to a benchmark algorithm of a large multi-sensor, multi-target environment. Since the number of targets in the environment is not readily available, additional analysis was performed on the number of clusters to achieve the maximum performance prior to degradation. Given the assumption that all sensors provide target state estimation, the clusters elements are modified using the partition matrix to select the most appropriate state estimates to include in the assignment problem of the cluster.

The algorithm demonstrated the ability to achieve similar performance results (i.e., correct tuple formulation) compared to the benchmark Sequential m-Best Assignment algorithm around an order of magnitude faster in processing time. Execution improved exponentially as the number of clusters was increased. This is the result of smaller assignment problem formulations, reducing the number of likelihood function calculations and inherent matrix inversions. Theoretically, execution time can be further reduced by parallel execution S-D assignment problem for each cluster, since each cluster is independent from one another. Therefore, execution time is predicted roughly at the current processing time divided by the number of clusters plus some additional overhead, reducing execution time by another order of magnitude.

An interesting finding is the consistency of cluster ratio (∼ 80%) to the true number of targets. Since the simulation is over a time series, track files are likely to overlap. In these cases, it is best to group all the overlapping track files in one cluster and allow the assignment algorithm to formulate the proper tuples based on likelihood calculations.

Future work is to intelligently predict the number of targets in the environment, either based on subtractive clustering techniques or on available sensor data to more effectively utilize the cluster ratio. For example, the algorithm could leverage the likelihood function
presented earlier, iterate over different number of track files, and use the number of targets that maximum the likelihood function. In addition, incorporating a sliding time scale of state estimates may improve overlapping track files such that the cluster ratio could be increased. Also of interest is exploring various norm-inducing matrices (such as the Gustafson-Kessel algorithm) for the fuzzy clustering algorithm to better take advantage of the partition matrix when deciding which track files to add or potentially remove from a cluster. Finally, the architecture of the approach is not limited to the Sequential m-Best Assignment Algorithm only but other approaches such as Lagrangian Relaxation could be inserted in a plug-and-play structure to test overall flexibility.

7.4 Fuzzy Mode Detection

CUSUM outperforms the greedy heuristic of detecting mode changes by only evaluating the current mode posterior probability. The fuzzy CUSUM uses the predicted mode change based on pre-planned flight route to adjust the sensitivity parameter and showed an average overall improvement in mode detection and time correction by 25% over CUSUM. However, adjusting the sensitivity parameter makes the algorithm more susceptible to false positives and doubled the number of false positives compared to CUSUM. Nevertheless, increasing the false positive rate from 0.04% to 0.08% is considered acceptable due to the overall improvement stated above.

The algorithm began to struggle with mode detection and false positives under two different conditions: a) the turn duration was short (less than 5 time steps) and b) the turn rate was greater than 6 degrees per second. However, all algorithms under consideration (maximum posterior probability, CUSUM, and Fuzzy CUSUM) demonstrated poor performance. The performance of each algorithm is dependent on the mode posterior probability value of the IMM. In all cases, the lack of stable and proper mode posterior probability dramatically affected each algorithms ability to identify and correct.
7. CONCLUSIONS AND FUTURE WORK

The Fuzzy CUSUM demonstrated improved performance but further exploration of the algorithm is possible. First, the algorithm relies on the fact that the flight plan is known. Using intent inference and prediction, the algorithm could be extended to evaluate for robustness to unexpected deviations. Second, explore the inclusion of the threshold parameter into the fuzzy logic controller and its effect on performance. Third, explore other input parameters that may further improve performance, such as, IMM turn rate estimation.

7.5 Sensor Placement Optimization

Integration of the \( p(T_t) \) value for each target is key to ensure the algorithm does not train to a single scenario. Otherwise, the conditions must be very similar when executing another simulation to achieve comparable performance. But the down side is the amount of training sets required. For instance, the examples used in the results section used 10 training sets to properly model \( p(T_5) = 0.2 \). Since each training run independently determines whether the target is present based on its \( p(T_t) \) value, there is a chance that \( T_5 \) could have appeared 1 or 3 times out of 10 instead of the anticipated 2. Further increasing the training set size from 10 to 100 would more properly model the anticipation of \( T_5 \) appearing in roughly 20\% of the cases. But there is a linear rise in execution time, so increasing the training set size from 10 to 100 would increase execution time by 10 times.

Also, there was no limitation on geographically positioning of the sensors, either in relation to one another or from a geographical point-of-view. First, it is feasible that multiple sensors are stacked on top of one another, however, that potential result did not occur as the final location of the sensors were spatially separated due to the inherent benefit of the sensor fusion at their respective positions. Second, it is possible that a sensor is placed in a geographical location that is not accessible or possible due to the terrain (e.g., in a lake or steep cliff) or ownership rights. Future work could require a minimum separation
within the cost function to ensure a minimal spatial distance, however, it is suggested that this separation value be adjustable during the research analysis as the inherent nature of the data fusion will result in a natural separation. It is also suggested that there are areas where sensors cannot be placed by including a penalty on such solutions in the cost function. ARGOS could leverage this ‘off-limits’ zone in multiple places within ARGOS as well such as the path-planning extension mention earlier.
References


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


