



January 2017

An Analysis Of Drop Outs And Unusual Behavior From Primary And Secondary Radar

Nicholas Allen

[How does access to this work benefit you? Let us know!](#)

Follow this and additional works at: <https://commons.und.edu/theses>

Recommended Citation

Allen, Nicholas, "An Analysis Of Drop Outs And Unusual Behavior From Primary And Secondary Radar" (2017). *Theses and Dissertations*. 2157.
<https://commons.und.edu/theses/2157>

This Thesis is brought to you for free and open access by the Theses, Dissertations, and Senior Projects at UND Scholarly Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UND Scholarly Commons. For more information, please contact und.commons@library.und.edu.

AN ANALYSIS OF DROP OUTS AND UNUSUAL BEHAVIOR FROM PRIMARY
AND SECONDARY RADAR

by

Nicholas J. Allen
Bachelor of Science, University of North Dakota, 2015

A Thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

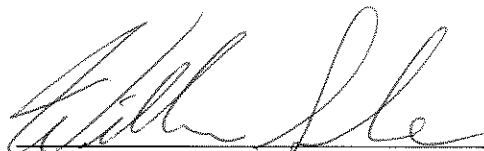
for the degree of

Master of Science

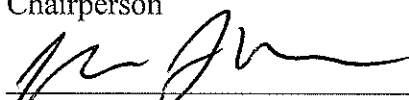
Grand Forks, North Dakota

August
2017

This thesis, submitted by Nicholas Allen in partial fulfillment of the requirements for the Degree of Master of Science from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.



Dr. William Semke
Chairperson

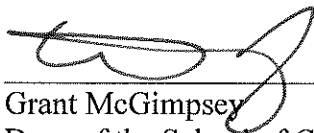


Dr. Jeremiah Neubert
Committee Member



Dr. Naima Kaabouch
Committee Member

This thesis is being submitted by the appointed advisory committee as having met all of the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.



Grant McGimpsey
Dean of the School of Graduate Studies

June 30, 2017

Date

PERMISSION

Title An Analysis of Drop Outs and Unusual Behavior from Primary and
 Secondary Radar

Department Mechanical Engineering

Degree Master of Science

In presenting this thesis in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis work or, in his absence, by the Chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this thesis or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my thesis.

Nicholas Allen
June 23, 2017

TABLE OF CONTENTS

LIST OF FIGURES	viii
LIST OF TABLES	x
ACKNOWLEDGEMENTS	xii
ABSTRACT	xiii
CHAPTER	
I. INTRODUCTION	1
Overview	1
Primary and Secondary Radar	3
Primary Radar	4
Secondary Radar	5
Radar Site & Classifications	7
Purpose of the Study	10
Previous Work	11
II. RADAR BEHAVIOR CLASSIFICATION ALGORITHM	13
Radar Data Anomaly Classification.....	13
Drop Outs	14
Altitude Discrepancies	15

Altitude Outliers.....	15
Prolonged Altitude Failure.....	15
Repeated Data.....	16
Multiple Aircraft with the Same Identification Number.....	16
Main Script.....	16
Data Read.....	23
Multiple Aircraft and Repeated Data.....	24
Drop Outs.....	27
Altitude Outliers.....	29
Prolonged Altitude Failure.....	30
Google Maps Plot & Draw Ellipse.....	31
Overlapping Analysis.....	33
Algorithm Verification.....	34
III. SUMMARY OF DATA ANOMALY RESULTS.....	36
Graphical Representation of Data Anomalies.....	37
Aircraft Count & Time Duration Summary.....	41
Time of Day & Location Summary.....	46
IV. DISCUSSION OF RESULTS.....	50
Overlapping Analysis.....	50
Hazard Assessment.....	52

Climate Effects.....	62
Comparison of Results to Previous Studies	67
V. CONCLUSIONS.....	71
APPENDIX A: MODE A TRANSPONDER IDENTIFICATION NUMBERS.....	74
APPENDIX B: MATLAB CODE	76
Main Script.....	76
Data Read.....	82
Multiple Aircraft and Repeated Data.....	83
Drop Outs.....	85
Altitude Outliers.....	88
Prolonged Altitude Failure.....	90
Plot Google Map.....	94
Draw Ellipse.....	107
Overlapping Analysis.....	110
APPENDIX C: RADAR DATA ANOMALIES SUMMARY.....	117
Aircraft Count/Location Summary	117
Drop Outs.....	119
Altitude Outliers.....	121
Prolonged Altitude Failures	123
Multiple Aircraft & Repeated Data.....	125

APPENDIX D: OVERLAPPING ANALYSIS RESULTS	127
Drop Outs	127
Altitude Outliers.....	128
APPENDIX E: CLIMATE EFFECTS SUMMARY	129
Fargo	129
Finley	131
REFERENCES	133

LIST OF FIGURES

Figure	Page
1. Radars at Fargo and Finley	2
2. Primary Radar Schematic	5
3. Secondary Surveillance Radar Schematic	7
4. Effective Radar Radii of the Fargo & Finley Radars.....	8
5. Aircraft Detected by the Fargo & Finley Radars	8
6. Azimuth and Range Diagram.....	19
7. Ground Range, Slant Range, and Altitude of Aircraft.....	19
8. Main Script Flow Chart	23
9. Multiple Aircraft/Repeated Data Function Flow Chart	26
10. Drop Out Function Flow Chart.....	28
11. Altitude Outliers Function Flow Chart	30
12. Prolonged Altitude Failure Function Flow Chart	31
13. Overlapping Analysis Algorithm Flow Chart.....	34
14. Drop Out Example Plot.....	38
15. Outlier Example Plot.....	38
16. Prolonged Altitude Failure Example Plot.....	39
17. Repeated Data Example Plot	39
18. Non-Discrete Multiple Aircraft Example Plot.....	40

19. Discrete Multiple Aircraft Example Plot	40
20. Drop Out Location Plots for Finley (left) and Fargo (right).....	47
21. Altitude Outlier Location Plots for Finley (left) and Fargo (right).....	47
22. Prolonged Altitude Failure Plots for Finley (left) and Fargo (right)	48
23. Repeated Data Location Plots for Finley (left) and Fargo (right).....	48
24. Multiple Aircraft with the Same Discrete ID Location Plots for Finley (left) and Fargo (right)	49
25. Multiple Aircraft with the Same non-Discrete ID Location Plots for Finley (left) and Fargo (right)	49
26. Wind Turbine Locations in the Fargo/Finley Radar Coverage Area	68
27. Close Up of the Wind Turbine Locations Near Fargo and Finley	69

LIST OF TABLES

Table	Page
1. Summary of Mode A Transponder Identification Codes.....	6
2. Summary of Radar Specifications	9
3. Data Removal Summary.....	18
4. Data Saved by the MATLAB Algorithm.....	22
5. Thresholds Used in the Multiple Aircraft and Repeated Data Function.....	25
6. Drop Out Threshold Values	27
7. Time Interval Classifications for Fargo and Finley	28
8. Ellipse Function Input Values.....	33
9. Aircraft Count Summary for Fargo and Finley.....	43
10. Drop Out Summary for Fargo and Finley.....	44
11. Prolonged Altitude Failure Summary (PAF) for Fargo and Finley	45
12. Radar and Transponder Failure Summary	51
13. FAA Severity Definitions	53
14. FAA Likelihood Definitions	53
15. Radar Data Anomaly Likelihood Definitions.....	53
16. Generic FAA Risk Matrix.....	54
17. Overall Radar Phenomena Risk Matrix	57
18. March Radar Phenomena Risk Matrix.....	58

19.	June Radar Phenomena Risk Matrix.....	59
20.	September Radar Phenomena Risk Matrix.....	60
21.	December Radar Phenomena Risk Matrix.....	61
22.	Fargo Climate Effects	63
23.	Finley Climate Effects	64
24.	Fargo Climate Effects for the Week of June.....	66

ACKNOWLEDGEMENTS

Firstly, I would like to thank my academic advisor, Dr. William Semke, for his patience, support, and guidance throughout my graduate career at the University of North Dakota (UND). I would like to thank my other committee member, Dr. Jeremiah Neubert for encouraging me to strive to do my best and for all his advice and support throughout the years. I would also like to thank my final committee member Dr. Naima Kaabouch for her advice and support as well.

This research would not have been possible without the initial funding provided by the ASSURE A6 Project: Surveillance Criticality. The work continued to be funded by the UND Mechanical Engineering Department and Unmanned Aircraft Systems Engineering (UASE) Laboratory. I would also like to thank Asma Tabassum, another student in the UASE Laboratory for her advice and support throughout the past semesters.

Finally, I thank my parents, John and Julie Allen, and my friends and family for their love, support, and encouragement. This would not have been possible without them. I appreciate everything you have done for me and continue to do for me. Lastly, I thank God for all the wonderful gifts and blessings he has provided in my life.

ABSTRACT

An evaluation of the radar systems in the Red River Valley of North Dakota (ND) and its surrounding areas for its ability to provide Detect and Avoid (DAA) capabilities for manned and unmanned aircraft systems (UAS) was performed. Additionally, the data was analyzed for its feasibility to be used in autonomous Air Traffic Control (ATC) systems in the future. With the almost certain increase in airspace congestion over the coming years, the need for a robust and accurate radar system is crucial. This study focused on the Airport Surveillance Radar (ASR) at Fargo, ND and the Air Route Surveillance Radar at Finley, ND. Each of these radar sites contain primary and secondary radars.

It was found that both locations exhibit data anomalies, such as: drop outs, altitude outliers, prolonged altitude failures, repeated data, and multiple aircraft with the same identification number (ID) number. Four weeks of data provided by Harris Corporation throughout the year were analyzed using a MATLAB algorithm developed to identify the data anomalies. The results showed Fargo intercepts on average 450 aircraft, while Finley intercepts 1274 aircraft. Of these aircraft an average of 34% experienced drop outs at Fargo and 69% at Finley. With the average drop out at Fargo of 23.58 seconds and 42.45 seconds at Finley, and several lasting more than several minutes, it shows these data anomalies can occur for an extended period of time. Between 1% to 26% aircraft experienced the other data anomalies, depending on the type of data anomaly and location. When aircraft were near airports or the edge of the effective radar radius, the largest proportion of data anomalies were experienced. It was also discovered that drop outs, altitude outliers, and

repeated data are radar induced errors, while prolonged altitude failures and multiple aircraft with the same ID are transponder induced errors. The risk associated with each data anomaly, by looking at the severity of the event and the occurrence was also produced. The findings from this report will provide meaningful data and likely influence the development of UAS DAA logic and the logic behind autonomous ATC systems.

CHAPTER I
INTRODUCTION

Overview

An analysis of radar performance in the Red River Valley of North Dakota (ND) and its surrounding area was completed to understand vulnerabilities and assess the current radar environment. From the literature review performed, this type of study is the first of its kind for information regarding radars open to the public. As unmanned aircraft systems (UAS) and autonomous air traffic control (ATC) towers are integrated into the aviation industry, the vulnerabilities of radar systems need to be understood to make reliable and robust designs. This study looked at radar sites in Fargo, ND and Finley, ND, which cover the states of North Dakota, South Dakota, Minnesota, and the provinces of Manitoba, Ontario, and Saskatchewan. These radars can be seen in Figure 1 [1, 2]. This study was funded by the Federal Aviation Administration's (FAA) Alliance for System Safety of UAS through Research Excellence (ASSURE), Harris Corporation, the University of North Dakota (UND) Mechanical Engineering Department, and the Unmanned Aircraft Systems Engineering (UASE) Laboratory. What began as supplemental work for the ASSURE A6 Project that investigated surveillance criticality for UAS, spun off into the research presented in this document.

ASSURE is an alliance of universities across the United States with partnerships from companies that help guide and fund the research being conducted. The mission of this

group is to provide the FAA with the research needed to quickly, efficiently, and safely integrate UAS into the national airspace (NAS) [3]. This partnership between the government, research universities, and industry will allow the UAS industry to grow and support the needs of the world's economy.



(a.) Fargo [1]



(b.) Finley [2]

Figure 1. Radars at Fargo and Finley

This study began by assessing the radar environment at Fargo and Finley on one day: June 15, 2015 for the ASSURE project. This analysis was all done by hand looking at the data, which showed many anomalies and unusual behavior exhibited by the radars. Some behaviors identified included: drop outs, altitude deviations, and multiple aircraft with the same aircraft identification number [4]. These behaviors, along with several others will be discussed in detail in Chapter II. This anomalous behavior can pose threats to the aircraft and people due to loss of well clear, additional stress on ATC personal, and failure of autonomous systems on UAS. After the realization of these data anomalies, follow on work was justified, which looked at one week of data in each season of the year. The dates of March 1-7, 2015; June 1-7, 2015; September 1-7, 2015; and December 1-7, 2015 were studied for both the Fargo and Finley radar locations. In order to process this extremely large amount of data, an algorithm was also developed to identify and classify the various

anomalous behavior initially discovered for the ASSURE study. What took weeks of time for the initial study just looking at June 15, took hours to process the months' worth of data with the algorithm.

Primary and Secondary Radar

Radar was originally an acronym that stands for radio detection and ranging. These systems have evolved substantially since their early days and are used for a wide variety of applications nowadays. A few of the many applications include: tracking aircraft, collision avoidance, weather observations, Earth resource monitoring, and mapping. Modern radars can track, identify, classify, and image targets while reducing or eliminating clutter and jamming [5].

While radar is one of the oldest technology to support the FAA's airspace monitoring, several other systems are used to supplement the radar systems, namely: traffic collision avoidance system (TCAS), automatic dependent surveillance – broadcast (ADS-B), and even simply the pilot's ability to see and avoid other aircraft. The current TCAS system was developed in the 1980's, and is a system that functions independently from the ground based ATC system [6]. TCAS provides pilots with traffic alerts and resolution advisories to prevent midair collisions with other transponder equipped aircraft. A traffic advisory instructs the pilot to visually search for other aircraft in the area. While a resolution advisory instructs the pilot to perform a certain maneuver to avoid the intruding aircraft which is inside the pre-defined envelope of the ownship aircraft. ADS-B is a real-time global positioning system (GPS) system which broadcasts the ownship's information, such as position, heading, and altitude [7]. This system allows the aircraft information to be received by any receiver within range of the aircraft. Lastly, every pilot is required, per

the FAA to see and avoid all aircraft according to §14 CFR 91.113 [8]. All these systems help supplement the existing radar network across the United States.

The FAA uses two primary radar systems to monitor the NAS over the United States, namely: primary radar and secondary radar. Both types of radar are used to help ATC monitor and control the air traffic in their respective regions. With rich histories dating back before World War II, these systems have matured over the following years. While primary and secondary radars complement each other, they operate on different principles which are described in the subsequent subsections.

Primary Radar

Primary radar is often thought of as traditional radar. This system sends high frequency pulses that are reflected off the target and received again; this reply is typically referred to as an echo [9]. This system is passive, which means that no equipment on the aircraft is needed for detection. The only piece of equipment needed is the ground radar station. Having a passive system is very important, because it allows rogue aircraft to be detected. An example of this is during the terrorist attacks on September 11, 2001 in the United States. The terrorist pilots turned off the aircraft transponders which made the aircrafts invisible to ATC [10]. The primary radar systems couldn't detect the aircraft in the city skyline, but in most other environments they would have been. However, despite the advantages of being a passive system, primary radar has several disadvantages. A large antenna is typically required, the system provides only position information, and is prone to clutter (unwanted replies from buildings or antenna sidelobes) [11]. The power loss equation is given by

$$P_{PSR} \sim \frac{1}{R^4}, \quad (1)$$

where P_{PSR} is the power of the primary surveillance radar and R is the range [9]. The basic schematic of a primary radar system can be seen in Figure 2 [5]. The major elements to this system are the transmitter, receiver, antenna, and signal processor.

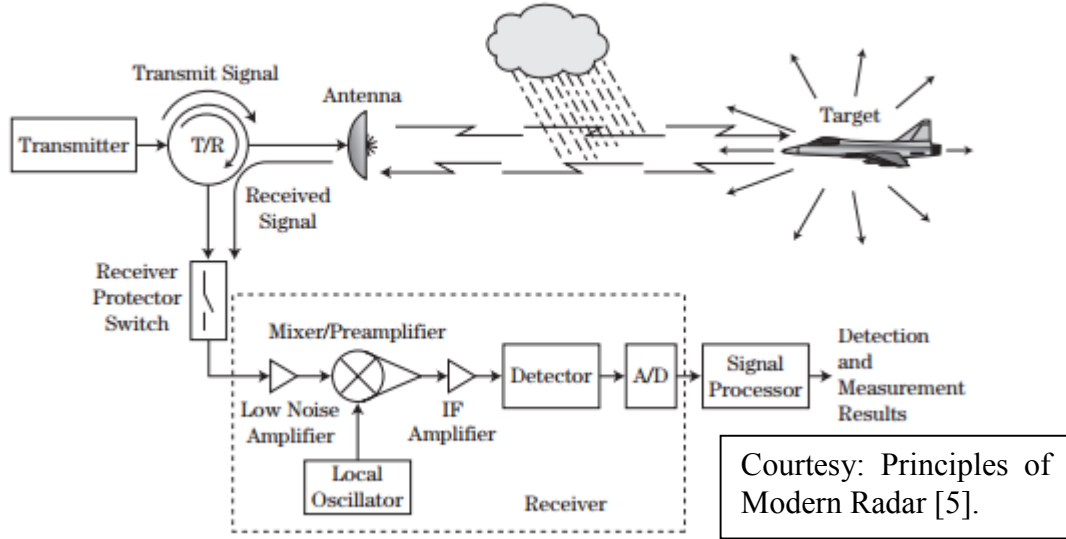


Figure 2. Primary Radar Schematic

Secondary Radar

Secondary radar is slightly different, where the aircraft needs to have a transponder onboard that replies to interrogations [9]. The interrogations are high frequency impulses, just like primary radar. In fact, the secondary radar transmitter is usually mounted on the antenna of a primary radar unit. The primary disadvantage of secondary radar is that it is an active system, which requires a transponder. The advantages of this radar system include not being effected by ground clutter or weather, and also having a longer range, because the pulse sent out only needs to travel one direction (the transponder reply is a different pulse) [12]. The power loss equation for secondary radar is given by

$$P_{SSR} \sim \frac{1}{R^2}, \quad (2)$$

where P_{SSR} is the power of the secondary surveillance radar and R is the range [9].

In fact, all aircraft are required to have a transponder that is capable of reporting the altitude and aircraft identification number according to the FAA’s Aeronautical Information Manual section 4-1-20 [13]. These transponders can be broken down into many classes. The two that concern this study are Mode A and Mode C. Mode A reports the aircraft identification number, while Mode C reports the barometric altitude from the aircraft’s barometric altimeter. A summary of a few of the aircraft identification numbers can be seen in Table 1 [14]. A full list of all the possible numbers are provided in Appendix A: Mode A Transponder Identification Numbers [14]. Each time an aircraft takes off, it is assigned one of the possible combinations of 4096 identification codes [15], which the pilot sets before takeoff.

Table 1. Summary of Mode A Transponder Identification Codes

Code	Description
0000	Should never be assigned
0100 – 0400	Allocated to Service Area Operations for assignment for use by Terminal/CERAP, NAS Stakeholder, Unique Purpose and Experimental activities.
1200	Visual Flight Rules (VFR) aircraft that may or may not be in radio contact with an ATC Facility.
5000 – 5057	Reserved for use by DOD
7601 – 7607	Allocated by the FAA for special use by Federal Law Enforcement
7400	Reserved for UAS experiencing a lost link situation
7500	Hijack in accordance with FAA JO 7610.4.

One of the biggest challenges of using two radar systems is syncing the data of both the primary and secondary radar systems. A brief schematic of the process can be seen in Figure 3 [16]. The ground unit, often called the interrogator sends a coded pulse to the

transmitter. Following this the transponder receives the pulse, decodes it, and transmits the appropriate response. The interrogator on the ground receives the reply, decodes it, and syncs the data with the primary data on the radar display.

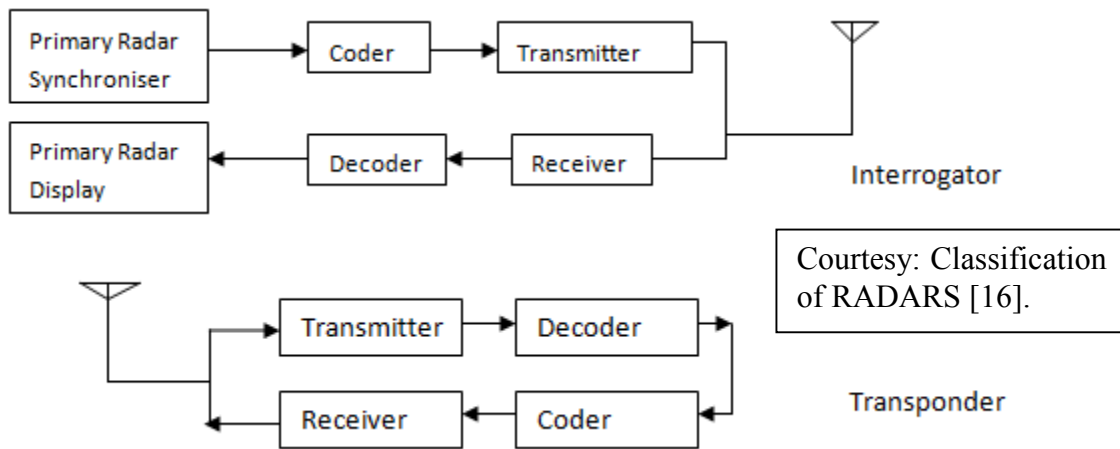


Figure 3. Secondary Surveillance Radar Schematic

Radar Site & Classifications

The Fargo and Finley radars cover a wide area around the Red River Valley and upper Midwest. Figure 4 shows the effective radar radius of both radars. The red circle represents the range of the Finley radar, while the blue circle represents the Fargo radar range. As a result, a wide variety of aircraft can be intercepted by the radars, while the exact aircraft can't be identified, because the Mode A transponder numbers change multiple times a day, even for the same aircraft. The general type of aircraft can be deciphered from the Mode A ID, along with knowledge of the type of aircraft that fly over the airspace. Figure 5 shows a few of the many aircraft that are intercepted by the Fargo and Finley radar, namely a Cessna and helicopter from the UND training fleet, a Delta commercial regional jet, and an Allegiant commercial jet [17-20].

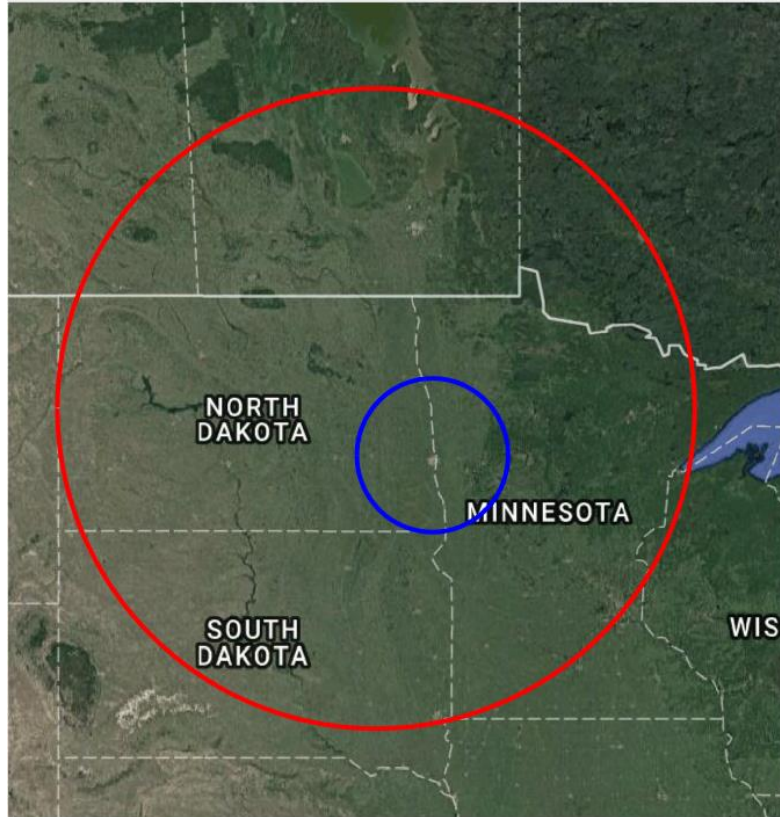


Figure 4. Effective Radar Radii of the Fargo & Finley Radars



Courtesy: [17-20]

Figure 5. Aircraft Detected by the Fargo & Finley Radars

The location not only varies between Fargo and Finley, but also the radar type. Fargo is classified as an Airport Surveillance Radar (ASR), or an ASR-11; while Finley is classified as an Air Route Surveillance Radar (ARSR), or an ARSR-4. The ASR-11 is an integrated primary and secondary radar system that is deployed at terminal air traffic control sites. The range of this system is 60 nautical miles [21]. This radar system is primarily used for approaches and flight plans around towered airports. The ARSR-4 is a long range surveillance radar with a range of up to 250 nautical miles [22]. These radar systems are used primarily for cross country or long range flight paths, using both primary and secondary radar as well. Table 2 provides a summary of the power, frequency, rotation rate, and other factors of the ASR-11 and ARSR-4 radars (highlighted in yellow) [23]. With a rotation rate of 12.5 RPM, the ASR-11 has a scan rate of 4.8 seconds. The ARSR-4 has a rotation rate of 5.0 RPM, which indicates a scan rate of 12 seconds. The scan rate is the amount of time for the radar unit to turn one full rotation (360 degrees).

Table 2. Summary of Radar Specifications

	TDWR (Raytheon)	ASR-9 (Northrop Grumman)	ASR-11 (Raytheon)	ARSR-4 (Northrop Grumman)
Transmitter				
Frequency	5.5 - 5.65 GHz ~ C Band	2.7-2.9 GHz	2.7-2.9 GHz	1.2-1.4 GHz
Polarization	Linear	Linear or Circular	Linear or Circular	Linear or Circular
Peak Power	250 KW	1.1 MW	20 kw	60 kw
Pulse Width	1.1 μ s	1.0 μ s	1.0 μ s, 80 μ s	150 μ s
PRF	2000 (max)	2 CPIs (~ 1000 Hz avg.)	4 CPIs (~ 1000 Hz avg.)	9-pulse CPI at variable spacing (288 Hz avg)
Receiver				
Sensitivity	0 dBz @ 190 km 1 m ² @ 460 km	0 dBz @ 20 km 1 m ² @ 111 km	0 dBz @ 20 km 1 m ² @ 111 km *	0 dBz @ 10 km 1 m ² @ 370 km
Antenna				
Elevation Beamwidth	0.55 Degrees (min)	5 Degrees	5 Degrees	2 Degrees (stacked)
Azimuth Beamwidth	0.55 Degrees	1.4 Degrees	1.4 Degrees	1.4 Degrees
Power Gain	50 dB	34 dB	34 dB	35 dB (transmit), 40 dB (receive)
Rotation Rate	5 RPM (max)	12.5 RPM	12.5 RPM	5.0 RPM
* 17dB sensitivity reduction in short-pulse processing range (0 - 6.5 nmi)				

Courtesy: [23]

Purpose of the Study

The primary purpose of this study was to assess the radar network in the surrounding area to understand vulnerabilities in the system. As UAS are integrated in the NAS, these vulnerabilities need to be understood to account for autonomous systems on the UAS and to better prepare pilots if radar data is used on the flight operations. With the New York Times dubbing North Dakota and specifically the Red River Valley area as the “Silicone Valley for drones” [24], it’s only a matter of time before these aircraft fill the skies. This claim to fame makes this study in the area especially important. The FAA also just published Part 107 in August 2016, which is the first step forward for commercial operations of UAS in the United States [25]. In fact, the FAA is predicting that there will be 7 million drones that could be active in the airspace by 2020 [26]. The United States Government Accountability Office also predicts that UAS will add \$82.1 billion to the economy over the next 10 years [27]. These numbers just show the importance for this study, especially if UAS use radar for Detect and Avoid (DAA) operations.

In addition to UAS, ATC centers will likely one day be autonomous. In fact, one group of researchers has looked at how to deal with conflict resolution and flight operation services for an autonomous system [28]. Additional publications have worked on other aspects of an autonomous ATC system, such as algorithms for short term conflict avoidance for UAS in an autonomous ATC [29], a network based ATC paradigm [30], and an autonomous ATC system for non-towered airports [31]. Additionally, several articles have stressed the importance of autonomous ATC systems and ATC systems for UAS [32, 33], because the man force required for providing ATC services for manned and unmanned vehicles is not sustainable. Right now if an ATC tower loses radar contact with an aircraft

even for one radar scan rate, they need to reestablish contact. ATC personal also instruct pilots to turn off their Mode A transponders if they are providing erroneous data. These are just two examples of the many functions humans provide if the radar system is not functioning properly. Knowing what types of failures radar experience and how to address them will help build a robust autonomous system without the interaction of humans.

Previous Work

Radar systems have evolved substantially since their inception dating back to the early 1900's. Despite this fact, radar, just like any other complex system is prone to problems. Many researchers have performed studies identifying these issues. Some work has been done on the reliability/performance of current radars, while others look at future applications. However, from the literature review performed, no work to date has identified radar data anomalies to the scale of the study presented in this paper.

Several reports have been published covering surveillance resolution, measurement errors, and coverage of various radars. Lieutenant Colonel Thomas prepared maps of radar coverage throughout the continental U.S. to identify blind spots in the U.S. radar network [34]. Healy, McDonald, Pomrink, and Conklin developed an operational test and evaluation of the ARSR-4 radar system and found that most basic functions were performed well [35]. Weber and Schanne also performed a similar analysis for the ASR-11 radar system [36]. A comparison and analysis of azimuth errors between the ASR-11 and ASR-9 radars was completed by Mayer and Tzanos [37]. A system error assessment for the ARSR-4 radar was performed by Busch and Bradbury [38]. Some specific studies were done on secondary radar identifying the vulnerabilities [39, 40]. One vulnerability included the potential of having an aircraft interrogated up to 20 times in one radar scan,

leading to congestion in the bandwidth and erroneous data. Additionally, there is no difference between a Mode A or Mode C response from a transponder. If a Mode A interrogation is issued, a Mode A response is expected, and likewise for Mode C. This could lead to problems decoding the response if more than one ground station was interrogating an aircraft. Lemmon, Carrol, Sanders, and Turner even performed an assessment of the effects of wind turbines on radars [41].

While some studies have identified the problems of the radar network in the U.S., others have looked at addressing the problems by developing the next generation of radar systems. Immoreev and Taylor discuss new capabilities for civilian radar by simply increasing the bandwidth [42]. Buckler discusses the benefits of using a phased array radar to replace current radars [43]; the FAA even published a document highlighting the national requirements for a phased array radar [44]. When arrays are implemented, the multiple secondary radar signals will need to be separated, which is why Petrochilos, Galati, Mené, and Pracci developed a simple and robust algorithm to do so [45]. Additionally, ADS-B and radar data fusion will be essential for ATC in the future, so several studies have theorized and analyzed this data fusion [46, 47].

CHAPTER II

RADAR BEHAVIOR CLASSIFICATION ALGORITHM

This chapter will provide an overview and an explanation behind the logic of the MATLAB algorithm developed to analyze and classify the radar data into the various different anomalies described in the first section of this chapter. The spreadsheets of data that Harris provided were separated by hour. For the analysis done in this paper, each hour of data for each day (24 hours total) was compiled into one spreadsheet. This was done for each day at each location, which resulted in 56 files, at a size of 4.28GB or 19,164,844 total lines long. This sheer amount of data justifies the need for an algorithm to process it. What takes hours for the computer to process, would take months to process by hand. Of the 50 columns of data contained in each spreadsheet, the program reads only time, aircraft ID, altitude, azimuth, and range. The following sections describe how each function and the main script operates. The entire algorithm can be found in Appendix B: MATLAB Code.

Radar Data Anomaly Classification

From the initial study performed, a variety of data anomalies were identified which included: drop outs, altitude outliers, prolonged altitude failures, repeated data, and multiple aircraft with the same identification (ID) number. These behaviors directed how the algorithm described later in this chapter was written. The majority of the anomalies discovered were associated with altitude from the secondary radar. However, position from

the primary radar was used to help identify repeated data and multiple aircraft with the same (ID) number. In some cases, aircraft experience several modes of unusual behavior. For example, the same aircraft may experience a drop out and an altitude outlier within a short period of time. The following sections in this chapter provide a description and a visual example of each data anomaly.

Drop Outs

A drop out occurs when there is a missed data point that should have been logged, which means the time between two data points is larger than the radar scan rate. Drop outs ranged in time from anywhere slightly greater than one radar scan rate to several minutes. The end limit was put in place to ensure that if aircraft went out of range for a period of time and came back into the radar's detection area it was not considered a drop out. A plot depicting a drop out can be seen in Figure 14 in Chapter III. Drop outs prevent ATC from getting updated information on the aircraft, which prevents up to date radar aircraft separation services. The typical separation of aircraft is 5 miles enroute and 3 miles in a terminal environment, or 1000 feet and 500 feet vertically, respectively [48]. To put the significance of drop outs in perspective, a drop out of just single scan rate forces ATC to reestablish contact with the aircraft. So a drop out just over 4.8 seconds at Fargo or just over 12 seconds at Finley puts additional strain on the ATC system. When aircraft are traveling several hundred miles an hour they can cover a great deal of distance in just one scan rate too. For example, an aircraft flying at 250mph would travel 0.33 miles in one radar scan at Fargo and 0.83 miles in one radar scan at Finley.

Altitude Discrepancies

Altitude Outliers

The first type of altitude discrepancy, an altitude outlier, happens when there is a significant deviation in the expected altitude for a given aircraft. A jump between data points is considered an outlier when the expected altitude is more than 800 feet off from nominal. Many times an outlier was accompanied by a drop out, where the data showed an outlier, and then a drop out occurred, or vice versa. Many of the outliers displayed an elevation of 0 feet above ground level (AGL) or over 120,00 feet. These readings were false because the aircraft was at an altitude above ground, and no commercial aircraft exists that can reach 120,000 feet. While altitude quantization and limitations on the barometric altimeter resolution are expected, large jumps observed are not nominal behaviors. The FAA requires altitude reported from Mode C to be quantized in 100 foot increments and be accurate within 2% of the actual value [49], so a properly functioning system shouldn't have more than a few hundred feet of variation from scan rate to scan rate. This data anomaly prevents ATC from providing accurate vertical separation between aircraft. An image showing altitude outliers can be seen in Figure 15 in Chapter III.

Prolonged Altitude Failure

The second type of altitude discrepancy is a prolonged altitude failure. This behavior occurs when there are a series of three or more zero altitude readings when the previous and/or past altitudes were greater than zero feet AGL. This failure is indicative of a transponder related issue, which will be expanded upon later in this paper. Again, this

type of behavior prevents ATC from providing proper vertical separation between aircraft in the NAS. Figure 16 depicts a plot of a prolonged altitude failure in Chapter III.

Repeated Data

Repeated data is defined as receiving multiple data points with near identical altitude and position within a fraction of a second. The radar site needs to receive and log two or more data points with a time period less than one second to qualify as this type of behavior. While this anomaly may not necessarily be harmful for aircraft or ATC, it is a type of behavior that was identified from this study, nonetheless. This behavior is shown in Figure 17 in Chapter III.

Multiple Aircraft with the Same Identification Number

The final data anomaly is multiple aircraft with the same identification number. As the name implies, there are several aircraft with equivalent Mode A transponder numbers which the pilot sets before takeoff. While some transponder ID's are non-discrete, meaning the same class of aircraft are all assigned the same number, such as VFR traffic, others are intended to be discrete. Even the discrete ID numbers have multiple aircraft occasionally. This problem stems back to only having 4096 possible combinations of numbers to assign to aircraft in the entire United States. Typically, there are more aircraft than that number in the sky at once. This data anomaly puts additional stress on ATC to determine the difference between two aircraft with the same ID. Figures 18 and 19 show examples of discrete and non-discrete multiple aircraft examples in Chapter III.

Main Script

The main script titled *Combined* calls all the individual functions and outputs the required information for analysis. The main script starts with three different for loops to

allow all the files to be processed at once. The for loop with the counter *ll* dictates whether the data is from Fargo or Finley. The for loop with the counter *ii* indicates which month the data is from; with 1 representing March, 2 representing June, 3 representing September, and lastly 4 representing December. Finally, the last for loop's counter *kk* indicates which day of the month the data is from, with 1 representing the first of the month, ranging all the way to 7 which represents the seventh of the month.

The rest of the code in the *Combined* file is contained within the three for loops. Three if statements are next in the code to direct the algorithm to the correct file path where the radar data spreadsheets are saved. Once the file path is determined the *Data_Read* function is called to read the data from the spreadsheet. After the data is read in and stored in a matrix, all the rows with missing data are eliminated. The missing data is from the secondary radar, namely the aircraft ID and altitude. Some aircraft were detected by just primary radar, while others were detected by both primary and secondary radar. Aircraft ID was the method to differentiate different aircraft for this study, so those rows of data with the missing aircraft ID/altitude had to be eliminated. Table 3 summarizes the average, maximum, and minimum row size for the original data, the updated data with rows eliminated that contained missing data, and the difference between the two. The average reduction in rows for Fargo was 73,628, accounting for an average of 33% of the original data. While Finley had an average reduction in rows of 117,324, approximately 25% of the original data.

Table 3. Data Removal Summary

	Fargo			Finley		
	Original Data	Updated Data	Difference	Original Data	Updated Data	Difference
Average	222,651	149,383	73,268	461,805	344,481	117,324
Maximum	393,112	225,151	186,210	749,859	459,712	357,503
Minimum	124,052	69,244	50,026	273,290	226,928	46,362

The next process is converting the azimuth and range to latitude and longitude. The azimuth is the angle in degrees between true north and where the target/aircraft is currently located. Range is the distance from the radar to the target/aircraft, given in nautical miles. An image showing these measurements is provided in Figure 6 [50]. The first step in converting azimuth and range to latitude and longitude is calculating the ground range, because the slant range is the range given in the data spreadsheets. This is calculated by Pythagoras Theorem,

$$a^2 + b^2 = c^2, \tag{3}$$

where a and b are the two legs of a right triangle, and c is the hypotenuse. In this case, the altitude is a , the ground range is b , and the slant range is c . The formula can be rearranged to solve for ground range,

$$b = \sqrt{c^2 - a^2}, \tag{4}$$

because the altitude and slant range are provided in the spreadsheet and already stored within the program. However, in some cases, when the aircraft gets too close to the radar, the altitude becomes larger than the slant range, this is shown in Figure 7; where GR is ground range, SR is slant range, and alt is altitude. When this happens, the number under

the radical in Equation 4 becomes negative, making the solution undefined. To correct this, new equations need to be used:

$$b' = \sqrt{a^2 - c^2}, \tag{5}$$

$$\beta = \sin^{-1} \frac{c}{a}, \tag{6}$$

and

$$b = b' + \sin \beta, \tag{7}$$

where β and b' are the angle and length, respectively given in Figure 13.

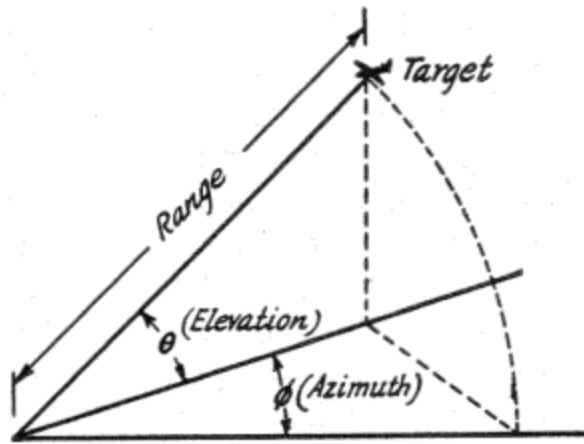
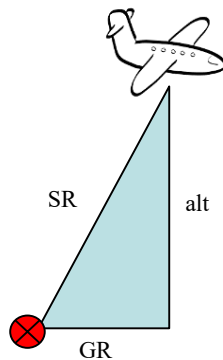


Figure 6. Azimuth and Range Diagram

Normal case



Close to Radar

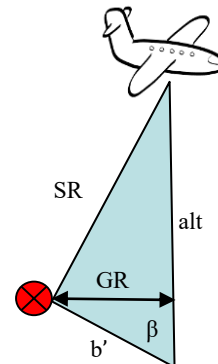


Figure 7. Ground Range, Slant Range, and Altitude of Aircraft

Once the ground range is calculated, the last step is to use latitude and longitude of the radar location, along with the azimuth and ground range to determine the latitude and longitude of the aircraft. This is given by Equation 8 which calculates the aircraft's latitude and Equation 9 which calculates the aircraft's longitude. These formulas calculate the latitude traveling along a great arc circle, knowing the start point, initial bearing (azimuth), and distance (ground range). Equations 8 and 9 are given by

$$lat_{aircraft} = \sin^{-1} \left(\sin(lat_{radar}) * \cos\left(\frac{b}{R}\right) + \cos(lat_{radar}) * \sin\left(\frac{b}{R}\right) * \cos(\emptyset) \right) \quad (8)$$

and

$$long_{aircraft} = long_{radar} + \tan^{-1} \left(\frac{\sin(\emptyset) * \sin\left(\frac{b}{R}\right) * \cos(lat_{radar}) - \sin(lat_{radar}) * \sin(lat_{aircraft})}{\cos\left(\frac{b}{R}\right)} \right), \quad (9)$$

where $lat_{aircraft}$ and lat_{radar} are the latitudes of the aircraft and radar, respectively. The variables $long_{aircraft}$ and $long_{radar}$ are the longitude of the aircraft and radar, respectively. R is the Earth's radius in nautical miles, b is the ground range in nautical miles, and lastly \emptyset is the azimuth.

This process to convert the azimuth and range to latitude and longitude is performed in a for loop, so the process is repeated for each row of data in the matrix stored within MATLAB. These locations are then added to the existing matrix in two new columns. There are two variants of the matrix that stores the time, aircraft ID, altitude, azimuth, range, latitude, and longitude. The first is titled *rd_sorted*, which has the data sorted by aircraft ID and within each ID number is chronological with respect to time. The second is a cell array, which has each aircraft data's array within a new cell, titled *rd_split*.

The remainder of the main script calls the various functions written and plots/saves the required data for analysis. Each type of data anomaly has its own function and after

each function is called, the locations of each occurrence are plotted on Google maps, along with the radar radius. A .png image and .fig MATLAB figure are saved for each map on each day, at each location for further analysis. At the very end of the script, all of the relevant data is output and saved for each day at each location. The data saved includes 32 different pieces of information, listed in Table 4. In addition to the Excel file saved, a .mat file was saved for each day at each location, so additional data could be referenced without running the algorithm again.

A flow chart summarizing the main script is shown in Figure 8. The algorithm starts and reads in the radar data with the data read function. Following that process, the missing data is eliminated and the position data is converted to latitude and longitude. The four data anomaly functions are then called which plot the location of the anomalies with open source software and the output from the four functions are also saved. Once this process is complete, the algorithm ends.

Table 4. Data Saved by the MATLAB Algorithm

Number	Description
1	Location (Fargo or Finley)
2	Month
3	Day
4	Number of aircraft detected by the radar
5	Number of aircraft that experienced drop outs
6-12	Summary of drop out instances in 7 different time intervals
13	Average drop out time length
14	Minimum drop out time length
15	Maximum drop out time length
16	Number of drop out instances
17	Number of aircraft that experienced altitude outliers
18	Number of altitude outlier instances
19-25	Summary of prolonged altitude failures
26	Average prolonged altitude failure time length
27	Minimum prolonged altitude failure time length
28	Maximum prolonged altitude failure time length
29	Number of prolonged altitude failure instances
30	Number of aircraft ID numbers that experienced multiple aircraft
31	Number of aircraft that experienced repeated data
32	Number of repeated data instances

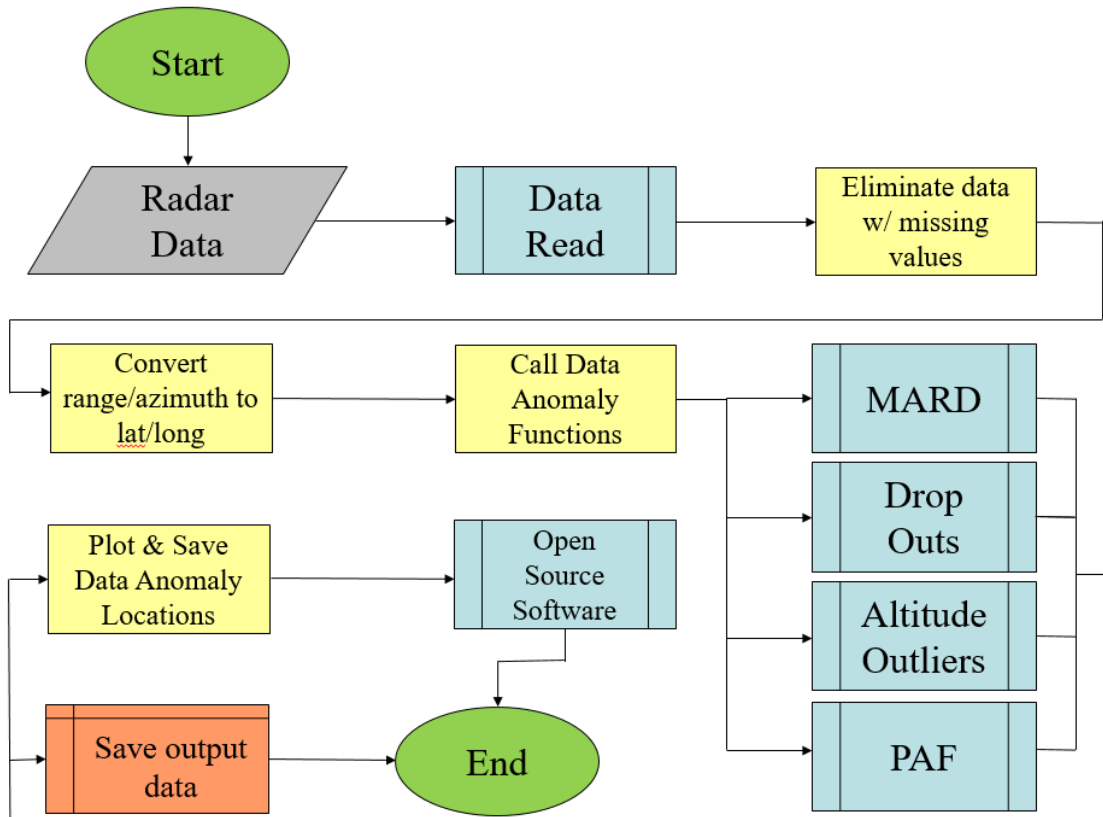


Figure 8. Main Script Flow Chart

Data Read

The first function that is utilized, reads in the radar data from the Excel spreadsheet, and is titled *Data_Read*. The input to this function is the filename of the spreadsheet. The output is a matrix called *rd*, which contains all the necessary radar data. Each piece of information is separated by column that is read into the program. The plane ID, time, altitude, azimuth, and range correspond to columns A through E, respectively. Once the data is read in, the function checks to ensure all the columns are the same length. If there is missing information in the last row of any of the columns, a value of Not-a-Number (NaN) is added to ensure all the columns are the same length. The five columns of information are then input into the matrix *rd*.

Multiple Aircraft and Repeated Data

The function that is called in the main script to identify the multiple aircraft and repeated data is titled *MARD* (short for Multiple Aircraft and Repeated Data). This function has the following inputs: the location identifier (1 for Fargo or 2 for Finley), the number of planes, *rd_split*, and *rd_sorted*. The outputs are the multiple aircraft data matrix, the repeated data matrix, and a matrix called *rd_wo_ma*. The matrix *rd_wo_ma* contains the radar data contained in *rd_split* and *rd_sorted*, without the multiple aircraft data. The data that was associated with multiple aircraft was taken out because it would skew the rest of the data anomaly identification and analysis.

The majority of the function is split into two if statements, based on the location. Within either if statement, the same fundamental logic remains the same, however the thresholds and parameters vary based on location. Once the input data is read in, the data points for each aircraft in *rd_split* are run through a for loop. In the loop, the time difference is taken, along with the difference in latitude and longitude from row to row in in each cell. Data that is under the time threshold based on the radar scan rate then either falls into the multiple aircraft category or repeated data category. The differentiator between the two is the latitude and longitude difference. If the positional difference is greater than some threshold, the data is considered a multiple aircraft. If the difference is less than that threshold, it is classified as repeated data. This logic is used because multiple aircraft with the same ID will have a large difference in position, while repeated data have almost identical positions. Once this process is complete, a new cell array titled *rd_wo_ma* is created that excludes any aircraft ID that corresponded to multiple aircraft. A table summarizing the thresholds is given in Table 5. These thresholds were selected in degrees

of latitude and longitude for convenience to plot the behaviors in the algorithm. Additionally, when converting latitude or longitude to a standard measurement such as nautical miles, there is very little variation for the area of interest in this study.

Table 5. Thresholds Used in the Multiple Aircraft and Repeated Data Function

Location	Time Threshold (seconds)	Latitude Threshold (degrees)	Longitude Threshold (degrees)
Fargo	3.8	.012	.035
Finley	10	.016	.16

The thresholds provided in Table 5 were experimentally determined from the data. Five different days chosen at random from Fargo, and five days chosen at random from Finley, with at least one day from each season was used to set the thresholds. An initial threshold was set, then using the 10 total days, the data was processed by hand checking for false positives, false negatives, true negatives, and true positives. A false positive is when the algorithm labeled data as a positive (multiple aircraft or repeated data), and it was not in fact a positive. False negative is when the function marked data as multiple aircraft or repeated data when it should not have been. A true negative is when the system correctly identified data that was not multiple aircraft or repeated data. Lastly, a true positive is when the program correctly marked the data that was multiple aircraft or repeated data. These values were adjusted until all the all the multiple aircraft and repeated data were correctly identified. The time threshold is slightly below the radar scan rate for both Fargo and Finley because the aircraft may be moving against the rotation of the radar, so it is detected sooner than the radar scan rate of the radar. Another interesting note is the latitude threshold between Fargo and Finley are relatively similar, while the longitude threshold is

significantly apart. This indicates that aircraft move more longitudinally in Finley, rather than latitudinally. Finley is a long range radar, picking up many aircraft in cross county routes, which tend to fly more east to west, rather than north to south. Additionally, Fargo is a regional radar, near an airport, so planes move more uniformly in both longitude and latitude on the approach paths to the Fargo International Airport. This explains why the latitude and longitude threshold at Fargo are closer to each other than the Finley thresholds. At the geographical location of either radar, the distance of one degree latitude is about 1/3 larger in nautical miles than one degree longitude, which explains why the longitude threshold is larger for both Fargo and Finley as well. A flowchart describing the process of the multiple aircraft/repeated data function is provided in Figure 9.

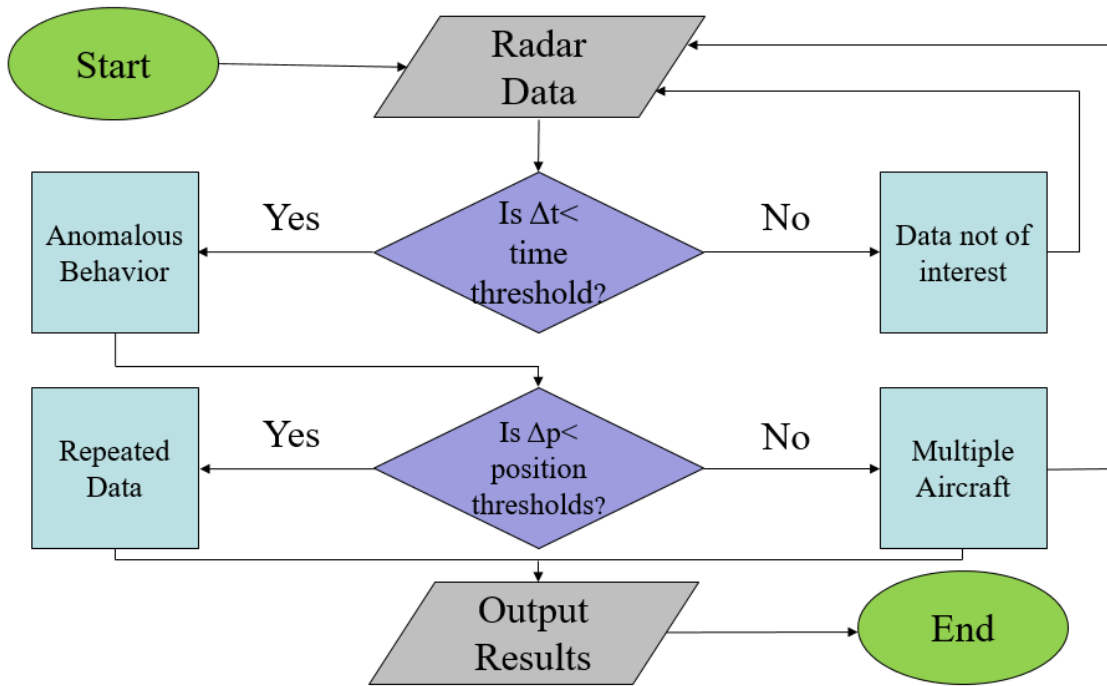


Figure 9. Multiple Aircraft/Repeated Data Function Flow Chart

Drop Outs

The drop out function, titled *DropOut*, is called in the main script to identify drop outs in the radar data. This function has inputs of the location identifier and *rd_wo_ma*. Its outputs include: all the data that has drop outs associated with them, a statistical summary of the drop outs, and a categorization of drop outs into time categories.

Once the data is input into the function, for each aircraft, it takes the time difference between each data entry. The function then looks at the upper and lower bounds of the time threshold to determine if it is a drop out. The lower bound is slightly above the scan rate, and the upper bound is several minutes. These thresholds vary by location, because of the different radar scan rates. A summary of the thresholds is given in Table 6. These thresholds again were selected based of setting a nominal value and modifying the threshold, until all the data from five different days at both the Fargo and Finley locations gave correct outputs.

Table 6. Drop Out Threshold Values

Location	Fargo	Finley
Lower Threshold (sec)	5.5	13
Upper Threshold (sec)	300	400

Once the drop outs are identified, statistics such as the average, minimum, and maximum values are taken. In addition, the drop outs are classified into seven different time intervals, to get an understanding of the time distribution of the drop outs. The number of drop outs are counted in each time interval. Due to the different radar scan rates, Fargo and Finley have different time intervals for the analysis. These time intervals are listed in Table 7.

Table 7. Time Interval Classifications for Fargo and Finley

Interval Number	Fargo	Finley
1	<10 sec	<24 sec
2	10-15 sec	24-36 sec
3	15-20 sec	36-48 sec
4	20-25 sec	48- 60 sec
5	25-30 sec	60-90 sec
6	30-60 sec	90-120 sec
7	>60 sec	>120 sec

A flow chart depicting the logic behind the drop out function is shown in Figure 10.

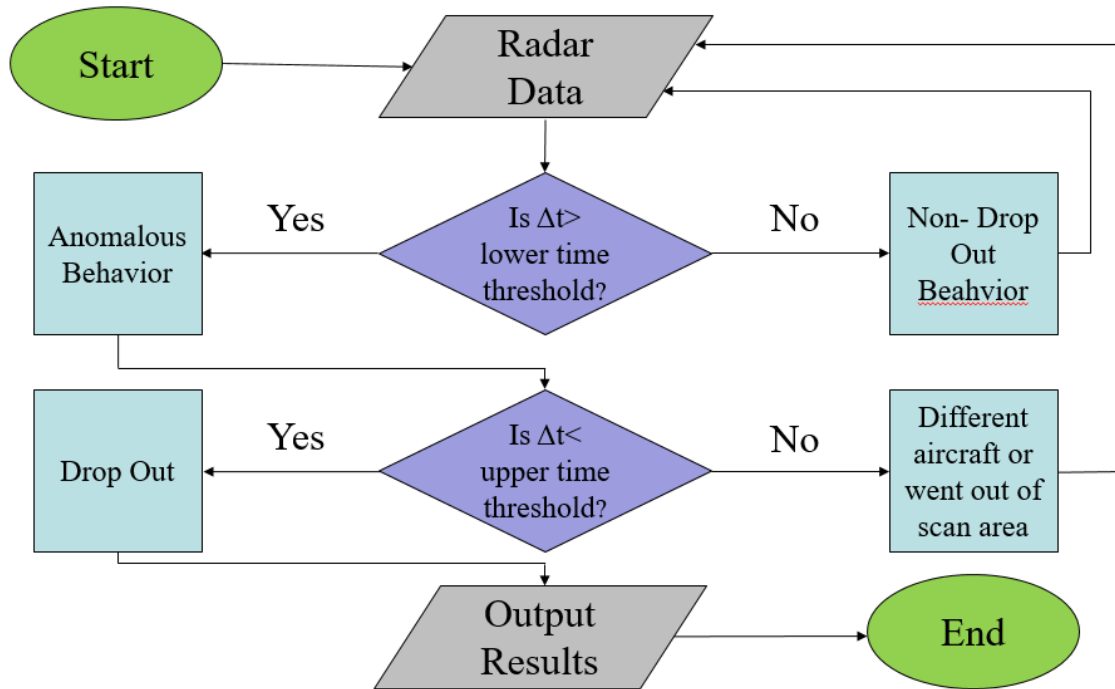


Figure 10. Drop Out Function Flow Chart

Altitude Outliers

The function, titled *Outliers*, identifies and classifies altitude outliers in the radar data. The inputs to this function are again, the location identifier and *rd_wo_ma*. The output is a matrix summarizing the outliers. After the radar data is input, the time difference between each data point and the altitude difference between each data point is taken. The time difference is used to ensure that the data being compared doesn't occur at different times of the day, or different aircraft (one aircraft lands and another takes off). If the altitude difference is greater than 800 feet, and the following altitude difference is greater 800 feet, this indicates an altitude outlier. The value of 800 feet was chosen to ensure an aircraft descending at a rapid rate were not misidentified as an aircraft with outliers. This was based on research of the maximum ascent/descent rate of the UND Cessna 172 training aircraft, a Bombardier CRJ700, and an Airbus A320. All these aircraft are likely to be detected by the radars at Fargo and Finley. The rate of climb of a Cessna 172 is 730fpm [51], 2,000fpm for the Bombardier CRJ700 [52], and 2,400fpm for the Airbus A320 [53]. Just considering the largest ascent/descent rate of 2,400fpm, the distance traveled vertically in one scan rate at Fargo is 200 feet and 480 feet at Finley. These values are much less than the 800-foot threshold chosen for this function. The threshold was also verified by processing several hours of data by hand, to ensure there were no false classifications. Once the outliers are detected, a matrix is produced containing the aircraft information, time, and location information. A flow chart highlighting the process of the altitude outlier function is given in Figure 11.

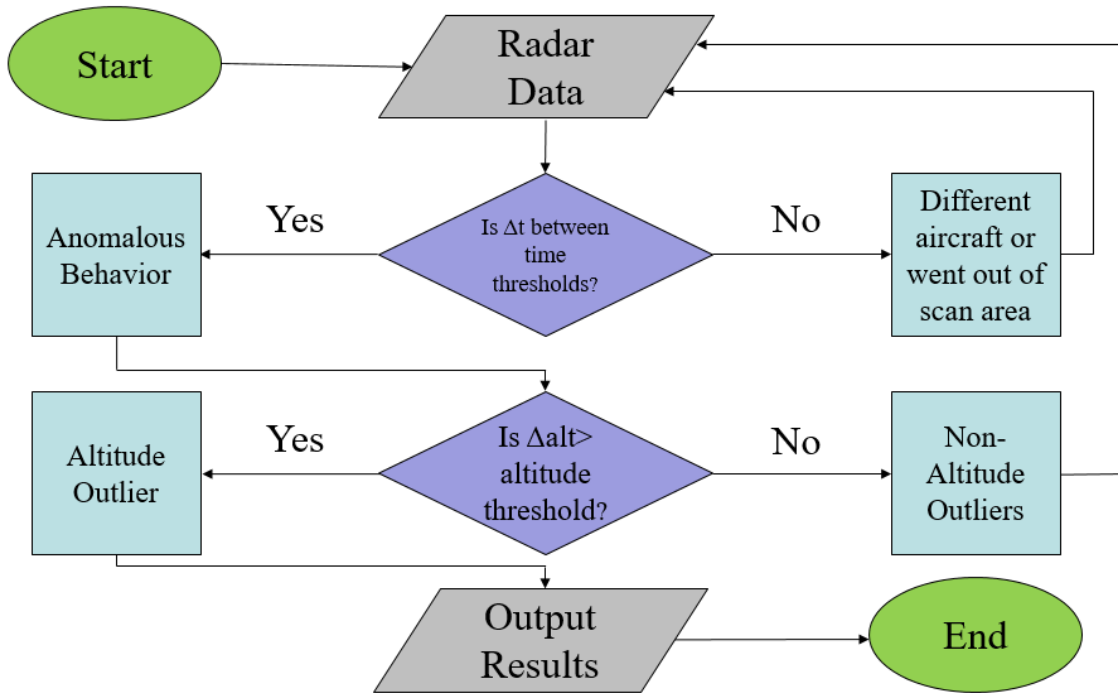


Figure 11. Altitude Outliers Function Flow Chart

Prolonged Altitude Failure

The function titled *prolonged_alt_failure*, processed the data to identify prolonged altitude failures. Just like many of the other functions, the inputs are the location identifier, along with *rd_wo_ma*. The outputs are matrices summarizing the prolonged altitude failures, along with their time durations. In addition, time statistics and time interval instances are output for analysis. The altitude information is looked at for a series of zero values. The function notes the start and stop indices of these series of zero values. Then only the strings of more than a length of two are kept, denoting a prolonged altitude failure. Both a cell array, separated by aircraft ID and a matrix with the time duration of the prolonged altitude failure are output. Again, just like the drop outs, the time statistics and time interval instances are output as well. These values are calculated the same exact way

as described in the previous sections. A flow chart highlighting the logic of the prolonged altitude failure function is shown in Figure 12.

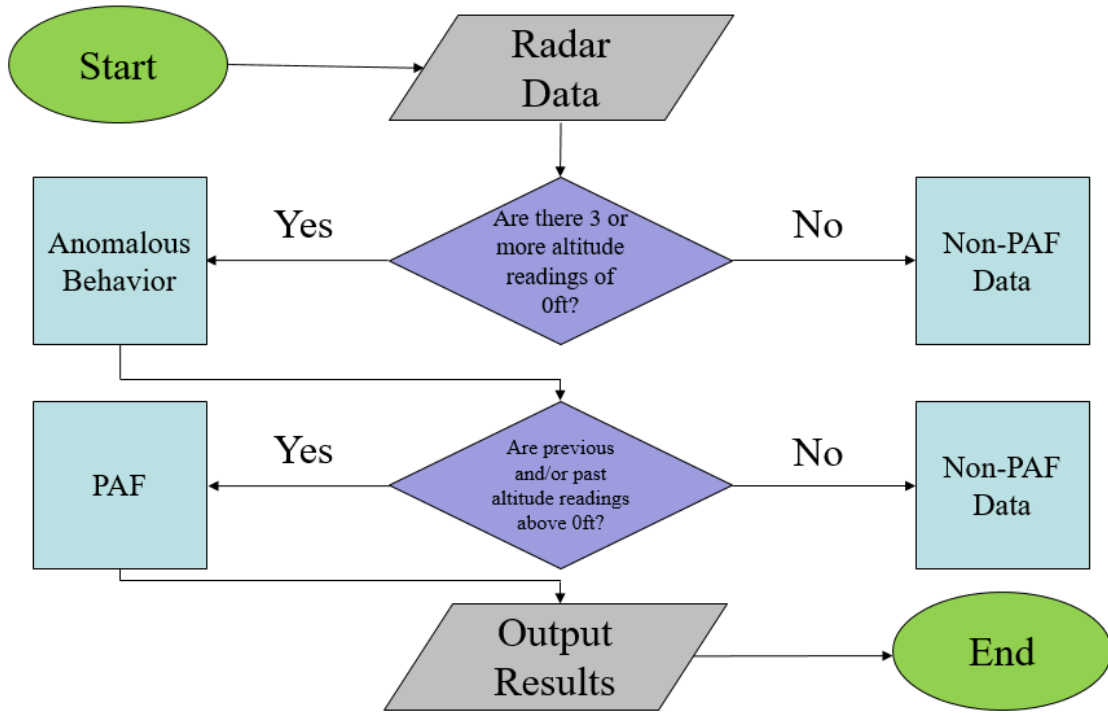


Figure 12. Prolonged Altitude Failure Function Flow Chart

Google Maps Plot & Draw Ellipse

The three functions used to plot the locations of the data anomaly locations and plot the effective radius on a Google Maps image were obtained online, through open source software. These functions were written for MATLAB. Due to these functions being open source and not authoring them myself, simply a broad description of how they are used will be provided in this section.

The function *plot_google_map*, allows a plot to be created with a static Google maps image to be inserted into the background [54]. The *plot* function within MATLAB is

used, plotting latitude and longitude, then the *plot_google_map* function is called to place a static Google maps image in the background. The image is automatically taken to encapsulate the latitude and longitude plotted. Many options can be varied on the Google maps image and output, such as the size, resolution of the image, and map type (roadmap, satellite, hybrid, etc.). Additional documentation can be found on the source site [54]. The *export_fig* function is also used to save the Google Map output [55].

To draw the effective radar radius, an ellipse needs to be drawn. The reason an ellipse is used, even though it appears as a circle on the map is due to the measurement differences in nautical miles of the one degree of latitude and one degree of longitude. There was no built in ellipse function into MATLAB, so an open source function titled *ellipse*, was utilized [56]. The inputs to the function include: the semi-major axis length, the semi-minor axis length, angle of the semi-major axis, center point, color of the drawn ellipse, and the number of points used. Table 8 summarizes the inputs into the ellipse function. The semi-major and semi-minor axis lengths were calculated based on the conversion factor between nautical miles to one degree of latitude or longitude at the radar's location. At Fargo, one degree of latitude and longitude is 60.0266nmi and 41.1279nmi, respectively. At Finley, one degree of latitude and longitude is 60.03287nmi and 40.66041nmi, respectively. To calculate the semi-major and semi-minor axis length, the radar range in nautical miles is simply divided by the conversion factor for one degree of latitude/longitude at that location. While the measurement will vary slightly, once a location deviates from the radar's location, it is minimal for the distances considered in this study. So, the conversion factor at either Fargo or Finley is simply used for the entire radar's range.

Table 8. Ellipse Function Input Values

Description	Fargo	Finley
Semi-Major Axis Length	$\frac{60}{41.1279}$	$\frac{250}{40.6604}$
Semi-Minor Axis Length	$\frac{60}{60.0266}$	$\frac{250}{60.0329}$
Semi-Major Axis Angle	0 degrees	0 degrees
Initial Longitude location	46.9202 degrees	47.5282 degrees
Initial Latitude Location	96.8122 degrees	97.9006 degrees

Overlapping Analysis

The script titled *Overlapping_Analysis* is a standalone algorithm that performs an overlapping analysis to understand whether the data anomalies were radar or transponder induced. If the data anomalies occurred at one location, it was considered a radar error, while if the data anomaly occurred at both locations it was considered a transponder error. This analysis will be described in more detail later in Chapter IV. The script reads in the .mat files that were created from the main script *Combined* for each of the Fargo and Finley locations. The script also has a nested for loop to process all the data at once from all 28 days. Once the .mat file is read in, the specific matrix associated to the data anomaly of interest is called. The program then looks for aircraft ID numbers that occur in both the Finley and Fargo data. Once those numbers are recorded, the timestamps associated with each location are analyzed. If the time difference was 17 seconds or less it was considered to occur at both the Fargo and Finley location (transponder error). The value of 17 seconds was obtained by adding the two scan rates together of both locations. This value was chosen because the aircraft could have just been missed by one radar and not be picked up by the

other radar until the end of its scan rate. The data points that did not occur at both sites are then considered radar failures. The script outputs the number of aircraft intercepted at each radar site, the number of transponder failures, and the percentage of aircraft with transponder or radar failures. This process is repeated for each data anomaly: drop outs, altitude outliers, prolonged altitude failures, repeated data, and multiple aircraft with the same ID. To summarize the logic of the overlapping analysis algorithm, a flow chart is provided in Figure 13.

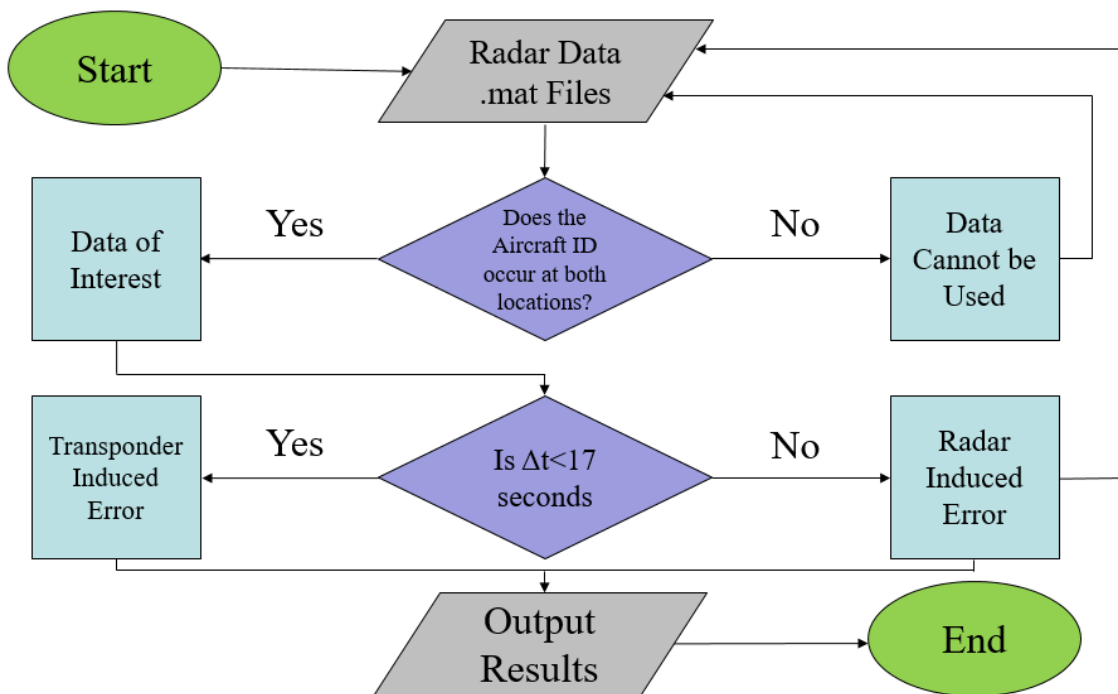


Figure 13. Overlapping Analysis Algorithm Flow Chart

Algorithm Verification

Once the entire algorithm was finished, it was checked against data that was analyzed by hand from the previous June 21, 2015 study. In addition, the data from March 1, 2015 was also analyzed for the purposes for algorithm verification. The algorithm found

all of the data anomalies identified in both those studies, along with several other randomly selected groups of time throughout the other data from March, June, September, and December. In fact, the program was able to find data anomalies that were missed by the hand analysis, showing how significant human error can be in large data studies such as this. Once the algorithm was completed, it took seven hours, 18 minutes to process all the data from four weeks of data at both Fargo and Finley.

CHAPTER III

SUMMARY OF DATA ANOMALY RESULTS

Once the radar data was processed, the engineering and scientific analysis of the data could begin. By using an algorithm, the large, daunting amount of data became more reasonable to analyze and understand. This chapter will highlight a basic summary of the results, including graphical depictions of the anomalies, the number of each type of data anomaly, time durations, and locations of the anomalies. This summary is provided for each location, and is an average of the data for brevity. All the data for each day analyzed is provided in Appendix C: Radar Data Anomalies Summary. There are a number of sections in Appendix C: Aircraft Count/Location, Drop Outs, Altitude Outliers, Prolonged Altitude Failures, and Multiple Aircraft/Repeated Data. Each data anomaly was plotted on an altitude versus time plot to provide a visual representation in the Graphical Representation of Data Anomalies section. These plots were chosen to provide what a typical case for each anomaly looks like. The Aircraft Count/Location section summarizes the number of aircraft detected each day, at each location. The Drop Outs section provides a daily summary of the number of aircraft with drop outs, the number of instances of drop outs that occur in various time durations, the average drop out length, the minimum drop out length, the maximum drop out length, and the number of instances of drop outs. The Altitude Outliers section summarizes the number of aircraft that experienced outliers each day, along with the number of instances of outliers. The Prolonged Altitude Failure

section provides a nearly identical summary when compared to the Drop Out section. Lastly, the Multiple Aircraft/Repeated Data section provides aircraft counts for each day for each anomaly. A more thorough discussion of the results will be provided in Chapter IV.

Graphical Representation of Data Anomalies

Each data anomaly was plotted on an altitude versus time plot to provide a better understanding of what the behavior truly looked like. An example of a drop out is shown in Figure 14, which plots altitude versus time for a single aircraft. Notice how there are both large and small gaps where there are no data points, this indicates a drop out. Figure 15 depicts several outliers, going between the aircraft's cruising altitude of 900/1000 feet AGL and 0 feet AGL. An example of a prolonged altitude failure is shown in Figure 16, which has a zero-foot altitude reading for 17 radar scan rates, then the altitude returns to nominal. Repeated data is shown in Figure 17. The red boxes are zoomed areas of the data points to show a close up of what is happening. In this example, there are four instances of repeated data, each with two data points within a fraction of a second. Figure 18 shows a multiple aircraft example plotting altitude versus time. The non-discrete ID number of 1200 is shown in this example which is assigned to VFR traffic in the airspace by ATC. Figure 18 shows many aircraft making the plot very congested, this is because many of the aircraft in the sky are VFR traffic, which are all assigned the same ID number. Figure 19 shows two aircraft with a discrete code of 2603 in the same airspace. The ID is discrete in Figure 19, meaning that ideally there would only be one aircraft that was assigned that number. However, in this case the number was used twice.

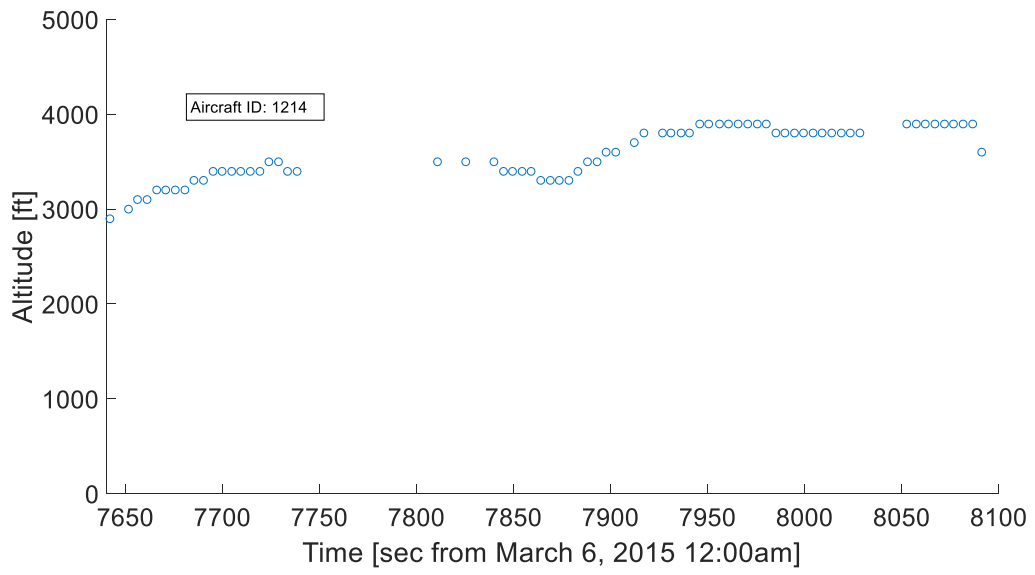


Figure 14. Drop Out Example Plot

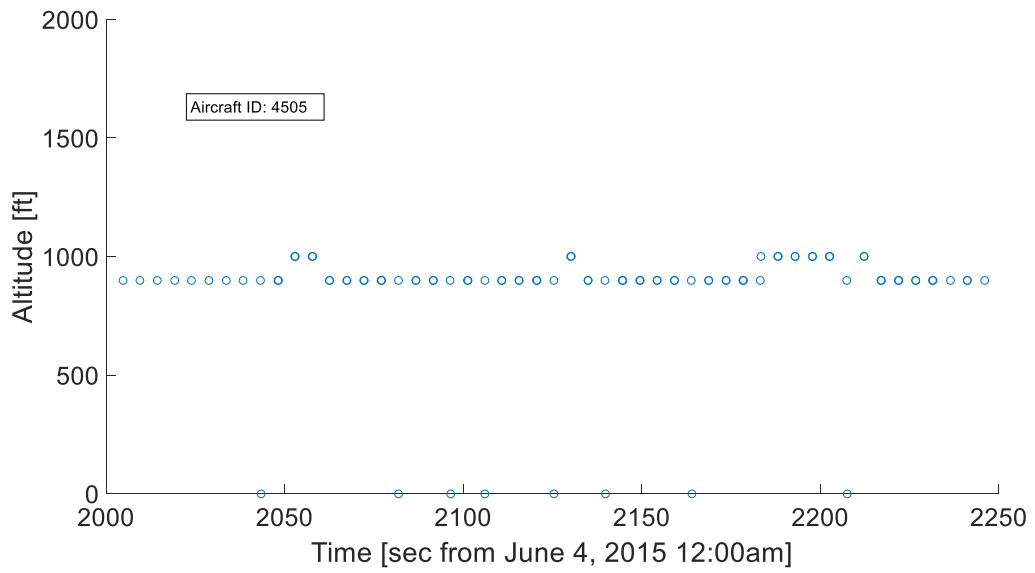


Figure 15. Outlier Example Plot

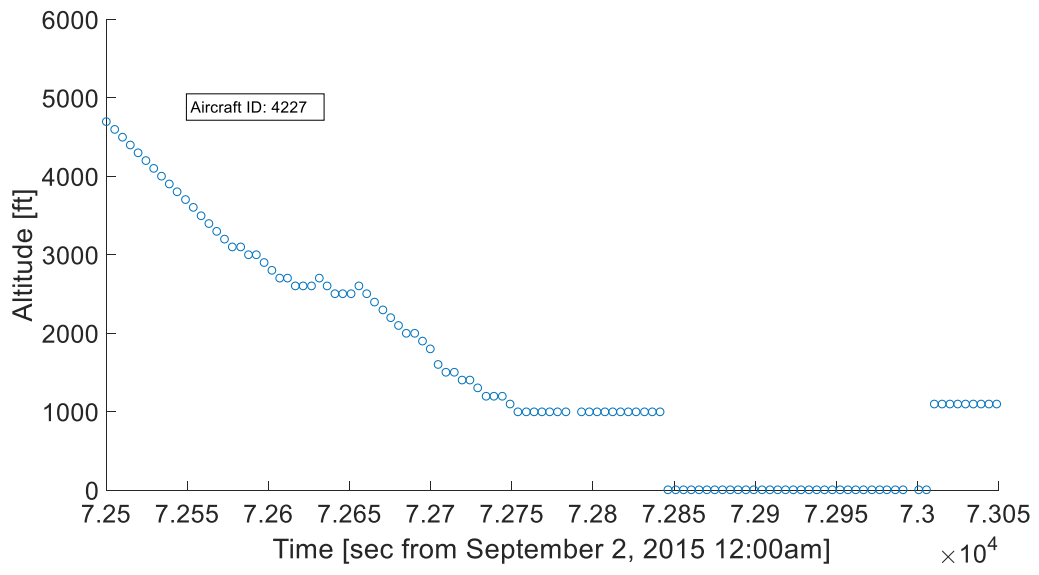


Figure 16. Prolonged Altitude Failure Example Plot

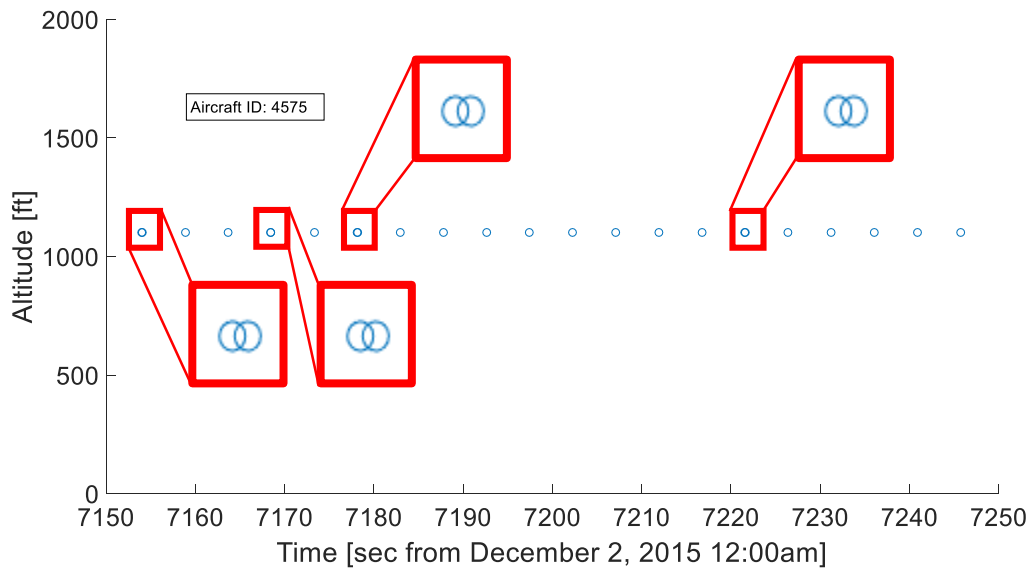


Figure 17. Repeated Data Example Plot

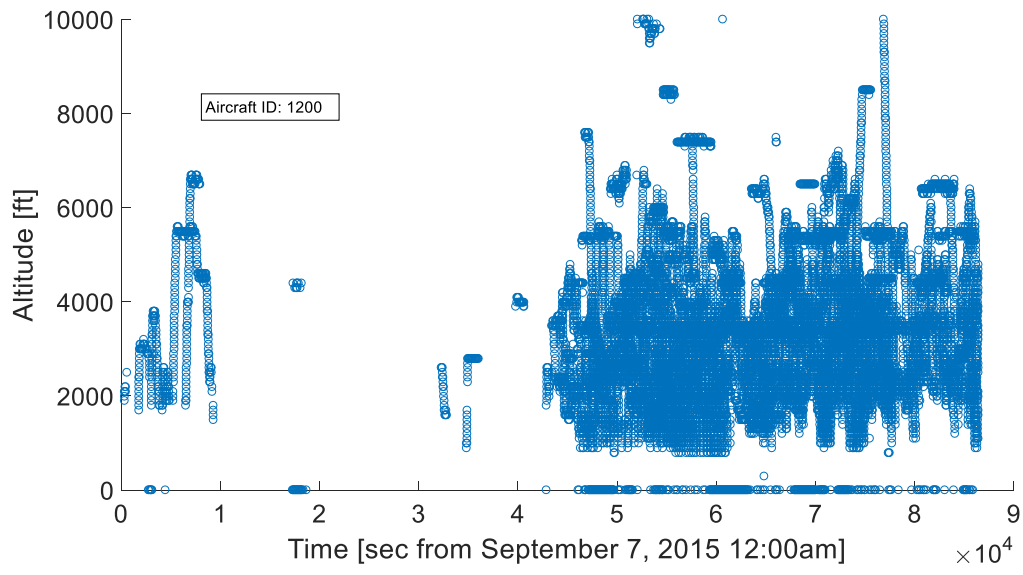


Figure 18. Non-Discrete Multiple Aircraft Example Plot

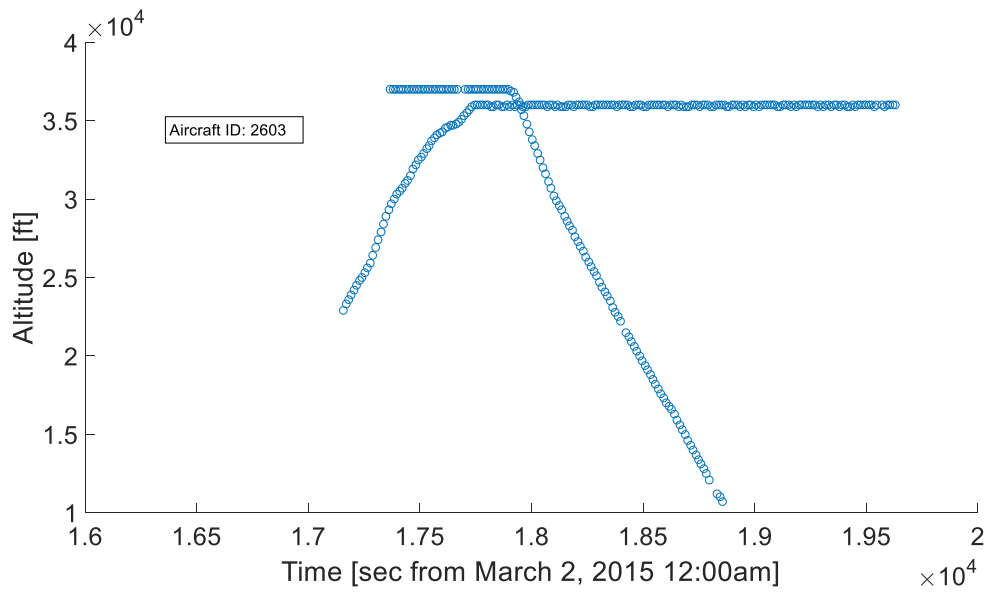


Figure 19. Discrete Multiple Aircraft Example Plot

Aircraft Count & Time Duration Summary

After the data was processed through the algorithm, the aircraft counts were analyzed, along with the time duration for drop outs and prolonged altitude failures. Table 9 summarizes the total number of aircraft observed each day at Fargo and Finley, along with the number of aircraft that experienced each type of anomaly. The number of aircraft is on the left column for each location, while the percentage of total aircraft effected is on the right hand column. The numbers in upper most section of Table 9 represent the average number over the four weeks of data analyzed, while each subsequent section is an average of each specific week. This provides a clear picture of the overall status of the radar network because the averages encapsulate four distinct different times of year over a wide variety of weather conditions.

When comparing the averaged data from each separate week to the to the overall averaged data, the results were relatively similar. It should be noted that more aircraft were intercepted in the fair weather months of June and September. While fewer aircraft flew in March and December when the weather typically is not as nice for flight. There also seemed to be a significantly larger number of drop outs in September compared to the nominal average percentage of drop outs for Fargo (37%) and especially Finley (84%). The opposite trend was observed in March for the percentage of drop outs at Finley with a significant decrease in drop outs, 52%, versus the overall average of 69%. There were not any noticeable trends for the other data anomalies from month to month. It is noteworthy that overall, 34% of aircraft experienced drop outs at Fargo, while 69% of aircraft experienced drop outs at Finley. Additionally, the other data anomalies ranged from 1-26% of aircraft

at either location. Both altitude outliers and drop outs were the two types of data anomalies that happened most frequently.

To understand the severity of the drop outs and prolonged altitude failures, the time duration was looked at. The average, minimum, and maximum values of occurrence were identified for each location. The anomalies were then counted in different time intervals that were set based on the radar scan rate at each location. Table 10 provides this summary of the drop out times and Table 11 provides the summary of the PAF times. Again, these tables are the average values over the course of the four weeks of data. Just the overall averages for the time durations were chosen to be displayed in the main body of this document. This was decided, because from Table 9, there was not any drastic change when comparing the weekly averages to the overall averages. All required data to do a weekly average analysis is provided in Appendix C. In the time interval classification section for each table, the left column represents the instances (there can be multiple instances per aircraft), and the right column is the percentage of instances relative to the total number of instances.

Table 9. Aircraft Count Summary for Fargo and Finley

		Fargo		Finley	
		Number of Aircraft	Percentage	Number of Aircraft	Percentage
Overall	Total Observed	450	-	1274	-
	Drop Outs	155	34%	885	69%
	Outliers	61	14%	326	26%
	PAF	7	2%	9	1%
	Repeated Data	11	2%	45	4%
	Multiple Aircraft	14	3%	11	1%
March	Total Observed	433	-	1150	-
	Drop Outs	147	34%	602	52%
	Outliers	60	14%	190	17%
	PAF	6	1%	7	1%
	Repeated Data	11	3%	41	4%
	Multiple Aircraft	6	1%	8	1%
June	Total Observed	459	-	1328	-
	Drop Outs	159	35%	936	70%
	Outliers	61	13%	304	23%
	PAF	5	1%	6	1%
	Repeated Data	11	2%	35	3%
	Multiple Aircraft	16	4%	10	1%
September	Total Observed	440	-	1332	-
	Drop Outs	161	37%	1116	84%
	Outliers	55	12%	442	33%
	PAF	6	1%	6	1%
	Repeated Data	6	1%	59	4%
	Multiple Aircraft	15	3%	12	1%
December	Total Observed	467	-	1287	-
	Drop Outs	153	33%	888	69%
	Outliers	69	15%	366	28%
	PAF	9	2%	16	1%
	Repeated Data	14	3%	44	3%
	Multiple Aircraft	18	4%	15	1%

Table 10. Drop Out Summary for Fargo and Finley

Fargo			Finley		
Drop Out Duration			Drop Out Duration		
Average (sec)	23.58	-	Average (sec)	42.45	-
Minimum (sec)	6.81	-	Minimum (sec)	17.17	-
Maximum (sec)	282.46	-	Maximum (sec)	391.15	-
Time Interval Classification			Time Interval Classification		
Number of Drop Outs	552	-	Number of Drop Outs	3279	-
Less than 10 sec	326	59%	Less than 24 sec	1105	34%
10 -15 sec	81	15%	24 -36 sec	1302	40%
15 -20 sec	10	2%	36-48 sec	296	9%
20 -25 sec	21	4%	48 - 60 sec	57	2%
25 -30 sec	16	3%	60 -90 sec	232	7%
30 -60 sec	52	9%	90 -120 sec	92	3%
Greater than 60 sec	47	9%	Greater than 120 sec	195	6%

Table 11. Prolonged Altitude Failure Summary (PAF) for Fargo and Finley

Fargo			Finley		
PAF Duration			PAF Duration		
Average (sec)	124.99	-	Average (sec)	170.15	-
Minimum (sec)	16.53	-	Minimum (sec)	39.40	-
Maximum (sec)	426.83	-	Maximum (sec)	495.42	-
Time Interval Classification			Time Interval Classification		
Number of PAF	6.5	-	Number of PAF	9.0	-
Less than 10 sec	0.8	13%	Less than 24 sec	1.0	11%
10 -15 sec	0.3	5%	24 -36 sec	1.3	15%
15 -20 sec	0.1	2%	36-48 sec	0.6	7%
20 -25 sec	0.3	4%	48 - 60 sec	0.8	9%
25 -30 sec	0.3	4%	60 -90 sec	1.0	11%
30 -60 sec	1.1	17%	90 -120 sec	0.7	7%
Greater than 60 sec	3.6	55%	Greater than 120 sec	3.6	40%

There was an average drop out of 23.58sec at Fargo and 42.45sec at Finley. While 74% drop outs occurred for less than three radar scan rates at both Fargo and Finley; this is still a significant period of time to not have ATC receive information on the aircraft. It should also be noted that if ATC loses an aircraft for even one radar scan rate, they need to reestablish contact with them, so even having a short drop out adds more strain to the current ATC system. The maximum values show the drop outs can last for several minutes in some cases. Additionally, the average PAF was 124.99sec at Fargo and 170.15sec at

Finley. This is a substantial amount of time for ATC to not have a correct altitude reading to help with vertical separation services. It is also noteworthy that 55% of PAF at Fargo lasted for more than one minute, while 58% of prolonged altitude failures at Finley occurred for greater than one minute.

Time of Day & Location Summary

Time of day and location were also investigated to understand if either factor impacted the drop outs and data anomalies. Other than the fact that more aircraft are flying during daylight hours, time of day not play a large role in the occurrence of the drop outs and data anomalies. However, despite that fact, location did appear to have an impact on the unique behavior observed. Figure 20 gives a typical plot of drop out locations at Fargo and Finley. Figure 21 provides a usual altitude outlier location plot for both locations. Figure 22 does the same for prolonged altitude failure, Figure 23 for repeated data, Figure 24 for multiple aircraft with the same discrete ID, and Figure 25 for multiple aircraft with the same non-discrete ID.

From Figures 20 and 21, it can be seen that majority of all outliers occur on the approach paths/near airports and on the edge of the effective radar radius. The prolonged altitude failure shown in Figure 22 appears to be a transponder related issue, with long trails showing the path of the aircraft. While Figure 23 show most repeated data occurs near airports as well. Figure 24 also shows that discrete multiple aircraft ID failures are transponder/aircraft specific. This is logical because ATC assigns each aircraft an ID before takeoff. However, Figure 25 shows that non-discrete ID failures are more sporadic. It is obvious there is a large concentration near airports and additionally transponder/aircraft flight paths can be seen. So this failure also appears to be transponder specific. The

behaviors that occurred near the edge of the radar effective radius occur because the radar is being pushed to its operational capacity to detect those aircraft. Additionally, airports are high traffic areas. As the airspace gets more congested, the radar is more likely to make mistakes. Based off Table 9 and Figures 20-25, drop outs and outliers are the most prominent unique behavior observed.

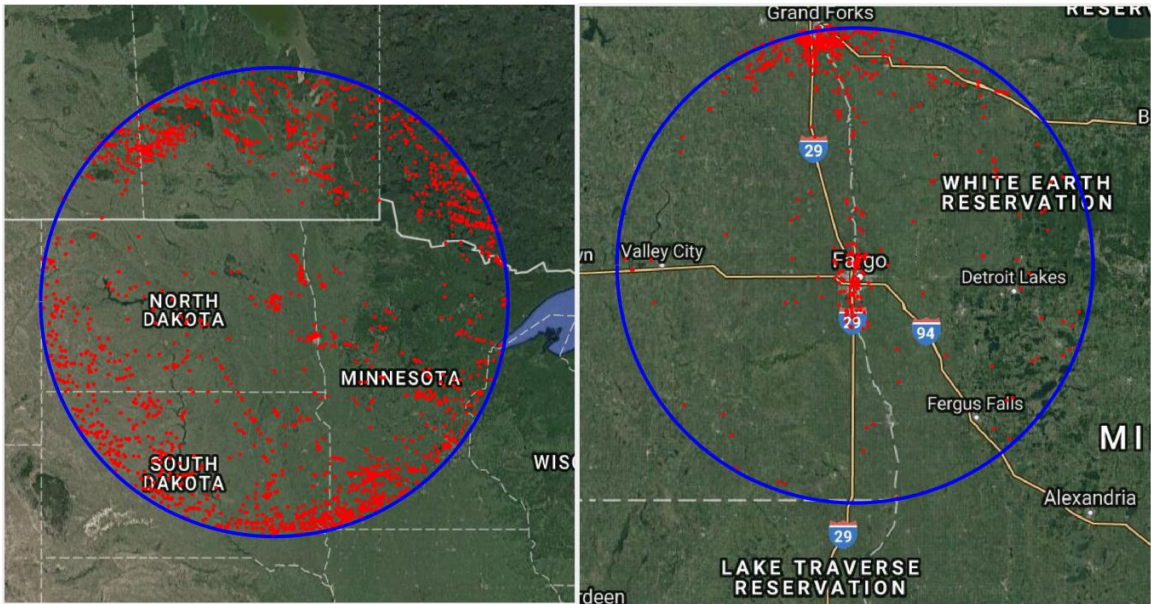


Figure 20. Drop Out Location Plots for Finley (left) and Fargo (right)

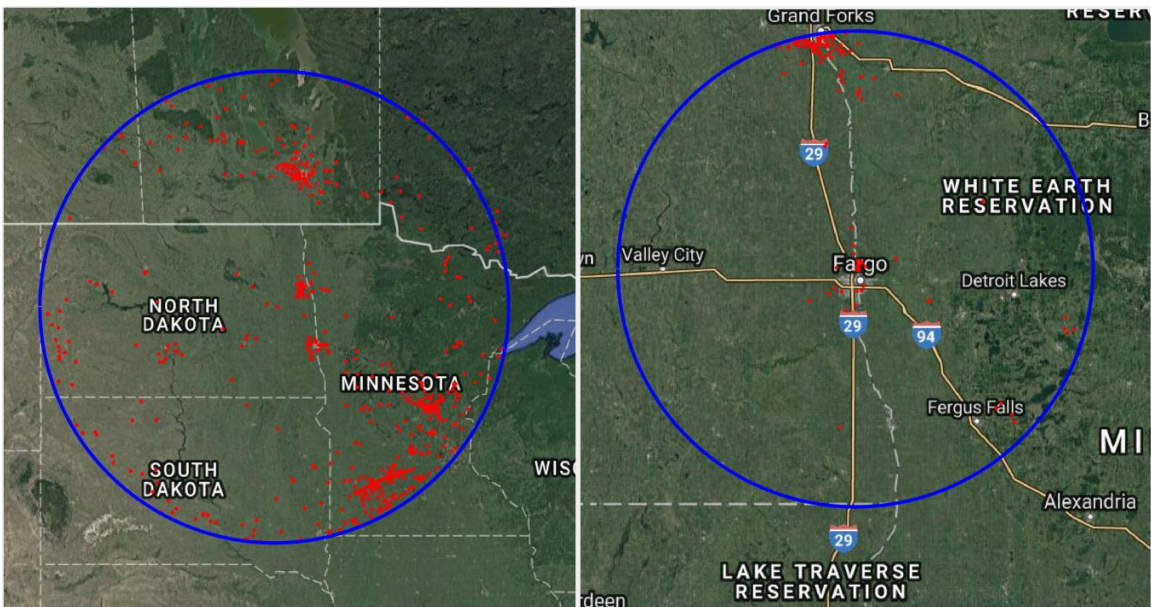


Figure 21. Altitude Outlier Location Plots for Finley (left) and Fargo (right)

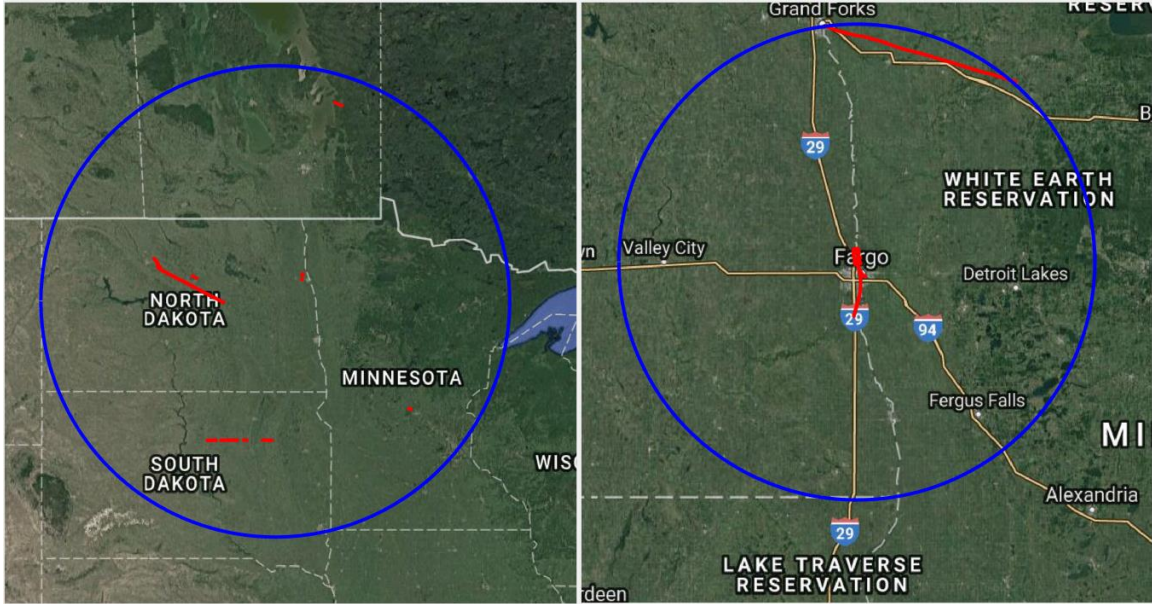


Figure 22. Prolonged Altitude Failure Plots for Finley (left) and Fargo (right)

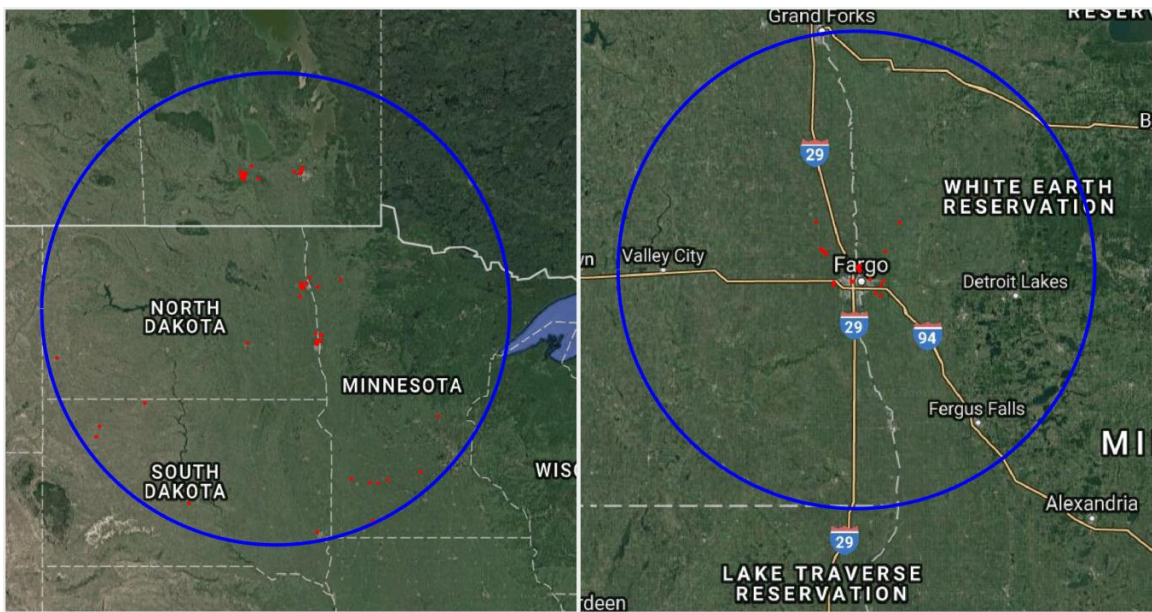


Figure 23. Repeated Data Location Plots for Finley (left) and Fargo (right)

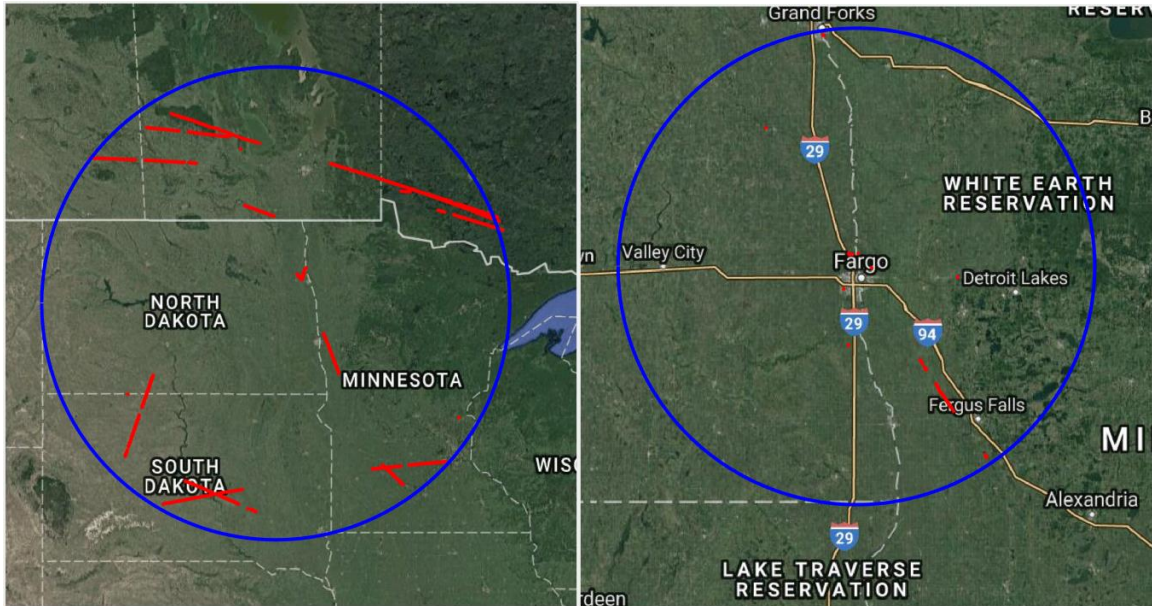


Figure 24. Multiple Aircraft with the Same Discrete ID Location Plots for Finley (left) and Fargo (right)

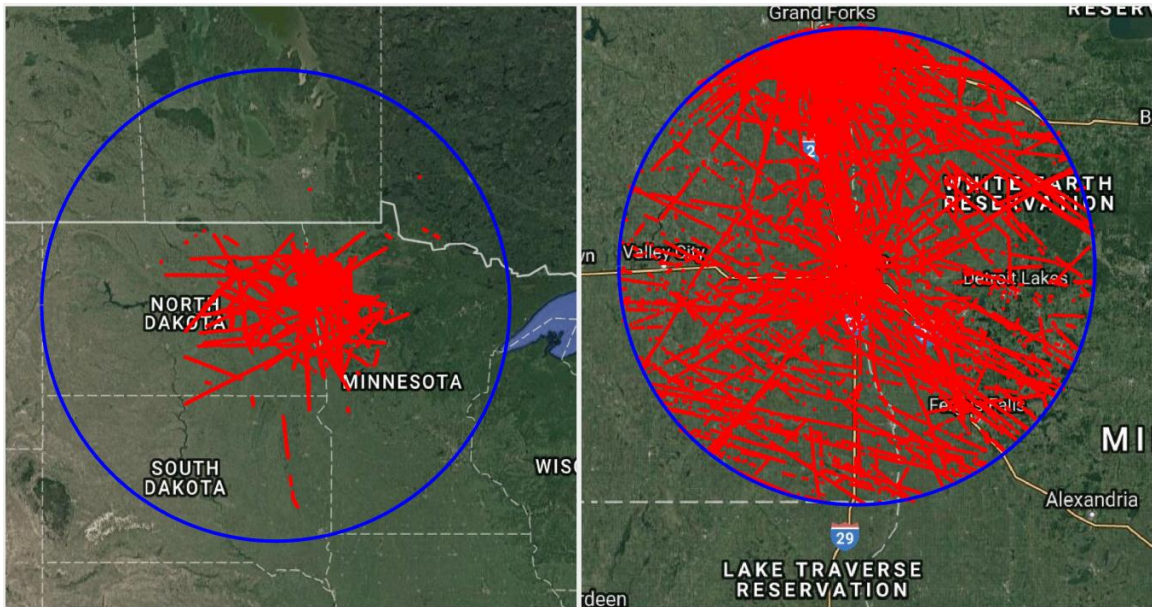


Figure 25. Multiple Aircraft with the Same non-Discrete ID Location Plots for Finley (left) and Fargo (right)

CHAPTER IV

DISCUSSION OF RESULTS

While Chapter III provided a summary of the results based on aircraft count, time duration, and location. This chapter will provide a more through explanation and conjecture for why some of these anomalies are occurring. The overlapping analysis mentioned earlier will be presented in this chapter, along with a correlation of weather/climate conditions on each day of this study. The final sections will cover a risk matrix/hazard assessment to understand the true severity of data anomalies, and lastly a comparison to previous studies will be assessed.

Overlapping Analysis

To understand if the data anomalies were radar or transponder induced errors, an overlapping analysis was performed. Only the aircraft that were intercepted at both Fargo and Finley were considered. Fargo's radar radius fits entirely inside Finley's radar radius, as shown by Figure 4, so theoretically all of the aircraft detected by Fargo should be detected by Finley. The two primary exceptions to that rule are if the aircraft were detected by Finley, but in Fargo's radar cone of silence, or the aircraft were too low to be intercepted by Finley's radar beam. If the data anomalies occurred at both sites, it was considered a transponder error, because it is highly unlikely both radar sites were experiencing failures at the same time. If the data anomalies occurred only at one site, it was considered a radar error. Only the drop outs and altitude outliers produced statistically significant results for this analysis, so only those behaviors were considered. Table 12 summarizes the percentage

of aircraft that experienced transponder induced errors and radar induced errors for drop outs and outliers. These percentages consider the average value of over the time period listed in the far left hand column of the table. The overall average, and average for each week of data was considered. For a complete day by day breakdown, a more detailed table can be seen in Appendix D: Overlapping Analysis Results.

Table 12. Radar and Transponder Failure Summary

	Data Anomaly Description	Percentage of Aircraft w/ Radar Error	Percentage of Aircraft w/ Transponder. Error
Overall	Drop Outs	93%	7%
	Outliers	51%	49%
Mar.	Drop Outs	94%	6%
	Outliers	49%	51%
June	Drop Outs	95%	5%
	Outliers	52%	48%
Sep.	Drop Outs	93%	7%
	Outliers	53%	48%
Dec.	Drop Outs	91%	9%
	Outliers	50%	50%

Table 12 shows there isn't a drastic change from season to season when compared to the overall results for the method of failure. In this study, the overall results showed 93% of drop outs were radar errors, while only 51% of altitude outliers experienced radar errors. The largest deviation from these values were by $\pm 2\%$ considering the weekly averages.

The most probable explanation for why outliers have roughly equal percentages for the source of error, between the radar and transponder, while drop outs are mostly caused by the radar is the number of systems that can contribute to the data anomalies. Outliers have many more systems involved than drop outs. The altimeter, system to convert the altitude reading to binary for the transponder, transponder, and all the systems at the ground station can all contribute to altitude outliers. The increase in sources of error, make the distribution more uniform between radar or transponder induced errors for the altitude outliers.

It should be noted that prolonged altitude failures and multiple aircraft with the same ID are also transponder induced errors, which can be seen from Figures 22, 24 and 25. However, there were not enough data points of those anomalies to be considered for the overlapping analysis.

Hazard Assessment

To assess risk, the FAA and other organizations use Safety Risk Management (SRM), which is a process to analyze, assess, and accept risk for designs, policies, and many other aspects. One tool that is used in SRM is a risk matrix to help quantify the amount of risk. The risk matrix takes into account the severity and likelihood of an event, then using the combination of both interactions, assigns a rating in terms of risk: unacceptable risk, acceptable risk with mitigation, and acceptable risk. Table 13 provides the FAA severity definitions [57] and Table 14 provides the FAA likelihood definitions [57]. Due to the broad nature of the likelihood definitions, a revised table, specific for this study was developed; this is shown in Table 15. Table 16 provides the generic FAA risk matrix [57].

Table 13. FAA Severity Definitions

Minimal 5	Minor 4	Major 3	Hazardous 2	Catastrophic 1
Negligible safety effect	<ul style="list-style-type: none"> - Physical discomfort to persons - Slight damage to aircraft/vehicle 	<ul style="list-style-type: none"> - Physical distress or injuries to persons - Substantial damage to aircraft/vehicle 	Multiple serious injuries; fatal injury to a relatively small number of persons (one or two); or a hull loss without fatalities	Multiple fatalities (or fatality to all on board) usually with the loss of aircraft/vehicle
Courtesy: Federal Aviation Administration [57].				

Table 14. FAA Likelihood Definitions

Frequent A	Expected to occur routinely
Probable B	Expected to occur often
Remote C	Expected to occur infrequently
Extremely Remote D	Expected to occur rarely
Extremely Improbable E	So unlikely that it is not expected to occur, but it is not impossible
Courtesy: Federal Aviation Administration [57].	

Table 15. Radar Data Anomaly Likelihood Definitions

Frequent (A)	Probable (B)	Remote (C)	Extremely Remote (D)	Extremely Improbable (E)
Occur on 40%+ of aircraft each day	Occur on 10-40% of aircraft each day	Occur on 1-10% aircraft each day	Occur on 0.05-1% of aircraft each day	Occur on less than 0.05% of aircraft each day

Table 16. Generic FAA Risk Matrix

Severity \ Likelihood	Minimal 5	Minor 4	Major 3	Hazardous 2	Catastrophic 1
Frequent A					
Probable B				[Red]	
Remote C			[Yellow]		
Extremely Remote D		[Green]			
Extremely Improbable E					*

Unacceptable Risk
Acceptable Risk with Mitigation
Acceptable Risk

* Unacceptable with Single Point and/or Common Cause Failures

Courtesy: Federal Aviation Administration [57].

In the risk assessment used for the radar anomalies, the ratings were assigned under the assumption that no other DAA system were in place, including the pilot’s ability to see and avoid. The risk assessment performed assumes only the radar system is in place, providing the five pieces of information run through the algorithm: time, aircraft ID, altitude, range, and azimuth. A study incorporating other DAA systems is beyond the scope of this project. By performing the risk analysis under these underlying assumptions, a true sense of the severity of these issues can be understood. Using the likelihoods assigned in Table 15 and severity ratings from Table 13, a risk matrix was assembled for the data anomalies. The severity was taken for each data anomaly to assume the worst case scenario,

given that condition. It should be noted that typically the severity at Finley was higher than Fargo for the data anomalies studied for the risk matrices. This is because the scan rate is larger and the radar covers a larger area, so there is an increase in the potential severity of events, if they do happen. An overall risk matrix, looking at the averages from the four weeks of data is shown in Table 17. Tables 18-21 show the averages from each week, starting in March and ending in December. The number from the severity rating and the letter from the likelihood rating, along with the risk color (green, yellow, or red) are shown in each risk matrix. This done for each location, with Fargo in the left column, and Finley in the right column. This work provides one of the biggest findings from this study, because it shows the true severity of the data anomalies. Risk matrices are one of the most commonly used methods for analysis in the aerospace industry, so this helps individuals with more of an aerospace background understand the significance of the engineering analysis in this report.

When comparing the overall results to each week, there weren't very many significant changes. The likelihood varied slightly from week to week in some categories. In fact, the only time the risk rating changed was from the prolonged altitude for greater than 6 scan rates in June, which went from unacceptable risk to acceptable risk with mitigation. Other than that specific case, the risk ratings stayed consistent. While the results look problematic, keep in mind that the risk assessment considers only the radar DAA system. The current airspace is safe, with near midair collisions (NMAC) and midair collisions (MAC) happening very infrequently. This is due to other DAA technologies and over 100 years of policy and technology going into the current NAS system. However, as UAS are incorporated into the NAS, they may only have one or two DAA systems, so the

results shown from Tables 17-21 may be more representative of that system. This shows that a robust logic to handle these anomalies is required to create a safe, efficient DAA system for UAS if radar was used. It also shows that multiple DAA systems may be required on UAS.

Additional findings show that Finley has more cases for unacceptable risk, when compared to Fargo. This can be explained by the increased surveillance area and scan rate, and typically an increased severity rating. Drop outs, altitude outliers, and prolonged altitude failures greater than six scan rates all pose unacceptable risk. Drop outs and altitude outliers are also considered the most catastrophic data anomalies. Every scenario with these behaviors poses either acceptable risk with mitigation or unacceptable risk. They also have identical risk ratings when comparing Fargo or Finley. Completely losing an aircraft with a drop out or having inconsistent altitude readings from altitude outliers poses the biggest risk for aircraft and ATC. Prolonged altitude readings are not as problematic because with a prolonged altitude reading of zero feet elevation, it is obvious that the aircraft is not at that elevation, so ATC can make more informed decisions on aircraft separation services. Additionally, an algorithm could do the same thing for an autonomous UAS. Multiple aircraft is the fourth lowest data anomaly in terms of risk, because ATC has protocols in place and is aware of the problem of assigning multiple aircraft with the same ID number. Lastly repeated data is the lowest risk, with acceptable risk for both locations. Receiving multiple data points doesn't contribute any harm to knowing the aircraft position. About the only potential harm is increasing bandwidth congestion in the airspace.

Table 17. Overall Radar Phenomena Risk Matrix

Description	Fargo Risk Rating	Finley Risk Rating
Drop out less than 2 scan rates	4B	3B
Drop outs lasting 2-6 scan rates	3C	2B
Drop outs lasting 6+ scan rates	2C	1C
Outlier less than 2 scan rates	4A	3B
Outlier lasting 2-6 scan rates	3C	2B
Outlier lasting 6+ scan rates	2C	1C
Prolonged alt failure less than 2 scan rates	4D	3D
Prolonged alt failure lasting 2-6 scan rates	3D	2D
Prolonged alt failure lasting 6+ scan rates	2C	1D
Multiple aircraft	3C	4C
Repeated data	5C	5C

Table 18. March Radar Phenomena Risk Matrix

Description	Fargo Risk Rating	Finley Risk Rating
Drop out less than 2 scan rates	4B	3B
Drop outs lasting 2-6 scan rates	3C	2B
Drop outs lasting 6+ scan rates	2C	1C
Outlier less than 2 scan rates	4A	3A
Outlier lasting 2-6 scan rates	3C	2B
Outlier lasting 6+ scan rates	2C	1C
Prolonged alt failure less than 2 scan rates	4D	3E
Prolonged alt failure lasting 2-6 scan rates	3D	2D
Prolonged alt failure lasting 6+ scan rates	2C	1D
Multiple aircraft	3C	4C
Repeated data	5C	5C

Table 19. June Radar Phenomena Risk Matrix

Description	Fargo Risk Rating	Finley Risk Rating
Drop out less than 2 scan rates	4B	3B
Drop outs lasting 2-6 scan rates	3C	2A
Drop outs lasting 6+ scan rates	2C	1C
Outlier less than 2 scan rates	4A	3A
Outlier lasting 2-6 scan rates	3C	2B
Outlier lasting 6+ scan rates	2C	1C
Prolonged alt failure less than 2 scan rates	4D	3D
Prolonged alt failure lasting 2-6 scan rates	3D	2D
Prolonged alt failure lasting 6+ scan rates	2D	1D
Multiple aircraft	3B	4C
Repeated data	5D	5C

Table 20. September Radar Phenomena Risk Matrix

Description	Fargo Risk Rating	Finley Risk Rating
Drop out less than 2 scan rates	4B	3B
Drop outs lasting 2-6 scan rates	3C	2A
Drop outs lasting 6+ scan rates	2C	1C
Outlier less than 2 scan rates	4A	3B
Outlier lasting 2-6 scan rates	3C	2B
Outlier lasting 6+ scan rates	2C	1C
Prolonged alt failure less than 2 scan rates	4D	3D
Prolonged alt failure lasting 2-6 scan rates	3D	2D
Prolonged alt failure lasting 6+ scan rates	2C	1D
Multiple aircraft	3B	4C
Repeated data	5C	5C

Table 21. December Radar Phenomena Risk Matrix

Description	Fargo Risk Rating	Finley Risk Rating
Drop out less than 2 scan rates	4B	3B
Drop outs lasting 2-6 scan rates	3C	2B
Drop outs lasting 6+ scan rates	2C	1C
Outlier less than 2 scan rates	4A	3B
Outlier lasting 2-6 scan rates	3C	2B
Outlier lasting 6+ scan rates	2C	1C
Prolonged alt failure less than 2 scan rates	4D	3D
Prolonged alt failure lasting 2-6 scan rates	3D	2D
Prolonged alt failure lasting 6+ scan rates	2C	1D
Multiple aircraft	3C	4C
Repeated data	5C	5C

Climate Effects

Weather is one of the factors that can impact the performance of a radar system. While clutter from weather, namely: rain and hail are important for professionals who study climate and the environment, they can severely hinder the performance of radars used to detect aircraft or other large objects. A significant portion of radar design goes into reducing the amount of clutter picked up. Typically, the Doppler frequency shift is the method to detect moving objects from the clutter of stationary objects. Circular polarization is another method to enhance the detection of aircraft in rain, because rain drops are symmetrical/spherical and aircraft are typically asymmetrical [58].

This section aims to understand the correlation between weather conditions and the data anomalies, if any. To do this weather information for each day of study was obtained [59]. For the Fargo analysis, the weather data from the Fargo International Airport was used. For the Finley study, the weather information from the Grand Forks Air Force Base was used, because this was the closest major airport with weather information dating back to 2015. Once the data was obtained, the weather was classified as fair weather and poor weather. The conditions were considered poor when the visibility was less than 1 mile, which is the minimum requirement for VFR traffic to take off and land in day light hours [60]. Conditions were also considered poor if the maximum wind speed exceeded 30mph, there was measured precipitation during the day, or any events were recorded (snow, fog, rain, or thunderstorms). If the weather during each day didn't experience any of the above conditions, it was considered fair weather.

The climate conditions were then compared to the percentage ratio of the instances of each type of data anomaly to the total number of aircraft detected that day. In some

cases, the ratios were greater than 100% because an aircraft can experience multiple instances of that specific data anomaly. Table 22 summarizes the instance/aircraft number ratio for each type of data anomaly in poor and fair weather at Fargo. The overall average results are listed first, with average of each week listed afterwards. Table 23 provides the same information, except for Finley.

Table 22. Fargo Climate Effects

	Weather Condition	Drop Outs	Altitude Outliers	Prolonged Altitude Failure	Repeated Data	Multiple Aircraft
Overall	Poor	94%	18%	1%	5%	4%
	Fair	134%	30%	2%	31%	2%
Mar.	Poor	75%	15%	2%	11%	1%
	Fair	118%	24%	1%	49%	1%
June	Poor	86%	14%	1%	7%	4%
	Fair	174%	38%	1%	15%	3%
Sep.	Poor	129%	22%	1%	2%	4%
	Fair	119%	26%	2%	82%	3%
Dec.	Poor	36%	9%	0%	0%	22%
	Fair	120%	30%	2%	23%	2%

Table 23. Finley Climate Effects

	Weather Condition	Drop Outs	Altitude Outliers	Prolonged Altitude Failure	Repeated Data	Multiple Aircraft
Overall	Poor	256%	38%	0%	4%	1%
	Fair	240%	47%	1%	8%	1%
Mar.	Poor	134%	20%	0%	2%	1%
	Fair	110%	28%	1%	8%	0%
June	Poor	201%	29%	0%	2%	1%
	Fair	269%	47%	1%	9%	1%
Sep.	Poor	388%	52%	0%	7%	1%
	Fair	487%	70%	1%	6%	1%
Dec.	Poor	246%	50%	1%	1%	2%
	Fair	232%	54%	1%	9%	1%

The effects on weather at Fargo can be seen much more easily than Finley, because Fargo covers a much smaller area than Finley. Finley may also be experiencing different climate conditions in different parts of its scan path, so it's difficult to draw meaningful conclusions. However, when looking at Table 22 for Fargo, drop outs, altitude outliers, and repeated data all seem to be effected by weather. It's interesting to see the ratio go down for poor weather days, while it increases for good weather days. This demonstrates the method of failure for those three anomalies listed above. With the poor weather, fewer aircraft are in the sky, however, for fair weather more aircraft are in the sky. This implies that drop outs, altitude outliers, and repeated data are a function of aircraft density. As more

aircraft are airborne, the density goes up, along with the ratio of instances to aircraft. The opposite trend can be seen with a lower aircraft density. The only exception to this behavior is the drop out average from September at Fargo. There was a slight increase in the instance to aircraft ratio for poor weather, rather than the typical increase for good weather. This was likely because of the small sample size in fair weather days (two days) versus the sample size in good weather days (five days). Other factors could also be in play to effect the results from that week of data.

The multiple aircraft with the same ID and prolonged altitude failures had minimal correlation with weather, with only a few percentage points separating the fair and poor weather days. The one exception to this rule was for multiple aircraft with the same ID in December at Fargo. Again, this is likely due to small sample size, with only one day of poor weather, versus six days of fair weather. This shows that transponder induced errors are not as susceptible to changes in climate conditions, because multiple aircraft with the same ID and prolonged altitude failures are the two behaviors that are most exclusively transponder induced. In the introduction, it was mentioned that secondary radar is better at handling poor weather conditions; these results further verify that statement.

To further show the correlation of climate and the data anomalies, a day to day summary for the June data at Fargo is provided in Table 24. When the weather is poor (rows highlighted in yellow/orange), the general trend is for the ratio of instances to aircraft to decrease when compared to fair weather days. A detailed day to day summary for all the data analyzed in this study at both Fargo and Finley is provided in Appendix E: Climate Effects Summary.

Table 24. Fargo Climate Effects for the Week of June

	6/1/2015	6/2/2015	6/3/2015	6/4/2015	6/5/2015	6/6/2015	6/7/2015
Temp. (°F)	high 74	80	71	75	76	75	86
	avg 63	71	64	66	66	66	71
	low 52	61	56	56	56	57	55
Dew Point (°F)	high 48	63	62	58	58	68	61
	avg 44	55	56	54	55	62	56
	low 37	45	53	49	54	51	49
Humidity (%)	high 66	73	97	93	93	90	100
	avg 52	59	85	73	70	79	66
	low 38	45	73	53	46	67	31
Sea Level Press. (in)	high 30.18	29.89	29.96	30.1	30.22	30.07	29.72
	avg 30.05	29.79	29.83	30.03	30.15	29.81	29.67
	low 29.9	29.64	29.66	29.96	30.07	29.65	29.63
Visibility (mi)	high 10	10	10	10	10	10	10
	avg 10	10	7	10	10	8	8
	low 10	10	1	10	10	1	0
Wind (mph)	high 29	37	18	20	15	26	23
	avg 19	18	13	9	9	12	10
	gust (high) 38	45	22	24	22	33	36
Precip. (in)	sum 0	T	0	0	0	0.58	T
Events		Rain				Rain	Fog, Rain
Ratio of Instances/Total Number of Aircraft Observed	Drop Outs 85%	95%	185%	181%	157%	88%	75%
	Outliers 16%	8%	49%	36%	29%	18%	16%
	Prolonged Altitude Failure 2.0%	0.3%	0.7%	0.9%	1.6%	1.0%	0.8%
	Repeated Data 11%	1%	24%	20%	1%	7%	9%
	Multiple Aircraft 0%	14%	1%	0%	8%	1%	1%

Additionally, with poor weather there are likely better pilots and better aircraft in the sky. With better aircraft, the equipment is less likely to malfunction, and pilots are

more likely to operate the equipment correctly. This may also explain why the ratio of data anomaly instances to total number of aircraft decreases for poor weather as well.

Comparison of Results to Previous Studies

While many studies have been performed on the performance and problems with radar, no study of this scale or scope has been done. This section will compare some previous work to the research presented in this document. Particularly the secondary radar vulnerabilities, the ADS-B/radar comparisons, and the wind turbine study. While the all the other research is relevant and important work, it is difficult to directly relate it to the findings from this study. If more specifics on the radar measurements were provided from Harris, studies looking at azimuth and range accuracy could also be addressed in the future.

The secondary radar vulnerability studies [39, 40] stated that aircraft transponders could be interrogated up to 20 times per scan rate depending on where the aircraft was relative to the radar site. This phenomenon likely explains why repeated data is occurring. With repeated data, many nearly identical data points were received within a fraction of a second. Although logic exists in the current radar system's computer to filter out most of the erroneous data, no software is ever perfect. The bandwidth congestion with interrogating the aircraft up to 20 times per scan rate, along with TCAS and ADS-B sharing some of the same frequencies as radar may contribute to the data anomalies observed as well. For the climate effects summary, it was shown as aircraft density increased, the ratio of data anomaly instances to total number of aircraft increased for drop outs, altitude outliers, and repeated data. When traffic density increases, so does the bandwidth congestion. In one of the ADS-B/radar studies [47], the authors struggled with correlating radar data to ADS-B data because of erratic radar and radar drop outs around the

Philadelphia airspace. This demonstrates that these radar data anomalies are occurring in other parts of the country, and especially in congested airspaces.

Additionally, the wind turbine study [41], has indicated that wind turbines around ATC radars can cause undesired effects, such as: electromagnetic shadowing, effects on Doppler, and clutter effects. While these effects are most often seen on primary radar resulting in erroneous data, the author suggests secondary radar could be effected as well. Some of the observed phenomenon from secondary radar effects were bearing and azimuth error, along with target splits (i.e. altitude outliers). A map depicting the over 47,000 wind turbines locations in the U.S. was obtained from the U.S. Geological Survey [61]. A map of the coverage area in the U.S. from both radars is provided in Figure 26, with a close up near Fargo and Finley provided in Figure 27. Each dot (the colors vary based on power output) represent a single wind turbine. The blue circle with an X represents Fargo and the red circle with an X represents Finley in both Figures 26 and 27.

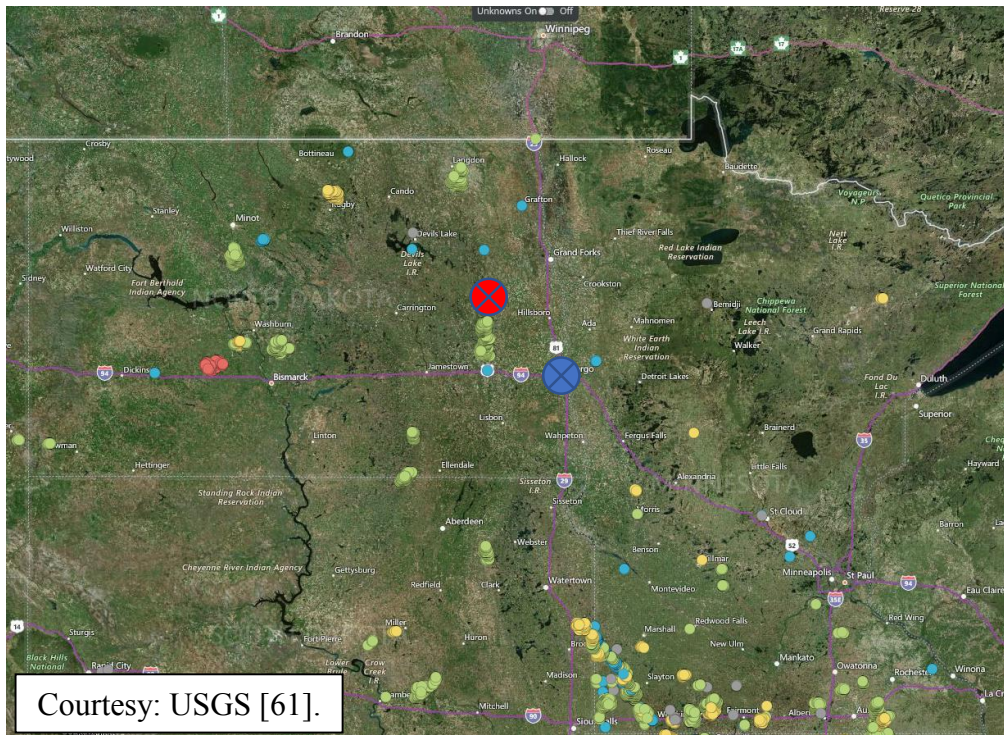


Figure 26. Wind Turbine Locations in the Fargo/Finley Radar Coverage Area

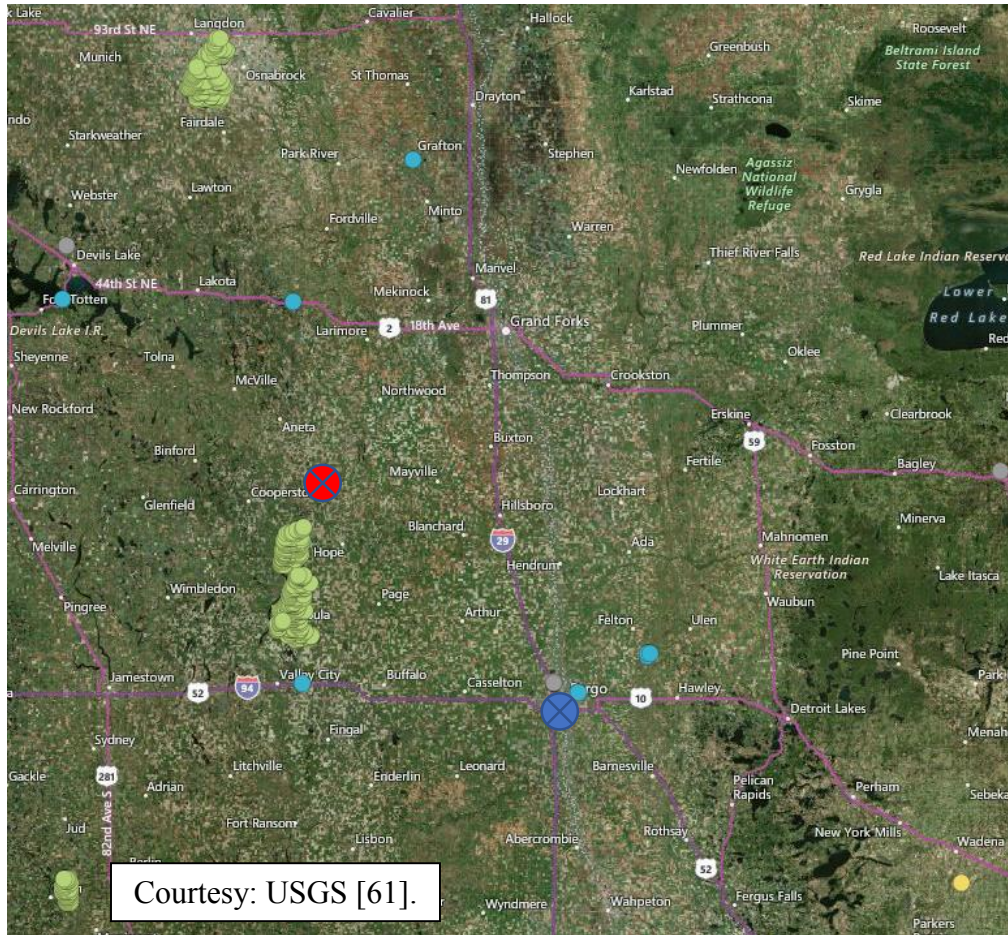


Figure 27. Close Up of the Wind Turbine Locations Near Fargo and Finley

It can be seen from Figures 26 and 27 there are quite a few wind turbines located close to the Fargo and Finley radar sites and in their scan areas. There are two wind turbines right next to the Fargo radar site, and a large wind turbine farm just south of the Finley radar site. It is likely these wind turbines may have contributed to the some of the drop outs and outliers due to interference with the radar's performance.

While many external factors may have contributed to the observed data anomalies, such as weather conditions, bandwidth congestion, traffic density, multiple interrogations per scan, and wind turbines. Other factors such as jamming (intentional or unintentional), engineering limitations of the radars, or simply design flaws could have contributed as

well. While it is important to understand where these data anomalies originated from, it is just as important to understand the anomalies themselves and know how to handle them if they do occur. Especially when radar data is used in UAS autonomy or DAA logic and for autonomous ATC systems.

CHAPTER V

CONCLUSIONS

The goal of this study was to understand the current state of the radar environment around the Red River Valley. To understand the status of the radar network, vulnerabilities needed to be understood. Chapter II described the five different types of data anomalies identified through this work, namely: drop outs, altitude outliers, prolonged altitude failures, repeated data, and multiple aircraft with the same ID. A MATLAB algorithm was developed to classify and sort out the data anomalies from the four weeks of data at both the Fargo radar site and the Finley radar site. This algorithm is described in detail in Chapter III.

Once the data was processed through the computer program developed, the results were analyzed by occurrence rate, time duration, and location. Finley always detected more aircraft each day because it is a long range radar, while Fargo is primarily used around the Fargo International Airport. The drop outs were the most prevalent data anomaly, with an average of 34% of aircraft at Fargo and 69% of aircraft at Finley experiencing at least one drop out. The other data anomalies ranked in order from most common (behind drop outs) to least common are altitude outliers, repeated data, multiple aircraft with the same ID, and prolonged altitude failures. These data anomalies occurred on 1-26% of the aircraft, depending on anomaly and location. Drop outs and prolonged altitude failures were analyzed by time duration. It was discovered the average drop lasted 23.58 seconds and 42.45 seconds, at Fargo and Finley, respectively. While 74% of drop outs occurred for less

than three scan rates at both locations, there were some drop outs lasting several minutes. The average prolonged altitude failure was 124.99 seconds at Fargo and 170.15 seconds at Finley, showing a much longer time duration than drop outs. Location was also analyzed, showing that many of the data anomalies occur near airports or on the edge of the effective radar radius.

Error source (radar or transponder induced), risk, weather impacts, and a correlation to previous studies was also done. The overlapping analysis, along with the location of data anomaly plots showed that drop outs, repeated data, and outliers are predominately radar induced errors. Multiple aircraft with the same ID and prolonged altitude failures are primarily transponder induced errors. While the weather conditions were not shown to directly impact the data anomalies, an indirect correlation was discovered. With poor weather, fewer aircraft are in the sky, decreasing the traffic density. It was also shown the ratio of instances to total number of aircraft detected decreased in poor weather as well. The opposite trend was observed with good weather, showing radar induced data anomalies are potentially a function of traffic density in the airspace. The risk assessment showed that multiple aircraft with the same ID and repeated data pose the lowest risk to aircraft, while prolonged altitude failures, altitude outliers, and drop outs pose the greatest risk. The comparison to previous studies also showed bandwidth congestion with multiple interrogations per scan rate and wind turbines can cause erroneous radar behavior.

While identifying the various data anomalies, their duration, and severity are eye opening. The airspace is still safe with the additional systems in place such as TCAS and ADS-B. However, with the addition of autonomous ATC systems and especially UAS into the NAS, the effect of these data anomalies could be compounded. If autonomous DAA

systems use radar data, work definitely needs to be done either on improving the radar systems or a robust logic to handle the data anomalies onboard the aircraft. Other systems will likely need to supplement UAS DAA, just like manned aircraft today. This will ensure that if there is a failure in one system, the aircraft is not flying without guidance. Future work could look at replicating the data anomalies to further understand how they are caused. This could be done at the Northern Plains UAS Test Site with their mobile radar unit. Additionally, to understand how radar data could be used for a DAA system onboard UAS, algorithms could be developed to handle the data anomalies and fuse other sources of data from other systems, such as: vision, ADS-B, or Light Detection and Ranging (LIDAR).

The information presented in this document can be used by developers of DAA logic for UAS and autonomous ATC systems. While knowing what types of behaviors radars are experiencing is important for radar designers, it is just as important for users of this data to know how to handle data anomalies. To create a robust DAA system and autonomous ATC system, understanding how to deal with radar data anomalies is crucial if that information is used for those systems. While newer technologies will supplement radar in the coming years, radar will likely be used for the foreseeable future.

APPENDIX A: MODE A TRANSPONDER IDENTIFICATION NUMBERS

0100-0400	Allocated to Service Area Operations for assignment for use by Terminal/CERAP, NAS Stakeholder, Unique Purpose and Experimental activities.
1200	Visual Flight Rules (VFR) aircraft that may or may not be in radio contact with an ATC Facility.
1201	For use by VFR aircraft in the immediate vicinity of LAX IAW FAR 93.95.
1202	Reserved for use by VFR gliders not in contact with ATC.
1205	(1)Reserved for use by VFR Helicopters within the Los Angeles region that may or may not be in contact with ATC. (2) VFR aircraft departing the DC Special Flight Rules Area (DC SFRA) fringe airports I.A.W. FAR 93.345.
1206	Reserved for use by VFR Law Enforcement, First Responder, Military and Public Service helicopters within the Los Angeles region that may or may not be in contact with ATC.
1234	VFR aircraft conducting pattern work at airports in the DC SFRA I.A.W. FAR 93.339.
1207-1272	Discrete 1200 series codes, unless otherwise allocated (e.g., 1255), designated for DVFR aircraft and only assigned by Flight Service Station (FSS).
1255	Firefighting aircraft.
1273-1275	Calibration Performance Monitoring Equipment (CPME), MRSM, and PARROT transponders.
1276	Air Defense Identification Zone (ADIZ) penetration when unable to establish communication with ATC or aeronautical facility.
1277	Designated Search and Rescue (SAR) aircraft.
0100-0700, 1000, 1100, 1300, 1500, 2000*, 2100, 2200, 2300, 2400, 4000	Non-discrete code assignments in accordance with FAA JO 7110.65, paragraph 5-2-2 Discreet Environment. 2000*, for use in oceanic airspace, unless another code is assigned by ATC.
4400	SR-71, F-12, U-2, B-57, pressure suit flights and aircraft operations above FL 600 in accordance with FAA JO 7110.65, paragraph 5-2-10., Beacon Code for Pressure Suit Flights Above FL 600.
4401-4433, 4466-4477	Reserved in accordance with FAA JO 7110.67, Special Aircraft Operations.
4434-4437	Weather reconnaissance, as appropriate.
4440-4441	Operations above FL600 for Lockheed/NASA from Moffett Field.
4442-4446	Operations above FL600 for Lockheed from Air Force Plant 42.
4447-4452	Allocated by the FAA for use in support of special flight activities I.A.W. FAA JO 7110.67.
4453	High balloon operations – National Scientific Balloon Facility, Palestine TX, and other providers, some in international operations.
4454-4465	Air Force operations above FL600 as designated in FAA JO 7610.4, Special Operations.
5100-5300	May be used by DOD aircraft beyond radar coverage but inside U.S. controlled airspace with coordination as appropriate with applicable Area Operations Directorate. DOD aircraft outside U.S. controlled airspace need to coordinate with the applicable Flight Information Region's (FIR) air traffic authorities.
5000-5057, 5063-5077 5400, 6100, 6400, 7501-7577	Reserved for use by DOD. The use of these code blocks can only be authorized and/or assigned by HQ NORAD or its designated representative. For information on the use of these codes contact NORAD J33C, n-nc.peterson.nj3.mbx.norad-j33c-omb@mail.mil
5061-5062, 5100, 5200,	Allocated by the FAA to Potomac TRACON (PCT) for use in the DC Special Flight Rules Area (SFRA) and Flight Restricted Zone (FRZ). Codes 5061 and 5062 will revert back to the DOD allocation code block when no longer needed in support of the NCR ADIZ.

7601-7607, 7701-7707	Allocated by the FAA for special use by Federal Law Enforcement Agencies.
7400	Reserved for an unmanned aircraft experiencing a lost link situation.
7500	Hijack in accordance with FAA JO 7610.4.
7600	Radio Failure in accordance with FAA JO 7110.65, paragraph 5-2-8 Radio Failure.
7700	Emergency in accordance with FAA JO 7110.65, paragraph 5-2-7 Emergency Code Assignment.
7777	DOD interceptor aircraft on active air defense missions and operating without ATC clearance in accordance with FAA JO 7610.4.
0500, 0600, 0700, 1000, 1100, 1300, 1400, 1500, 1600, 1700, 2000, 2100, 2200, 2300, 2400, 2500, 2600, 2700, 3000, 3100, 3200, 3300, 3400, 3500, 3600, 3700, 4000, 4100, 5600, 5700, 6000, 6200, 6300, 6500, 6600, 6700, 7000, 7100, 7200, 7300, 7610-7676, 7710-7776	External ARTCC subsets (Discrete codes of blocks only except for first primary block, which is used as the ARTCC's non-discrete code if all discrete codes are assigned.)
0000, 4200, 4300, 4500, 4600, 4700, 5100, 5200, 5300, 5500	Internal ARTCC subsets assigned by En Route Procedures Group, AJV-83. (Discrete codes only except for first primary block to be used as non-discrete if all discrete codes are assigned.)

APPENDIX B: MATLAB CODE

Main Script

```
clc
clear all
close all

all_data=[];
counter=0;

%Read file data
for ll=1:2
    location=ll
    for ii=1:4
        for kk=1:7
            if 3*ii<10
                Date= ['2015.0', num2str(3*ii), '.0', num2str(kk)]
            else
                Date= ['2015.', num2str(3*ii), '.0', num2str(kk)]
            end

Filepath='C:\Users\nicholas.allen\OneDrive - North Dakota University
System\UAS Research\Radars Analysis\Radars Data Merged Short\';

            if location==1
                Location='Fargo';
                file=['Short_Merged_', Date, '_', Location];
                filename= [Filepath, file, '.xlsx'];
            end

            if location==2
                Location='Finley';
                file=['Short_Merged_', Date, '_', Location];
                filename= [Filepath, file, '.xlsx'];
            end

            rd=Data_Read(filename);

            % New data ==> deletes all the rows with missing data
            rd_new = rd(all(~isnan(rd),2),:);

            %Convert Range & Azimuth to Latitude and longitude
            if location ==1
                lat=46.920222;
                lat1=deg2rad(lat);
                long=-96.812167;
                long1=deg2rad(long);

                radii1=60/41.1279;
                radii2=60/60.0266;
                %1 degree of latitude = 60.02647nmi
                %1 degree of longitude = 41.1279nmi
            end
end
```

```

if location ==2
lat=47.528167;
lat1=deg2rad(lat);
long=-97.90061;
long1=deg2rad(long);

radii1=250/40.6604;
radii2=250/60.0329;
%1 degree of latitude = 60.0329nmi
%1 degree of longitude = 40.6604nmi
end

ER=3440.27694; %Earth's radius in nautical miles
count= size(rd_new(:,1)); %count how many data points
lat_in=[];
long_in=[];

for i=1:count(1,1)
alt(i)=0.000164578833693*rd_new(i,2); %feet to nmi
if alt(i)< rd_new(i,5)
ran(i)=sqrt(rd_new(i,5)^2-alt(i)^2); %ground range (range in
data is slant range)
else
ran_prime(i)=sqrt(alt(i)^2-rd_new(i,5)^2);
beta(i)= asind(rd_new(i,5)/alt(i));
ran(i)=ran_prime(i)*sind(beta(i));
end
b(i)=ran(i)/ER; %conversion
rad(i)=deg2rad(rd_new(i,4));
lat2(i)=asin(sin(lat1)*cos(b(i))+cos(lat1)*sin(b(i))*cos(rad(i)));
a(i)=atan2(sin(rad(i))*sin(b(i))*cos(lat1), (cos(b(i))-
sin(lat1)*sin(lat2(i))));
long2(i)=long1+a(i);
long_f(i)=rad2deg(long2(i));
lat_f(i)=rad2deg(lat2(i));
long_in=[long_in long_f(i)];
lat_in=[lat_in lat_f(i)];
end

latitude=lat_in';
longitude=long_in';
rd_new_pos= [rd_new, ran', latitude, longitude];

% Sorted rows ==> sorts the data based on the aircraft ID number
rd_sorted= sortrows(rd_new_pos);

%split data ==> splits rd_sorted matrix into a cell array based on the
aircraft ID number
rd_split=arrayfun(@(x) rd_sorted(rd_sorted(:,1) == x, :),
unique(rd_sorted(:,1)), 'uniformoutput',false);
fprintf('done split rd \n')

%Count planes

```

```

    count_planes= size(unique(rd_new(:,1))); %count how many planes there
are

%% Multiple Aircraft & Repeated Data
[ multiple_aircraft_repeated_data, multiple_aircraft, repeated_data,
rd_wo_ma ] = MARD( location, count_planes, rd_split, rd_sorted );

Filepath_ma='C:\Users\nicholas.allen\OneDrive - North Dakota University
System\UAS Research\Radars Analysis\Multiple Aircraft Maps\';

addpath('altmany-export_fig-5be2ca4')

%Plot Multiple Aircraft
if length(unique(multiple_aircraft(:,2)))>2
ma_loc=[];
for mm=1:length(multiple_aircraft)
    if multiple_aircraft(mm,2) ~=0
        if multiple_aircraft(mm,2) ~=1200
            ma_loc=[ma_loc;multiple_aircraft(mm,:)];
        end
    end
end
end
plot(ma_loc(:,9),ma_loc(:,8),'.r','MarkerSize',10)
h=ellipse(radii1, radii2,0, long,lat, 'b');
set(gca,'visible','off')
set(gcf, 'Units', 'Inches', 'Position', [0, 0, 9, 9], 'PaperUnits',
'Inches', 'PaperSize', [9, 9])
plot_google_map('matype', 'hybrid','scale', 2, 'resize', 2)
savefig([Filepath_ma,file,'_multiple_aircraft_map.fig'])
set(gcf,'renderer','zbuffer')
export_fig([Filepath_ma,file,'_multiple_aircraft_map.png'])

close all
end

Filepath_rd='C:\Users\nicholas.allen\OneDrive - North Dakota University
System\UAS Research\Radars Analysis\Repeated Data Maps\';

%Plot Repeated Data
plot(repeated_data(:,9),repeated_data(:,8),'.r','MarkerSize',10)
h=ellipse(radii1, radii2,0, long,lat, 'b');
set(gca,'visible','off')
set(gcf, 'Units', 'Inches', 'Position', [0, 0, 9, 9], 'PaperUnits',
'Inches', 'PaperSize', [9, 9])
plot_google_map('matype', 'hybrid','scale', 2, 'resize', 2)
savefig([Filepath_rd,file,'_repeated_data_map.fig'])
set(gcf,'renderer','zbuffer')
export_fig([Filepath_rd,file,'_repeated_data_map.png'])

close all

%% Drop Outs
[ data_do, drop_out_summary, do_time_categorization ] = DropOut(
location, rd_wo_ma );

```

```

Filepath_do='C:\Users\nicholas.allen\OneDrive - North Dakota University
System\UAS Research\Radars Analysis\Drop Out Maps\';

plot(data_do(:,7),data_do(:,6),'.r','MarkerSize',10)
h=ellipse(radii1, radii2,0, long,lat, 'b');
set(gca,'visible','off')
set(gcf, 'Units', 'Inches', 'Position', [0, 0, 9, 9], 'PaperUnits',
'Inches', 'PaperSize', [9, 9])
plot_google_map('matype', 'hybrid','scale', 2, 'resize', 2)
savefig([Filepath_do,file,'_drop_out_map.fig'])
set(gcf,'renderer','zbuffer')
export_fig([Filepath_do,file,'_drop_out_map.png'])

close all

%% Outliers
[ outliers ] = Outliers( location, rd_wo_ma );

Filepath_out='C:\Users\nicholas.allen\OneDrive - North Dakota
University System\UAS Research\Radars Analysis\Outlier Maps\';

plot(outliers(:,9),outliers(:,8),'.r','MarkerSize',10)
h=ellipse(radii1, radii2,0, long,lat, 'b');
set(gca,'visible','off')
set(gcf, 'Units', 'Inches', 'Position', [0, 0, 9, 9], 'PaperUnits',
'Inches', 'PaperSize', [9, 9])
plot_google_map('matype', 'hybrid','scale', 2, 'resize', 2)
savefig([Filepath_out,file,'_outlier_map.fig'])
set(gcf,'renderer','zbuffer')
export_fig([Filepath_out,file,'_outlier_map.png'])

close all

%% Prolonged Altitude Failure
[p_alt_failure_pre, p_alt_failure, p_alt_failure_time,
alt_failure_summary, alt_failure_time_categorization ] =
prolonged_alt_failure( rd_wo_ma, location );

%obtain lat and long values
p_alt_lat_long=[];
for zz=1:length(p_alt_failure)
p_alt_lat_long=[p_alt_lat_long;p_alt_failure{zz}(:,7:8)];
end

Filepath_paf='C:\Users\nicholas.allen\OneDrive - North Dakota
University System\UAS Research\Radars Analysis\Prolonged Altitude
Failure Maps\';

if length(p_alt_lat_long)>=1
plot(p_alt_lat_long(:,2),p_alt_lat_long(:,1),'.r','MarkerSize',10)
h=ellipse(radii1, radii2,0, long,lat, 'b');
set(gca,'visible','off')
set(gcf, 'Units', 'Inches', 'Position', [0, 0, 9, 9], 'PaperUnits',
'Inches', 'PaperSize', [9, 9])
plot_google_map('matype', 'hybrid','scale', 2, 'resize', 2)

```

```

savefig([Filepath_paf,file,'p_alt_failure_map.fig'])
set(gcf,'renderer','zbuffer')
export_fig([Filepath_paf,file,'p_alt_failure_map.png'])
end

close all

%% Store all data
counter=counter+1;
repeated_data_instances=size(repeated_data);
alt_failure_summary_size=size(alt_failure_summary);

if alt_failure_summary_size(1)==0
    alt_failure_summary=cell(1,4);
    alt_failure_summary{1}=0;
    alt_failure_summary{2}=0;
    alt_failure_summary{3}=0;
    alt_failure_summary{4}=0;
    alt_failure_time_categorization=cell(1,7);
    alt_failure_time_categorization{1}=0;
    alt_failure_time_categorization{2}=0;
    alt_failure_time_categorization{3}=0;
    alt_failure_time_categorization{4}=0;
    alt_failure_time_categorization{5}=0;
    alt_failure_time_categorization{6}=0;
    alt_failure_time_categorization{7}=0;

end

if repeated_data_instances(1)>0
    all_data=[all_data; ll, 3*ii, kk, count_planes(1),
length(unique(data_do(:,1))), do_time_categorization{:, :},
drop_out_summary{:, :}, length(unique(outliers(:,1))),
length(outliers(:,1)), alt_failure_time_categorization{:, :},
alt_failure_summary{:, :}, length(unique(multiple_aircraft(:,1))),
length(unique(repeated_data(:,1))), repeated_data_instances(1)];
elseif repeated_data_instances(1)==0
    all_data=[all_data; ll, 3*ii, kk, count_planes(1),
length(unique(data_do(:,1))), do_time_categorization{:, :},
drop_out_summary{:, :}, length(unique(outliers(:,1))),
length(outliers(:,1)), alt_failure_time_categorization{:, :},
alt_failure_summary{:, :}, length(unique(multiple_aircraft(:,1))), 0,
repeated_data_instances(1)];
end

if counter==28
    xlswrite('Summary of Fargo Data Anomalies.xlsx', all_data)
elseif counter==56
    xlswrite('Summary of Fargo & Finley Data Anomalies.xlsx',
all_data)
end

%% Save .mat file
matFileDir= 'C:\Users\nicholas.allen\OneDrive - North Dakota
University System\UAS Research\Radars Analysis\mat Files\';

```

```
save([matFileDir,file, '.mat'])
disp('done data write')

% Clear variables
clearvars -except ii kk ll location all_data counter
end
end
end
```


Data Read

```
function [ rd ] = Data_Read( filename)
%This function reads in the data from the Excel sheets & creates the
radar
%data matrix with the required information in the matrix titled rd
% Location 1 is Fargo & Location 2 is Finley
% The output contains columns of the data in the following order:
planeid, time, altitude, azimuth, range

planeid = xlsread(filename, 'A:A');
time = xlsread(filename, 'B:B');
altitude = xlsread(filename, 'C:C');
azimuth = xlsread(filename, 'D:D');
range = xlsread(filename, 'E:E');
length1=length(planeid);
length2=length(altitude);
length3=length(time);
length4=length(azimuth);
length5=length(range);
length_comb=[length1, length2, length3, length4, length5];
max_size=max(length_comb);
for i=1:5
    diff=abs(max_size-length_comb(i));
    if diff>0
        if i==1
            planeid= [planeid; NaN(diff,1)];
        elseif i==2
            altitude= [altitude; NaN(diff,1)];
        elseif i==3
            time= [time; NaN(diff,1)];
        elseif i==4
            azimuth= [azimuth; NaN(diff,1)];
        elseif i==5
            range= [range; NaN(diff,1)];
        end
    end
end

rd= [planeid, time, altitude, azimuth, range];
end
```

Multiple Aircraft and Repeated Data

```

function [ multiple_aircraft_repeated_data, multiple_aircraft,
repeated_data, rd_wo_ma ] = MARD( location, count_planes, rd_split,
rd_sorted )
%This function identifies the plane ID's with multiple aircraft and
%repeated data
% Inputs: location = location number defined previously in the
script, count_planes=number of aircraft, rd_split=radar data split into
a cell array, rd_sorted= radar data
% Outputs: multiple_aircraft_repeated_data= a combination of the
multiple aircraft and repeated data, multiple_aircraft= data that has
multiple aircraft, repeated_data= data that has repeated data,
rd_wo_ma= radar data without multiple aircraft

multiple_aircraft_repeated_data=[];
multiple_aircraft=[];
if location==1
for kk=1:count_planes(1,1)
    datapts=size(rd_split{kk}(:,2)); %Count how many data points
there are for each aircraft
    for ll=1:(datapts(1,1)-1)
        delta_t_ma= rd_split{kk}(ll+1,2) - rd_split{kk}(ll,2);
        delta_lat= rd_split{kk}(ll+1,7) - rd_split{kk}(ll,7);
        delta_long= rd_split{kk}(ll+1,8) - rd_split{kk}(ll,8);
        if rd_split{kk}(ll,1)==0

multiple_aircraft=[multiple_aircraft;kk,rd_split{kk}(ll,:),delta_t_ma,
delta_lat, delta_long];
            end
            if delta_t_ma<3.8

multiple_aircraft_repeated_data=[multiple_aircraft_repeated_data;kk,rd_
split{kk}(ll,:),delta_t_ma, delta_lat, delta_long];
                if abs(delta_lat)>0.012
                    if abs(delta_long)>0.035

multiple_aircraft=[multiple_aircraft;kk,rd_split{kk}(ll,:),delta_t_ma,
delta_lat, delta_long];
                        end
                    end
                end
            end
        end
    end

%Repeated Data
repeated_data_potential=setdiff(multiple_aircraft_repeated_data,
multiple_aircraft, 'rows');

%Eliminate multiple aircraft from repeated data matrix
repeated_data=repeated_data_potential;
ma_unique=unique(multiple_aircraft(:,2));
for qq=1:length(ma_unique)
    repeated_data(any(repeated_data(:,2)==ma_unique(qq),2),:)=[];
end
end
end

```

```

if location==2
for kk=1:count_planes(1,1)
    datapts=size(rd_split{kk}(:,2)); %Count how many data points
there are for each aircraft
    for ll=1:(datapts(1,1)-1)
        delta_t_ma= rd_split{kk}(ll+1,2) - rd_split{kk}(ll,2);
        delta_lat= rd_split{kk}(ll+1,7) - rd_split{kk}(ll,7);
        delta_long= rd_split{kk}(ll+1,8) - rd_split{kk}(ll,8);
        if delta_t_ma<10

multiple_aircraft_repeated_data=[multiple_aircraft_repeated_data;kk,rd_
split{kk}(ll,:),delta_t_ma, delta_lat, delta_long];
            if abs(delta_lat)>0.016
                if abs(delta_long)>0.16

multiple_aircraft=[multiple_aircraft;kk,rd_split{kk}(ll,:),delta_t_ma,
delta_lat, delta_long];
                    end
                end
            end
        end
    end
end

%Repeated Data
repeated_data_potential=setdiff(multiple_aircraft_repeated_data,
multiple_aircraft, 'rows');

    %Eliminate multiple aircraft from repeated data matrix
repeated_data=repeated_data_potential;
ma_unique=unique(multiple_aircraft(:,2));
for qq=1:length(ma_unique)
    repeated_data(any(repeated_data(:,2)==ma_unique(qq),2),:)=[];
end

end

% Sort out multiple aircraft from data

%Find indices in drop out data that correspond to multiple aircraft
indices=[];
for oo=1:length(ma_unique)
    indices=[indices;find(rd_sorted(:,1)== ma_unique(oo))];
end

rd_wo_ma_combined=rd_sorted;
rd_wo_ma_combined(indices,:)=[]; %Remove indices that correspond to
multiple aircraft

%split data ==> splits rd_sorted matrix into a cell array based on the
%aircraft ID number without multiple aircraft
rd_wo_ma=arrayfun(@(x) rd_wo_ma_combined(rd_wo_ma_combined(:,1) == x,
:), unique(rd_wo_ma_combined(:,1)), 'uniformoutput',false);

end

```

Drop Outs

```
function [ data_do, drop_out_summary, do_time_categorization ] =
DropOut( location, rd_wo_ma )
%This function identifies drop outs from the radar without the multiple
%aircraft
% Inputs: location = location indicated by user (1 or 2, for Fargo or
Finley), rd_wo_ma= radar data without multiple aircraft
% Outputs: data_do= drop out data, drop_out_summary=avg, min, max and
% total number of drop outs, do_time_categorization= time
categorization of drop outs

data_do=[];

if location==1
for ii=1:length(rd_wo_ma)
    datapts=size(rd_wo_ma{ii}(:,2)); %Count how many data points
there are for each aircraft

    for jj=1:(datapts(1,1)-1)
        delta_t= rd_wo_ma{ii}(jj+1,2) - rd_wo_ma{ii}(jj,2);

        if delta_t>5.5 && delta_t<300

data_do=[data_do;ii,rd_wo_ma{ii}(jj,1),delta_t,rd_wo_ma{ii}(jj,2:3),rd_
wo_ma{ii}(jj,7:8)];

            end
        end
    end
end

if location==2
for ii=1:length(rd_wo_ma)
    datapts=size(rd_wo_ma{ii}(:,2)); %Count how many data points
there are for each aircraft

    for jj=1:(datapts(1,1)-1)
        delta_t= rd_wo_ma{ii}(jj+1,2) - rd_wo_ma{ii}(jj,2);
        if delta_t>13 && delta_t<400

data_do=[data_do;ii,rd_wo_ma{ii}(jj,1),delta_t,rd_wo_ma{ii}(jj,2:3),rd_
wo_ma{ii}(jj,7:8)];

            end
        end
    end
end

%% Categorize drop outs
average_do=mean(data_do(:,3)); %average
```

```

min_do=min(data_do(:,3)); %minimum
max_do=max(data_do(:,3)); %maximum
total_do=length(data_do(:,3)); %total number of drop outs
drop_out_summary=table(average_do, min_do, max_do, total_do);

%Time interval classification
do_count1=[];
do_count2=[];
do_count3=[];
do_count4=[];
do_count5=[];
do_count6=[];
do_count7=[];

if location==1
for aa=1:length(data_do(:,3))
    if data_do(aa,3)<10
        do_count1=[do_count1;data_do(aa,3)];
    elseif data_do(aa,3)>=10 && data_do(aa,3)<15
        do_count2=[do_count2;data_do(aa,3)];
    elseif data_do(aa,3)>=15 && data_do(aa,3)<20
        do_count3=[do_count3;data_do(aa,3)];
    elseif data_do(aa,3)>=20 && data_do(aa,3)<25
        do_count4=[do_count4;data_do(aa,3)];
    elseif data_do(aa,3)>=25 && data_do(aa,3)<30
        do_count5=[do_count5;data_do(aa,3)];
    elseif data_do(aa,3)>=30 && data_do(aa,3)<60
        do_count6=[do_count6;data_do(aa,3)];
    elseif data_do(aa,3)>=60
        do_count7=[do_count7;data_do(aa,3)];
    end
end

do_less10=length(do_count1);
do_10_15=length(do_count2);
do_15_20=length(do_count3);
do_20_25=length(do_count4);
do_25_30=length(do_count5);
do_30_60=length(do_count6);
do_greater60=length(do_count7);
do_time_categorization=table(do_less10,do_10_15, do_15_20, do_20_25,
do_25_30, do_30_60, do_greater60);
end

if location==2
for aa=1:length(data_do(:,3))
    if data_do(aa,3)<24
        do_count1=[do_count1;data_do(aa,3)];
    elseif data_do(aa,3)>=24 && data_do(aa,3)<36
        do_count2=[do_count2;data_do(aa,3)];
    elseif data_do(aa,3)>=36 && data_do(aa,3)<48
        do_count3=[do_count3;data_do(aa,3)];
    elseif data_do(aa,3)>=48 && data_do(aa,3)<60
        do_count4=[do_count4;data_do(aa,3)];
    elseif data_do(aa,3)>=60 && data_do(aa,3)<90
        do_count5=[do_count5;data_do(aa,3)];

```

```

elseif data_do(aa,3)>=90 && data_do(aa,3)<120
    do_count6=[do_count6;data_do(aa,3)];
elseif data_do(aa,3)>=120
    do_count7=[do_count7;data_do(aa,3)];
end
end

do_less24=length(do_count1);
do_24_36=length(do_count2);
do_36_48=length(do_count3);
do_48_60=length(do_count4);
do_60_90=length(do_count5);
do_90_120=length(do_count6);
do_greater120=length(do_count7);
do_time_categorization= table(do_less24,do_24_36, do_36_48, do_48_60,
do_60_90, do_90_120, do_greater120);
end

end

```

Altitude Outliers

```

function [ outliers ] = Outliers( location, rd_wo_ma )
%This function identifies drop outs from the radar without the multiple
%aircraft
% Inputs: location = location indicated by user (1 or 2, for Fargo or
Finley), rd_wo_ma= radar data without multiple aircraft
% Outputs: outliers= data with outliers, outlier_summary=average
time, min time, max time, and number of outliers
outlier_time_categorization= outliers categorized by time

outliers=[];
rd_split_outliers=cell(length(rd_wo_ma),1);
if location==1
for bb=1:length(rd_wo_ma)
    rows_to_last_row=size(rd_wo_ma{bb}(:,2));
    last_row=rd_wo_ma{bb}(rows_to_last_row(1,1),:);
    rd_split_outliers{bb}=[rd_wo_ma{bb}(:,:);last_row];
    datapts=size(rd_split_outliers{bb}(:,2));%Count how many data
points there are for each aircraft
    if datapts(1,1)>2
        for cc=1:(datapts(1,1)-2)

            delta_t_out= rd_split_outliers{bb}(cc+1,2) -
rd_split_outliers{bb}(cc,2);
            delta_alt_out= rd_split_outliers{bb}(cc+1,3) -
rd_split_outliers{bb}(cc,3);
            delta_alt_out_prev= rd_split_outliers{bb}(cc+2,3) -
rd_split_outliers{bb}(cc+1,3);

            if delta_t_out>4 && delta_t_out<300
                if delta_alt_out>800 || delta_alt_out<-800
                    if abs(delta_alt_out_prev)<800

outliers=[outliers;bb,rd_split_outliers{bb}(cc,:),delta_alt_out,delta_a
lt_out_prev, delta_t_out];
                    end
                end
            end
        end
    end
end
end

if location==2
for bb=1:length(rd_wo_ma)
    rows_to_last_row=size(rd_wo_ma{bb}(:,2));
    last_row=rd_wo_ma{bb}(rows_to_last_row(1,1),:);
    rd_split_outliers{bb}=[rd_wo_ma{bb}(:,:);last_row];
    datapts=size(rd_split_outliers{bb}(:,2));%Count how many data
points there are for each aircraft
    if datapts(1,1)>2
        for cc=1:(datapts(1,1)-2)

```

```

        delta_t_out= rd_split_outliers{bb}(cc+1,2) -
rd_split_outliers{bb}(cc,2);
        delta_alt_out= rd_split_outliers{bb}(cc+1,3) -
rd_split_outliers{bb}(cc,3);
        delta_alt_out_prev= rd_split_outliers{bb}(cc+2,3) -
rd_split_outliers{bb}(cc+1,3);

        if delta_t_out>10 && delta_t_out<400
            if delta_alt_out>800 || delta_alt_out<-800
                if abs(delta_alt_out_prev)<800

outliers=[outliers;bb,rd_split_outliers{bb}(cc,:),delta_alt_out,delta_a
lt_out_prev, delta_t_out];
                    end
                end
            end
        end
end
end

if length(outliers)<1
    outliers=[];
    return
end

```


Prolonged Altitude Failure

```

function [p_alt_failure_split_pre, p_alt_failure, p_alt_failure_time,
alt_failure_summary, alt_failure_time_categorization ] =
prolonged_alt_failure( rd_wo_ma, location )
%This function finds all the prolonged altitude failures at 0ft
% Inputs: rd_wo_ma= radar data without multiple aircraft
% Outputs: p_alt_failure: data with prolonged altitude failure,
% p_alt_failure_time= time duration of failures,
alt_failure_summary=avg, min, max and total number of alt failures,
alt_failure_time_categorization= time categorization of alt failures

start_stop=cell(length(rd_wo_ma),1);
all_indices=cell(length(rd_wo_ma),1);
p_alt_failure_pre=[];
for ii=1:length(rd_wo_ma)
    datapts=size(rd_wo_ma{ii}(:,2)); %Count how many data points
    there are for each aircraft

    alt= rd_wo_ma{ii}(:,3)';
    %logical array for when alt is equal to 0 ft
    t_alt= (abs(alt) >= 1);
    %Find the starting and ending indices and duration of the
string of zeroes
    d_alt = diff([1 t_alt 1]);
    startIndex = find(d_alt < 0);
    endIndex = find(d_alt > 0)-1;
    duration = endIndex-startIndex+1;
    %Find the strings of zeros with a duration greater than 2
to indicate a prolonged altitude failure
    stringIndex = (duration >= 2);
    startIndex = startIndex(stringIndex);
    endIndex = endIndex(stringIndex);
    start_stop{ii}= [startIndex; endIndex];

    %Find indices corresponding to prolonged altitude failure
    indices = zeros(1,max(endIndex)+1);
    indices(startIndex) = 1;
    indices(endIndex+1) = indices(endIndex+1)-1;
    indices = find(cumsum(indices));
    all_indices{ii}=indices;

    if length(endIndex)>=1
        for kk=1:length(endIndex)
            p_alt_failure_pre=[p_alt_failure_pre;
rd_wo_ma{ii}(startIndex(kk):endIndex(kk),:)]';
        end
    end
end

%Output matrix with altitude readings of 0ft 2 more times in a row
p_alt_failure_split_pre=arrayfun(@ (x)
p_alt_failure_pre(p_alt_failure_pre(:,1) == x, :),
unique(p_alt_failure_pre(:,1)), 'uniformoutput',false);

```

```

%Separate aircraft at different time of day & only include aircraft
with
%readings of 3 or more 0ft alt readings in a row
p_alt_failure=[];

for mm=1:length(p_alt_failure_split_pre)
    current_length=length(p_alt_failure);

    time_diff=find(diff(p_alt_failure_split_pre{mm}(:,2))<300);
    index=setdiff(1:(length(p_alt_failure_split_pre{mm}(:,2))-
1),time_diff);
    index=[0, index, length(p_alt_failure_split_pre{mm}(:,2))];
    if length(index)>2
        ind_count=length(index);
        for hh = 1:(ind_count-1)

            p_alt_failure{current_length+hh}=
p_alt_failure_split_pre{mm}((index(hh)+1):index(hh+1),:);
        end
    else
        p_alt_failure{current_length+1}=p_alt_failure_split_pre{mm}(:,:);
    end

end
for qq=1:length(p_alt_failure)
    length_matrix=size(p_alt_failure{qq});
    if length_matrix(1)<3
        p_alt_failure{qq}=[];
    end
end

%eliminate empty arrays
TF=[];
for aa=1:length(p_alt_failure)
    TF=[TF; isempty(p_alt_failure{aa})];
end
p_alt_failure=p_alt_failure(find(TF==0));
p_alt_failure=p_alt_failure';

    if length(p_alt_failure)<1
        p_alt_failure=[];
        p_alt_failure_time=[];
        alt_failure_summary=[];
        alt_failure_time_categorization=[];
        return
    end

%Categorize by time interval & stats
p_alt_failure_time=[];
    for mm=1:length(p_alt_failure)
        points=length(p_alt_failure{mm}(:,1)); %Count how many data points
there are for each aircraft

p_alt_failure_time=[p_alt_failure_time;p_alt_failure{mm}(points,2)-
p_alt_failure{mm}(1,2)];

```

```

end

alt_failures=p_alt_failure_time;
average_alt_failure=mean(alt_failures(:,1)); %average
min_alt_failure=min(alt_failures(:,1)); %minimum
max_alt_failure=max(alt_failures(:,1)); %maximum
total_alt_failures=length(alt_failures(:,1)); %total number of alt
failure
alt_failure_summary=table(average_alt_failure, min_alt_failure,
max_alt_failure, total_alt_failures);

%Time interval classification
alt_failure_count1=[];
alt_failure_count2=[];
alt_failure_count3=[];
alt_failure_count4=[];
alt_failure_count5=[];
alt_failure_count6=[];
alt_failure_count7=[];

if location==1
for aa=1:length(alt_failures(:,1))
    if alt_failures(aa,1)<10
        alt_failure_count1=[alt_failure_count1;alt_failures(aa,1)];
    elseif alt_failures(aa,1)>=10 && alt_failures(aa,1)<15
        alt_failure_count2=[alt_failure_count2;alt_failures(aa,1)];
    elseif alt_failures(aa,1)>=15 && alt_failures(aa,1)<20
        alt_failure_count3=[alt_failure_count3;alt_failures(aa,1)];
    elseif alt_failures(aa,1)>=20 && alt_failures(aa,1)<25
        alt_failure_count4=[alt_failure_count4;alt_failures(aa,1)];
    elseif alt_failures(aa,1)>=25 && alt_failures(aa,1)<30
        alt_failure_count5=[alt_failure_count5;alt_failures(aa,1)];
    elseif alt_failures(aa,1)>=30 && alt_failures(aa,1)<60
        alt_failure_count6=[alt_failure_count6;alt_failures(aa,1)];
    elseif alt_failures(aa,1)>=60
        alt_failure_count7=[alt_failure_count7;alt_failures(aa,1)];
    end
end

alt_failure_less10=length(alt_failure_count1);
alt_failure_10_15=length(alt_failure_count2);
alt_failure_15_20=length(alt_failure_count3);
alt_failure_20_25=length(alt_failure_count4);
alt_failure_25_30=length(alt_failure_count5);
alt_failure_30_60=length(alt_failure_count6);
alt_failure_greater60=length(alt_failure_count7);
alt_failure_time_categorization=
table(alt_failure_less10,alt_failure_10_15, alt_failure_15_20,
alt_failure_20_25, alt_failure_25_30, alt_failure_30_60,
alt_failure_greater60);
end

if location==2
for aa=1:length(alt_failures(:,1))
    if alt_failures(aa,1)<24
        alt_failure_count1=[alt_failure_count1;alt_failures(aa,1)];

```

```

elseif alt_failures(aa,1)>=24 && alt_failures(aa,1)<36
    alt_failure_count2=[alt_failure_count2;alt_failures(aa,1)];
elseif alt_failures(aa,1)>=36 && alt_failures(aa,1)<48
    alt_failure_count3=[alt_failure_count3;alt_failures(aa,1)];
elseif alt_failures(aa,1)>=48 && alt_failures(aa,1)<60
    alt_failure_count4=[alt_failure_count4;alt_failures(aa,1)];
elseif alt_failures(aa,1)>=60 && alt_failures(aa,1)<90
    alt_failure_count5=[alt_failure_count5;alt_failures(aa,1)];
elseif alt_failures(aa,1)>=90 && alt_failures(aa,1)<120
    alt_failure_count6=[alt_failure_count6;alt_failures(aa,1)];
elseif alt_failures(aa,1)>=120
    alt_failure_count7=[alt_failure_count7;alt_failures(aa,1)];
end
end

alt_failure_less24=length(alt_failure_count1);
alt_failure_24_36=length(alt_failure_count2);
alt_failure_36_48=length(alt_failure_count3);
alt_failure_48_60=length(alt_failure_count4);
alt_failure_60_90=length(alt_failure_count5);
alt_failure_90_120=length(alt_failure_count6);
alt_failure_greater120=length(alt_failure_count7);
alt_failure_time_categorization=
table(alt_failure_less24,alt_failure_24_36, alt_failure_36_48,
alt_failure_48_60, alt_failure_60_90, alt_failure_90_120,
alt_failure_greater120);
end

end

```

Plot Google Map

```
function varargout = plot_google_map(varargin)
% function h = plot_google_map(varargin)
% Plots a google map on the current axes using the Google Static Maps
API
%
% USAGE:
% h = plot_google_map(Property, Value,...)
% Plots the map on the given axes. Used also if no output is specified
%
% Or:
% [lonVec latVec imag] = plot_google_map(Property, Value,...)
% Returns the map without plotting it
%
% PROPERTIES:
%   Axis          - Axis handle. If not given, gca is used.
%   Height (640)  - Height of the image in pixels (max 640)
%   Width  (640)  - Width of the image in pixels (max 640)
%   Scale (2)     - (1/2) Resolution scale factor. Using Scale=2 will
%                  double the resolution of the downloaded image (up
%                  to 1280x1280) and will result in finer rendering,
%                  but processing time will be longer.
%   Resize (1)   - (recommended 1-2) Resolution upsampling factor.
%                  Increases image resolution using imresize(). This
%                  results
%                  in a finer image but it needs the image
%                  processing
%                  toolbox and processing time will be longer.
%   MapType      - ('roadmap') Type of map to return. Any of
%                  [roadmap,
%                  satellite, terrain, hybrid]. See the Google Maps
%                  API for
%                  more information.
%   Alpha (1)    - (0-1) Transparency level of the map (0 is fully
%                  transparent). While the map is always moved to
%                  the
%                  bottom of the plot (i.e. will not hide previously
%                  drawn items), this can be useful in order to
%                  increase
%                  readability if many colors are plotted
%                  (using SCATTER for example).
%   ShowLabels (1) - (0/1) Controls whether to display city/street
%                  textual labels on the map
%   Style         - (string) A style configuration string. See:
%                  https://developers.google.com/maps/documentation/static-
%                  maps/?csw=1#StyledMaps
%                  http://instrument.github.io/styled-maps-wizard/
%   Language     - (string) A 2 letter ISO 639-1 language code for
%                  displaying labels in a
%                  local language instead of English (where
%                  available).
%                  For example, for Chinese use:
%                  plot_google_map('language','zh')
%                  For the list of codes, see:
```

```

%          http://en.wikipedia.org/wiki/List_of_ISO_639-
l_codes
%      Marker      - The marker argument is a text string with fields
%                  conforming to the Google Maps API. The
%                  following are valid examples:
%                  '43.0738740,-70.713993' (default midsize orange
marker)
%                  '43.0738740,-70.713993,blue' (midsize blue
marker)
%                  '43.0738740,-70.713993,yellowa' (midsize yellow
%                  marker with label "A")
%                  '43.0738740,-70.713993,tinyredb' (tiny red marker
%                  with label "B")
%      Refresh (1)  - (0/1) defines whether to automatically refresh
the
%                  map upon zoom/pan action on the figure.
%      AutoAxis (1) - (0/1) defines whether to automatically adjust the
axis
%                  of the plot to avoid the map being stretched.
%                  This will adjust the span to be correct
%                  according to the shape of the map axes.
%      FigureResizeUpdate (1) - (0/1) defines whether to automatically
refresh the
%                  map upon resizing the figure. This will ensure
map
%                  isn't stretched after figure resize.
%      APIKey      - (string) set your own API key which you obtained
from Google:
%
%      http://developers.google.com/maps/documentation/staticmaps/#api_key
%                  This will enable up to 25,000 map requests per
day,
%                  compared to a few hundred requests without a key.
%                  To set the key, use:
%
%      plot_google_map('APIKey','SomeLongStringObtainedFromGoogle')
%                  You need to do this only once to set the key.
%                  To disable the use of a key, use:
%                  plot_google_map('APIKey','')
%
% OUTPUT:
%      h          - Handle to the plotted map
%
%      lonVect    - Vector of Longitude coordinates (WGS84) of the
image
%      latVect    - Vector of Latitude coordinates (WGS84) of the
image
%      imag       - Image matrix (height,width,3) of the map
%
% EXAMPLE - plot a map showing some capitals in Europe:
%      lat = [48.8708  51.5188  41.9260  40.4312  52.523  37.982];
%      lon = [2.4131  -0.1300  12.4951  -3.6788  13.415  23.715];
%      plot(lon,lat,'.r','MarkerSize',20)
%      plot_google_map
%
% References:
%      http://www.mathworks.com/matlabcentral/fileexchange/24113

```

```

% http://www.maptiler.org/google-maps-coordinates-tile-bounds-
projection/
% http://developers.google.com/maps/documentation/staticmaps/
%
% Acknowledgements:
% Val Schmidt for his submission of get_google_map.m
%
% Author:
% Zohar Bar-Yehuda
%
% Version 1.8 - 25/04/2016 - By Hannes Diethelm
%     - Add resize parameter to resize image using imresize()
%     - Fix scale parameter
% Version 1.7 - 14/04/2016
%     - Add custom style support
% Version 1.6 - 12/11/2015
%     - Use system temp folder for writing image files (with fallback
to current dir if missing write permissions)
% Version 1.5 - 20/11/2014
%     - Support for MATLAB R2014b
%     - several fixes for complex layouts: several maps in one
figure,
%         map inside a panel, specifying axis handle as input (thanks
to Luke Plausin)
% Version 1.4 - 25/03/2014
%     - Added the language parameter for showing labels in a local
language
%     - Display the URL on error to allow easier debugging of API
errors
% Version 1.3 - 06/10/2013
%     - Improved functionality of AutoAxis, which now handles any
shape of map axes.
%     - Now also updates the extent of the map if the figure is
resized.
%     - Added the showLabels parameter which allows hiding the
textual labels on the map.
% Version 1.2 - 16/06/2012
%     - Support use of the "scale=2" parameter by default for finer
rendering (set scale=1 if too slow).
%     - Auto-adjust axis extent so the map isn't stretched.
%     - Set and use an API key which enables a much higher usage
volume per day.
% Version 1.1 - 25/08/2011

persistent apiKey useTemp
if isnumeric(apiKey)
    % first run, check if API key file exists
    if exist('api_key.mat','file')
        load api_key
    else
        apiKey = '';
    end
end

if isempty(useTemp)
    % first run, check if we have write access to the temp folder

```

```

try
    tempfilename = tempname;
    fid = fopen(tempfilename, 'w');
    if fid > 0
        fclose(fid);
        useTemp = true;
        delete(tempfilename);
    else
        % Don't have write access to temp folder or it doesn't
        exist, fallback to current dir
        useTemp = false;
    end
catch
    % in case tempname fails for some reason
    useTemp = false;
end
end

```

```
hold on
```

```
% Default parametr
```

```

axHandle = gca;
height = 640;
width = 640;
scale = 2;
resize = 1;
maptype = 'roadmap';
alphaData = 1;
autoRefresh = 1;
figureResizeUpdate = 1;
autoAxis = 1;
showLabels = 1;
language = '';
markeridx = 1;
markerlist = {};
style = '';

```

```
% Handle input arguments
```

```

if nargin >= 2
    for idx = 1:2:length(varargin)
        switch lower(varargin{idx})
            case 'axis'
                axHandle = varargin{idx+1};
            case 'height'
                height = varargin{idx+1};
            case 'width'
                width = varargin{idx+1};
            case 'scale'
                scale = round(varargin{idx+1});
                if scale < 1 || scale > 2
                    error('Scale must be 1 or 2');
                end
            case 'resize'
                resize = varargin{idx+1};
            case 'maptype'
                maptype = varargin{idx+1};

```



```

        case 'alpha'
            alphaData = varargin{idx+1};
        case 'refresh'
            autoRefresh = varargin{idx+1};
        case 'showlabels'
            showLabels = varargin{idx+1};
        case 'figureresizeupdate'
            figureResizeUpdate = varargin{idx+1};
        case 'language'
            language = varargin{idx+1};
        case 'marker'
            markerlist{markeridx} = varargin{idx+1};
            markeridx = markeridx + 1;
        case 'autoaxis'
            autoAxis = varargin{idx+1};
        case 'apikey'
            apiKey = varargin{idx+1}; % set new key
            % save key to file
            funcFile = which('plot_google_map.m');
            pth = fileparts(funcFile);
            keyFile = fullfile(pth, 'api_key.mat');
            save(keyFile, 'apiKey')
        case 'style'
            style = varargin{idx+1};
        otherwise
            error(['Unrecognized variable: ' varargin{idx}])
    end
end
end
if height > 640
    height = 640;
end
if width > 640
    width = 640;
end

% Store paramters in axis handle (for auto refresh callbacks)
ud = get(axHandle, 'UserData');
if isempty(ud)
    % explicitly set as struct to avoid warnings
    ud = struct;
end
ud.gmap_params = varargin;
set(axHandle, 'UserData', ud);

curAxis = axis(axHandle);
if max(abs(curAxis)) > 500 || curAxis(3) > 90 || curAxis(4) < -90
    warning('Axis limits are not reasonable for WGS1984, ignoring.
Please make sure your plotted data in WGS1984 coordinates,')
    return;
end

% Enforce Latitude constraints of EPSG:900913
if curAxis(3) < -85
    curAxis(3) = -85;
end

```

```

if curAxis(4) > 85
    curAxis(4) = 85;
end
% Enforce longitude constrains
if curAxis(1) < -180
    curAxis(1) = -180;
end
if curAxis(1) > 180
    curAxis(1) = 0;
end
if curAxis(2) > 180
    curAxis(2) = 180;
end
if curAxis(2) < -180
    curAxis(2) = 0;
end

if isequal(curAxis,[0 1 0 1]) % probably an empty figure
    % display world map
    curAxis = [-200 200 -85 85];
    axis(curAxis)
end

if autoAxis
    % adjust current axis limit to avoid stretched maps
    [xExtent,yExtent] = latLonToMeters(curAxis(3:4), curAxis(1:2) );
    xExtent = diff(xExtent); % just the size of the span
    yExtent = diff(yExtent);
    % get axes aspect ratio
    drawnow
    org_units = get(axHandle, 'Units');
    set(axHandle, 'Units', 'Pixels')
    ax_position = get(axHandle, 'position');
    set(axHandle, 'Units', org_units)
    aspect_ratio = ax_position(4) / ax_position(3);

    if xExtent*aspect_ratio > yExtent
        centerX = mean(curAxis(1:2));
        centerY = mean(curAxis(3:4));
        spanX = (curAxis(2)-curAxis(1))/2;
        spanY = (curAxis(4)-curAxis(3))/2;

        % enlarge the Y extent
        spanY = spanY*xExtent*aspect_ratio/yExtent; % new span
        if spanY > 85
            spanX = spanX * 85 / spanY;
            spanY = spanY * 85 / spanY;
        end
        curAxis(1) = centerX-spanX;
        curAxis(2) = centerX+spanX;
        curAxis(3) = centerY-spanY;
        curAxis(4) = centerY+spanY;
    elseif yExtent > xExtent*aspect_ratio

        centerX = mean(curAxis(1:2));

```

```

        centerY = mean(curAxis(3:4));
        spanX = (curAxis(2)-curAxis(1))/2;
        spanY = (curAxis(4)-curAxis(3))/2;
        % enlarge the X extent
        spanX = spanX*yExtent/(xExtent*aspect_ratio); % new span
        if spanX > 180
            spanY = spanY * 180 / spanX;
            spanX = spanX * 180 / spanX;
        end

        curAxis(1) = centerX-spanX;
        curAxis(2) = centerX+spanX;
        curAxis(3) = centerY-spanY;
        curAxis(4) = centerY+spanY;
    end
    % Enforce Latitude constraints of EPSG:900913
    if curAxis(3) < -85
        curAxis(3:4) = curAxis(3:4) + (-85 - curAxis(3));
    end
    if curAxis(4) > 85
        curAxis(3:4) = curAxis(3:4) + (85 - curAxis(4));
    end
    axis(axHandle, curAxis); % update axis as quickly as possible,
before downloading new image
    drawnow
end

% Delete previous map from plot (if exists)
if nargout <= 1 % only if in plotting mode
    curChildren = get(axHandle, 'children');
    map_objs = findobj(curChildren, 'tag', 'gmap');
    bd_callback = [];
    for idx = 1:length(map_objs)
        if ~isempty(get(map_objs(idx), 'ButtonDownFcn'))
            % copy callback properties from current map
            bd_callback = get(map_objs(idx), 'ButtonDownFcn');
        end
    end
    delete(map_objs)
end

% Calculate zoom level for current axis limits
[xExtent, yExtent] = latLonToMeters(curAxis(3:4), curAxis(1:2) );
minResX = diff(xExtent) / width;
minResY = diff(yExtent) / height;
minRes = max([minResX minResY]);
tileSize = 256;
initialResolution = 2 * pi * 6378137 / tileSize; % 156543.03392804062
for tileSize 256 pixels
zoomlevel = floor(log2(initialResolution/minRes));

% Enforce valid zoom levels
if zoomlevel < 0
    zoomlevel = 0;
end

```

```

if zoomlevel > 19
    zoomlevel = 19;
end

% Calculate center coordinate in WGS1984
lat = (curAxis(3)+curAxis(4))/2;
lon = (curAxis(1)+curAxis(2))/2;

% Construct query URL
preamble = 'http://maps.googleapis.com/maps/api/staticmap';
location = ['?center=' num2str(lat,10) ',' num2str(lon,10)];
zoomStr = ['&zoom=' num2str(zoomlevel)];
sizeStr = ['&scale=' num2str(scale) '&size=' num2str(width) 'x'
num2str(height)];
maptypeStr = ['&maptype=' maptype ];
if ~isempty(apiKey)
    keyStr = ['&key=' apiKey];
else
    keyStr = '';
end
markers = '&markers=';
for idx = 1:length(markerlist)
    if idx < length(markerlist)
        markers = [markers markerlist{idx} '%7C'];
    else
        markers = [markers markerlist{idx}];
    end
end

if showLabels == 0
    if ~isempty(style)
        style(end+1) = '|';
    end
    style = [style 'feature:all|element:labels|visibility:off'];
end

if ~isempty(language)
    languageStr = ['&language=' language];
else
    languageStr = '';
end

if ismember(maptype,{'satellite','hybrid'})
    filename = 'tmp.jpg';
    format = '&format=jpg';
    convertNeeded = 0;
else
    filename = 'tmp.png';
    format = '&format=png';
    convertNeeded = 1;
end
sensor = '&sensor=false';

if ~isempty(style)
    styleStr = ['&style=' style];
else

```

```

        styleStr = '';
    end

url = [preamble location zoomStr sizeStr maptypeStr format markers
languageStr sensor keyStr styleStr];

% Get the image
if useTemp
    filepath = fullfile(tempdir, filename);
else
    filepath = filename;
end

try
    urlwrite(url,filepath);
catch % error downloading map
    warning(['Unable to download map form Google Servers.\n' ...
            'Matlab error was: %s\n\n' ...
            'Possible reasons: missing write permissions, no network
connection, quota exceeded, or some other error.\n' ...
            'Consider using an API key if quota problems persist.\n\n' ...
            'To debug, try pasting the following URL in your browser, which
may result in a more informative error:\n%s'], lasterr, url);
    varargout{1} = [];
    varargout{2} = [];
    varargout{3} = [];
    return
end

[M, Mcolor] = imread(filepath);
Mcolor = uint8(Mcolor * 255);
%M = cast(M,'double');
delete(filepath); % delete temp file
width = size(M,2);
height = size(M,1);

% We now want to convert the image from a colormap image with an uneven
% mesh grid, into an RGB truecolor image with a uniform grid.
% This would enable displaying it with IMAGE, instead of PCOLOR.
% Advantages are:
% 1) faster rendering
% 2) makes it possible to display together with other colormap
% annotations (PCOLOR, SCATTER etc.)

% Convert image from colormap type to RGB truecolor (if PNG is used)
if convertNeeded
    imag = zeros(height,width,3, 'uint8');
    for idx = 1:3
        cur_map = Mcolor(:,idx);
        imag(:, :, idx) = reshape(cur_map(M+1),height,width);
    end
else
    imag = M;
end
% Resize if needed
if resize ~= 1

```

```

    imag = imresize(imag, resize, 'bilinear');
end

% Calculate a meshgrid of pixel coordinates in EPSG:900913
width = size(imag,2);
height = size(imag,1);
centerPixelY = round(height/2);
centerPixelX = round(width/2);
[centerX,centerY] = latLonToMeters(lat, lon ); % center coordinates in
EPSG:900913
curResolution = initialResolution / 2^zoomlevel / scale / resize; %
meters/pixel (EPSG:900913)
xVec = centerX + ((1:width)-centerPixelX) * curResolution; % x vector
yVec = centerY + ((height:-1:1)-centerPixelY) * curResolution; % y
vector
[xMesh,yMesh] = meshgrid(xVec,yVec); % construct meshgrid

% convert meshgrid to WGS1984
[lonMesh,latMesh] = metersToLatLon(xMesh,yMesh);

% Next, project the data into a uniform WGS1984 grid
uniHeight = round(height*resize);
uniWidth = round(width*resize);
latVect = linspace(latMesh(1,1),latMesh(end,1),uniHeight);
lonVect = linspace(lonMesh(1,1),lonMesh(1,end),uniWidth);
[uniLonMesh,uniLatMesh] = meshgrid(lonVect,latVect);
uniImag = zeros(uniHeight,uniWidth,3);

% Fast Interpolation to uniform grid
uniImag = myTurboInterp2(lonMesh,latMesh,imag,uniLonMesh,uniLatMesh);

if nargin <= 1 % plot map
    % display image
    hold(axHandle, 'on');
    cax = caxis;
    h = image(lonVect,latVect,uniImag, 'Parent', axHandle);
    caxis(cax); % Preserve caxis that is sometimes changed by the call
to image()
    set(axHandle,'YDir','Normal')
    set(h,'tag','gmap')
    set(h,'AlphaData',alphaData)

    % add a dummy image to allow pan/zoom out to x2 of the image extent
    h_tmp = image(lonVect([1 end]),latVect([1
end]),zeros(2),'Visible','off','Parent', axHandle);
    set(h_tmp,'tag','gmap')

    uistack(h,'bottom') % move map to bottom (so it doesn't hide
previously drawn annotations)
    axis(axHandle, curAxis) % restore original zoom
    if nargin == 1
        varargout{1} = h;
    end

    % if auto-refresh mode - override zoom callback to allow automatic

```

```

% refresh of map upon zoom actions.
figHandle = axHandle;
while ~strcmpi(get(figHandle, 'Type'), 'figure')
    % Recursively search for parent figure in case axes are in a
panel
    figHandle = get(figHandle, 'Parent');
end

zoomHandle = zoom(axHandle);
panHandle = pan(figHandle); % This isn't ideal, doesn't work for
contained axis
if autoRefresh
    set(zoomHandle, 'ActionPostCallback', @update_google_map);
    set(panHandle, 'ActionPostCallback', @update_google_map);
else % disable zoom override
    set(zoomHandle, 'ActionPostCallback', []);
    set(panHandle, 'ActionPostCallback', []);
end

% set callback for figure resize function, to update extents if
figure
% is stretched.
if figureResizeUpdate && isempty(get(figHandle, 'ResizeFcn'))
    % set only if not already set by someone else
    set(figHandle, 'ResizeFcn', @update_google_map_fig);
end

% set callback properties
set(h, 'ButtonDownFcn', bd_callback);
else % don't plot, only return map
    varargout{1} = lonVect;
    varargout{2} = latVect;
    varargout{3} = uniImag;
end

% Coordinate transformation functions

function [lon,lat] = metersToLatLon(x,y)
% Converts XY point from Spherical Mercator EPSG:900913 to lat/lon in
WGS84 Datum
originShift = 2 * pi * 6378137 / 2.0; % 20037508.342789244
lon = (x ./ originShift) * 180;
lat = (y ./ originShift) * 180;
lat = 180 / pi * (2 * atan( exp( lat * pi / 180)) - pi / 2);

function [x,y] = latLonToMeters(lat, lon )
% Converts given lat/lon in WGS84 Datum to XY in Spherical Mercator
EPSG:900913"
originShift = 2 * pi * 6378137 / 2.0; % 20037508.342789244
x = lon * originShift / 180;
y = log(tan((90 + lat) * pi / 360 )) / (pi / 180);
y = y * originShift / 180;

```

```

function ZI = myTurboInterp2(X,Y,Z,XI,YI)
% An extremely fast nearest neighbour 2D interpolation, assuming both
input
% and output grids consist only of squares, meaning:
% - uniform X for each column
% - uniform Y for each row
XI = XI(1,:);
X = X(1,:);
YI = YI(:,1);
Y = Y(:,1);

xiPos = nan*ones(size(XI));
xLen = length(X);
yiPos = nan*ones(size(YI));
yLen = length(Y);
% find x conversion
xPos = 1;
for idx = 1:length(xiPos)
    if XI(idx) >= X(1) && XI(idx) <= X(end)
        while xPos < xLen && X(xPos+1)<XI(idx)
            xPos = xPos + 1;
        end
        diffs = abs(X(xPos:xPos+1)-XI(idx));
        if diffs(1) < diffs(2)
            xiPos(idx) = xPos;
        else
            xiPos(idx) = xPos + 1;
        end
    end
end
% find y conversion
yPos = 1;
for idx = 1:length(yiPos)
    if YI(idx) <= Y(1) && YI(idx) >= Y(end)
        while yPos < yLen && Y(yPos+1)>YI(idx)
            yPos = yPos + 1;
        end
        diffs = abs(Y(yPos:yPos+1)-YI(idx));
        if diffs(1) < diffs(2)
            yiPos(idx) = yPos;
        else
            yiPos(idx) = yPos + 1;
        end
    end
end
ZI = Z(yiPos,xiPos,:);

function update_google_map(obj, evd)
% callback function for auto-refresh
drawnow;
try
    axHandle = evd.Axes;
catch ex
    % Event doesn't contain the correct axes. Panic!
    axHandle = gca;
end

```



```

end
ud = get(axHandle, 'UserData');
if isfield(ud, 'gmap_params')
    params = ud.gmap_params;
    plot_google_map(params{:});
end

function update_google_map_fig(obj, evd)
% callback function for auto-refresh
drawnow;
axes_objs = findobj(get(gcf, 'children'), 'type', 'axes');
for idx = 1:length(axes_objs)
    if ~isempty(findobj(get(axes_objs(idx), 'children'), 'tag', 'gmap'));
        ud = get(axes_objs(idx), 'UserData');
        if isfield(ud, 'gmap_params')
            params = ud.gmap_params;
        else
            params = {};
        end

        % Add axes to inputs if needed
        if ~sum(strcmpi(params, 'Axis'))
            params = [params, {'Axis', axes_objs(idx)}];
        end
        plot_google_map(params{:});
    end
end
end

```

Draw Ellipse

```
function h=ellipse(ra,rb,ang,x0,y0,C,Nb)
% Ellipse adds ellipses to the current plot
%
% ELLIPSE(ra,rb,ang,x0,y0) adds an ellipse with semimajor axis of ra,
% a semimajor axis of radius rb, a semimajor axis of ang, centered at
% the point x0,y0.
%
% The length of ra, rb, and ang should be the same.
% If ra is a vector of length L and x0,y0 scalars, L ellipses
% are added at point x0,y0.
% If ra is a scalar and x0,y0 vectors of length M, M ellipse are with
the same
% radii are added at the points x0,y0.
% If ra, x0, y0 are vectors of the same length L=M, M ellipses are
added.
% If ra is a vector of length L and x0, y0 are vectors of length
% M~L, L*M ellipses are added, at each point x0,y0, L ellipses of
radius ra.
%
% ELLIPSE(ra,rb,ang,x0,y0,C)
% adds ellipses of color C. C may be a string ('r','b',...) or the RGB
value.
% If no color is specified, it makes automatic use of the colors
specified by
% the axes ColorOrder property. For several circles C may be a vector.
%
% ELLIPSE(ra,rb,ang,x0,y0,C,Nb), Nb specifies the number of points
% used to draw the ellipse. The default value is 300. Nb may be used
% for each ellipse individually.
%
% h=ELLIPSE(...) returns the handles to the ellipses.
%
% as a sample of how ellipse works, the following produces a red
ellipse
% tipped up at a 45 deg axis from the x axis
% ellipse(1,2,pi/8,1,1,'r')
%
% note that if ra=rb, ELLIPSE plots a circle
%

% written by D.G. Long, Brigham Young University, based on the
% CIRCLES.m original
% written by Peter Blattner, Institute of Microtechnology, University
of
% Neuchatel, Switzerland, blattner@imt.unine.ch

% Check the number of input arguments

if nargin<1,
    ra=[];
end;
if nargin<2,
```

```

    rb=[];
end;
if nargin<3,
    ang=[];
end;

%if nargin==1,
% error('Not enough arguments');
%end;

if nargin<5,
    x0=[];
    y0=[];
end;

if nargin<6,
    C=[];
end

if nargin<7,
    Nb=[];
end

% set up the default values

if isempty(ra),ra=1;end;
if isempty(rb),rb=1;end;
if isempty(ang),ang=0;end;
if isempty(x0),x0=0;end;
if isempty(y0),y0=0;end;
if isempty(Nb),Nb=300;end;
if isempty(C),C=get(gca,'colororder');end;

% work on the variable sizes

x0=x0(:);
y0=y0(:);
ra=ra(:);
rb=rb(:);
ang=ang(:);
Nb=Nb(:);

if isstr(C),C=C(:);end;

if length(ra)~=length(rb),
    error('length(ra)~=length(rb)');
end;
if length(x0)~=length(y0),
    error('length(x0)~=length(y0)');
end;

% how many inscribed ellipses are plotted

if length(ra)~=length(x0)

```

```

    maxk=length(ra)*length(x0);
else
    maxk=length(ra);
end;

% drawing loop

for k=1:maxk

    if length(x0)==1
        xpos=x0;
        ypos=y0;
        radm=ra(k);
        radn=rb(k);
        if length(ang)==1
            an=ang;
        else
            an=ang(k);
        end;
    elseif length(ra)==1
        xpos=x0(k);
        ypos=y0(k);
        radm=ra;
        radn=rb;
        an=ang;
    elseif length(x0)==length(ra)
        xpos=x0(k);
        ypos=y0(k);
        radm=ra(k);
        radn=rb(k);
        an=ang(k)
    else
        rada=ra(fix((k-1)/size(x0,1))+1);
        radb=rb(fix((k-1)/size(x0,1))+1);
        an=ang(fix((k-1)/size(x0,1))+1);
        xpos=x0(rem(k-1,size(x0,1))+1);
        ypos=y0(rem(k-1,size(y0,1))+1);
    end;

    co=cos(an);
    si=sin(an);
    the=linspace(0,2*pi,Nb(rem(k-1,size(Nb,1))+1,:)+1);
    % x=radm*cos(the)*co-si*radn*sin(the)+xpos;
    % y=radm*cos(the)*si+co*radn*sin(the)+ypos;
    h(k)=line(radm*cos(the)*co-
    si*radn*sin(the)+xpos,radm*cos(the)*si+co*radn*sin(the)+ypos);
    set(h(k),'color',C(rem(k-1,size(C,1))+1,:));

end;

```

Overlapping Analysis

```
clear all
close all
clc

counter=0;
all_data_overlap=[];
%% Read in data
addpath('mat Files')
for ii=1:4
    for kk=1:7
        if 3*ii<10
            Date= ['2015.0', num2str(3*ii), '.0', num2str(kk)]
        else
            Date= ['2015.', num2str(3*ii), '.0', num2str(kk)]
        end
    end

    Fargo_data=load(['Short_Merged_', Date, '_', 'Fargo.mat']);
    Finley_data=load(['Short_Merged_', Date, '_', 'Finley.mat']);

    %% Drop Outs

    %Load Data
    Fargo_do=Fargo_data.data_do;
    Finley_do=Finley_data.data_do;

    %Find all the ID numbers that experienced drop outs
    Finley_unique_do=unique(Finley_do(:,2), 'rows', 'sorted');
    Fargo_unique_do=unique(Fargo_do(:,2), 'rows', 'sorted');

    %Compare Finley to Fargo data
    do_logical_same_ID_Fargo=ismember(Fargo_do(:,2), Finley_unique_do);
    do_Fargo_same_ID=Fargo_do(do_logical_same_ID_Fargo==1, :);
    Fargo_do_size=size(do_Fargo_same_ID);

    %Compare Fargo to Finley data
    do_logical_same_ID_Finley=ismember(Finley_do(:,2), Fargo_unique_do);
    do_Finley_same_ID=Finley_do(do_logical_same_ID_Finley==1, :);
    Finley_do_size=size(do_Finley_same_ID);

    %Check for overlap at same time instance => transponder failure
    do_overlap_summary=[];
    for aa=1:Finley_do_size(1)
        for bb=1:Fargo_do_size(1)
            if do_Finley_same_ID(aa,2)==do_Fargo_same_ID(bb,2)
                if (abs(do_Finley_same_ID(aa,4) -
                    do_Fargo_same_ID(bb,4))<17
                    do_overlap_summary=[do_overlap_summary;
                    do_Fargo_same_ID(bb,:), 00000000000000000000, do_Finley_same_ID(aa,:)];
                end
            end
        end
    end
end
```

```

%Summary of results for drop outs
instances_transponder_do=size(do_overlap_summary);
instances_transponder_do=instances_transponder_do(1); %transponder
failure instances
if length(do_overlap_summary)>=1
    planes_transponder_do=length(unique(do_overlap_summary(:,2)));
%number of planes with transponder failures
else
    planes_transponder_do=0;
end

planes_Finley_radar_do=length(unique(do_Finley_same_ID(:,2)));%number
of planes with radar failures at Finley
planes_Fargo_radar_do=length(unique(do_Fargo_same_ID(:,2)));%number of
planes with radar failures at Fargo
percentage_radar_do=((planes_Finley_radar_do(1)-
planes_transponder_do(1))/planes_Finley_radar_do(1)); %percentage of
radar failures
percentage_transponder_do=((planes_transponder_do(1))/planes_Finley_rad
ar_do(1)); %percentage of transponder failures

%% Outliers

%Load data
Fargo_out=Fargo_data.outliers;
Finley_out=Finley_data.outliers;

%Find all the ID numbers that experienced outliers
Finley_unique_out=unique(Finley_out(:,2),'rows','sorted');
Fargo_unique_out=unique(Fargo_out(:,2),'rows','sorted');

%Compare Finley to Fargo data
out_logical_same_ID_Fargo=ismember(Fargo_out(:,2),Finley_unique_out);
out_Fargo_same_ID=Fargo_out(out_logical_same_ID_Fargo==1,:);
Fargo_out_size=size(out_Fargo_same_ID);

%Compare Fargo to Finley data
out_logical_same_ID_Finley=ismember(Finley_out(:,2),Fargo_unique_out);
out_Finley_same_ID=Finley_out(out_logical_same_ID_Finley==1,:);
Finley_out_size=size(out_Finley_same_ID);

%Check for overlap at same time instance => transponder failure
out_overlap_summary=[];
for aa=1:Finley_out_size(1)
    for bb=1:Fargo_out_size(1)
        if out_Finley_same_ID(aa,2)==out_Fargo_same_ID(bb,2)
            if (abs(out_Finley_same_ID(aa,4)-
out_Fargo_same_ID(bb,4))<17
                out_overlap_summary=[out_overlap_summary;
out_Fargo_same_ID(bb,:),00000000000000000000, out_Finley_same_ID(aa,:)];
            end
        end
    end
end
end
end

```

```

%Summary of results for outliers
instances_transponder_out=size(out_overlap_summary);
instances_transponder_out=instances_transponder_out(1); %transponder
failure instances
if length(out_overlap_summary)>=1
    planes_transponder_out=length(unique(out_overlap_summary(:,2)));
%number of planes with transponder failures
else
    planes_transponder_out=0;
end

planes_Finley_radar_out=length(unique(out_Finley_same_ID(:,2)));%number
of planes with radar failures at Finley
planes_Fargo_radar_out=length(unique(out_Fargo_same_ID(:,2)));%number
of planes with radar failures at Fargo
percentage_radar_out=((planes_Finley_radar_out(1)-
planes_transponder_out(1))/planes_Finley_radar_out(1)); %percentage of
radar failures
percentage_transponder_out=((planes_transponder_out(1))/planes_Finley_r
adar_out(1)); %percentage of transponder failures

%% Prolonged Altitude Failure

if length(Fargo_data.p_alt_failure)<1
    instances_transponder_paf=0;
    planes_transponder_paf=0;
    planes_Finley_radar_paf=0;
    planes_Fargo_radar_paf=0;
    percentage_radar_paf=0;
    percentage_transponder_paf=0;
else

%Load data
Fargo_paf=cell2mat(Fargo_data.p_alt_failure);
Finley_paf=cell2mat(Finley_data.p_alt_failure);

%Find all the ID numbers that experienced prolonged altitude failures
Finley_unique_paf=unique(Finley_paf(:,2), 'rows', 'sorted');
Fargo_unique_paf=unique(Fargo_paf(:,2), 'rows', 'sorted');

%Compare Finley to Fargo data
paf_logical_same_ID_Fargo=ismember(Fargo_paf(:,2),Finley_unique_paf);
paf_Fargo_same_ID=Fargo_paf(paf_logical_same_ID_Fargo==1,:);
Fargo_paf_size=size(paf_Fargo_same_ID);

%Compare Fargo to Finley data
paf_logical_same_ID_Finley=ismember(Finley_paf(:,2),Fargo_unique_paf);
paf_Finley_same_ID=Finley_paf(paf_logical_same_ID_Finley==1,:);
Finley_paf_size=size(paf_Finley_same_ID);

%Check for overlap at same time instance => transponder failure
paf_overlap_summary=[];
for aa=1:Finley_paf_size(1)
    for bb=1:Fargo_paf_size(1)

```

```

        if paf_Finley_same_ID(aa,2)==paf_Fargo_same_ID(bb,2)
            if (abs(paf_Finley_same_ID(aa,4)-
paf_Fargo_same_ID(bb,4))<17
                paf_overlap_summary=[paf_overlap_summary;
paf_Fargo_same_ID(bb,:),00000000000000000000, paf_Finley_same_ID(aa,:)];
            end
        end
    end
end

%Summary of results for prolonged alititude failure
instances_transponder_paf=size(paf_overlap_summary);
instances_transponder_paf=instances_transponder_paf(1); %transponder
failure instances
if length(paf_overlap_summary)>=1
    planes_transponder_paf=length(unique(paf_overlap_summary(:,2)));
%number of planes with transponder failures
else
    planes_transponder_paf=0;
end
planes_Finley_radar_paf=length(unique(paf_Finley_same_ID(:,2)));%number
of planes with radar failures at Finley
planes_Fargo_radar_paf=length(unique(paf_Fargo_same_ID(:,2)));%number
of planes with radar failures at Fargo
percentage_radar_paf=((planes_Finley_radar_paf(1)-
planes_transponder_paf(1))/planes_Finley_radar_paf(1)); %percentage of
radar failures
percentage_transponder_paf=((planes_transponder_paf(1))/planes_Finley_r
adar_paf(1)); %percentage of transponder failures
end

%% Repeated Data

%Load Data
Fargo_rd=Fargo_data.repeated_data;
Finley_rd=Finley_data.repeated_data;

%Find all the ID numbers that experienced repeated data
Finley_unique_rd=unique(Finley_rd(:,2), 'rows', 'sorted');
Fargo_unique_rd=unique(Fargo_rd(:,2), 'rows', 'sorted');

%Compare Finley to Fargo data
rd_logical_same_ID_Fargo=ismember(Fargo_rd(:,2),Finley_unique_rd);
rd_Fargo_same_ID=Fargo_rd(rd_logical_same_ID_Fargo==1,:);
Fargo_rd_size=size(rd_Fargo_same_ID);

%Compare Fargo to Finley data
rd_logical_same_ID_Finley=ismember(Finley_rd(:,2),Fargo_unique_rd);
rd_Finley_same_ID=Finley_rd(rd_logical_same_ID_Finley==1,:);
Finley_rd_size=size(rd_Finley_same_ID);

%Check for overlap at same time instance => transponder failure
rd_overlap_summary=[];
for aa=1:Finley_rd_size(1)
    for bb=1:Fargo_rd_size(1)

```



```

        if rd_Finley_same_ID(aa,2)==rd_Fargo_same_ID(bb,2)
            if (abs(rd_Finley_same_ID(aa,4)-
rd_Fargo_same_ID(bb,4))<17
                rd_overlap_summary=[rd_overlap_summary;
rd_Fargo_same_ID(bb,:),00000000000000000000, rd_Finley_same_ID(aa,:)];
            end
        end
    end
end

%Summary of results for repeated data
instances_transponder_rd=size(rd_overlap_summary);
instances_transponder_rd=instances_transponder_rd(1); %transponder
failure instances
if length(rd_overlap_summary)>=1
    planes_transponder_rd=length(unique(rd_overlap_summary(:,2)));
%number of planes with transponder failures
else
    planes_transponder_rd=0;
end
planes_Finley_radar_rd=length(unique(rd_Finley_same_ID(:,2)));%number
of planes with radar failures at Finley
planes_Fargo_radar_rd=length(unique(rd_Fargo_same_ID(:,2)));%number of
planes with radar failures at Fargo
percentage_radar_rd=((planes_Finley_radar_rd(1)-
planes_transponder_rd(1))/planes_Finley_radar_rd(1)); %percentage of
radar failures
percentage_transponder_rd=((planes_transponder_rd(1))/planes_Finley_rad
ar_rd(1)); %percentage of transponder failures

%% Multiple Aircraft

%Load Data
Fargo_ma_pre=Fargo_data.multiple_aircraft;
Finley_ma_pre=Finley_data.multiple_aircraft;

%Eliminate ID numbers of 0 and 1200
Fargo_ma=[];
for mm=1:length(Fargo_ma_pre)
    if Fargo_ma_pre(mm,2) ~=0
        if Fargo_ma_pre(mm,2) ~=1200
            Fargo_ma=[Fargo_ma;Fargo_ma_pre(mm,:)];
        end
    end
end

Finley_ma=[];
for mm=1:length(Finley_ma_pre)
    if Finley_ma_pre(mm,2) ~=0
        if Finley_ma_pre(mm,2) ~=1200
            Finley_ma=[Finley_ma;Finley_ma_pre(mm,:)];
        end
    end
end

if length(Fargo_ma)>=1

```

```

if length(Finley_ma)>=1
    %Find all the ID numbers that experienced multiple aircraft
    Finley_unique_ma=unique(Finley_ma(:,2),'rows','sorted');
    Fargo_unique_ma=unique(Fargo_ma(:,2),'rows','sorted');

    %Compare Finley to Fargo data

ma_logical_same_ID_Fargo=ismember(Fargo_ma(:,2),Finley_unique_ma);
ma_Fargo_same_ID=Fargo_ma(ma_logical_same_ID_Fargo==1,:);
Fargo_ma_size=size(ma_Fargo_same_ID);

    %Compare Fargo to Finley data

ma_logical_same_ID_Finley=ismember(Finley_ma(:,2),Fargo_unique_ma);
ma_Finley_same_ID=Finley_ma(ma_logical_same_ID_Finley==1,:);
Finley_ma_size=size(ma_Finley_same_ID);

    %Check for overlap at same time instance => transponder failure
ma_overlap_summary=[];
for aa=1:Finley_ma_size(1)
    for bb=1:Fargo_ma_size(1)
        if ma_Finley_same_ID(aa,2)==ma_Fargo_same_ID(bb,2)
            if (abs(ma_Finley_same_ID(aa,4)-
ma_Fargo_same_ID(bb,4))<17
                ma_overlap_summary=[ma_overlap_summary;
ma_Fargo_same_ID(bb,:),0000000000000000000, ma_Finley_same_ID(aa,:)];
            end
        end
    end
end

    %Summary of results for multiple aircraft
instances_transponder_ma=size(ma_overlap_summary);
instances_transponder_ma=instances_transponder_ma(1);
%transponder failure instances
if length(ma_overlap_summary)>=1

planes_transponder_ma=length(unique(ma_overlap_summary(:,2))); %number
of planes with transponder failures
else
    planes_transponder_ma=0;
end

planes_Finley_radar_ma=length(unique(ma_Finley_same_ID(:,2)));%number
of planes with radar failures at Finley

planes_Fargo_radar_ma=length(unique(ma_Fargo_same_ID(:,2)));%number of
planes with radar failures at Fargo
percentage_radar_ma=((planes_Finley_radar_ma(1)-
planes_transponder_ma(1))/planes_Finley_radar_ma(1)); %percentage of
radar failures

percentage_transponder_ma=((planes_transponder_ma(1))/planes_Finley_radar_ma(1)); %percentage of transponder failures

```

```

    end
else
    instances_transponder_ma=0;
    planes_transponder_ma=0;
    planes_Finley_radar_ma=0;
    planes_Fargo_radar_ma=0;
    percentage_radar_ma=0;
    percentage_transponder_ma=0;
end

%% Store all data
counter=counter+1;

all_data_overlap=[all_data_overlap; planes_Finley_radar_do,
planes_Fargo_radar_do, planes_transponder_do, instances_transponder_do,
percentage_radar_do, percentage_transponder_do,
planes_Finley_radar_out, planes_Fargo_radar_out,
planes_transponder_out, instances_transponder_out,
percentage_radar_out, percentage_transponder_out,
planes_Finley_radar_paf, planes_Fargo_radar_paf,
planes_transponder_paf, instances_transponder_paf,
percentage_radar_paf, percentage_transponder_paf,
planes_Finley_radar_rd, planes_Fargo_radar_rd, planes_transponder_rd,
instances_transponder_rd, percentage_radar_rd,
percentage_transponder_rd, planes_Finley_radar_ma,
planes_Fargo_radar_ma, planes_transponder_ma, instances_transponder_ma,
percentage_radar_ma, percentage_transponder_ma];

if counter==28
    xlswrite('Overlapping Analysis.xlsx', all_data_overlap)
end

%% Save .mat file
matFileDir= 'C:\Users\nicholas.allen\OneDrive - North Dakota
University System\UAS Research\Radars Analysis\Overlapping Mat Files\';
save([matFileDir, Date, '.mat'])
disp('done data write')
end
end

```

APPENDIX C: RADAR DATA ANOMALIES SUMMARY

Aircraft Count/Location Summary

Fargo Aircraft Count/Location			
Location	Month	Day	Number of Aircraft
Fargo	3	1	328
Fargo	3	2	392
Fargo	3	3	322
Fargo	3	4	472
Fargo	3	5	468
Fargo	3	6	586
Fargo	3	7	463
Fargo	6	1	512
Fargo	6	2	386
Fargo	6	3	438
Fargo	6	4	569
Fargo	6	5	565
Fargo	6	6	382
Fargo	6	7	360
Fargo	9	1	564
Fargo	9	2	551
Fargo	9	3	458
Fargo	9	4	464
Fargo	9	5	369
Fargo	9	6	231
Fargo	9	7	446
Fargo	12	1	313
Fargo	12	2	492
Fargo	12	3	549
Fargo	12	4	506
Fargo	12	5	457
Fargo	12	6	442
Fargo	12	7	511

Finley Aircraft Count/Location			
Location	Month	Day	Number of Aircraft
Finley	3	1	987
Finley	3	2	1149
Finley	3	3	1008
Finley	3	4	1203
Finley	3	5	1284
Finley	3	6	1298
Finley	3	7	1118
Finley	6	1	1313
Finley	6	2	1283
Finley	6	3	1264
Finley	6	4	1464
Finley	6	5	1512
Finley	6	6	1254
Finley	6	7	1209
Finley	9	1	1503
Finley	9	2	1515
Finley	9	3	1515
Finley	9	4	1417
Finley	9	5	1065
Finley	9	6	1035
Finley	9	7	1274
Finley	12	1	1133
Finley	12	2	1433
Finley	12	3	1487
Finley	12	4	1364
Finley	12	5	1126
Finley	12	6	1208
Finley	12	7	1261

Drop Outs

Fargo Drop Outs												
Date	# of Aircraft	<10	10-15	15-20	20-25	25-30	30-60	>60	Avg.	Min	Max	# of Instances
3/1/2015	93	136	36	4	10	6	24	28	27.28	5.50	231.66	244
3/2/2015	131	210	69	3	13	9	36	48	26.82	8.76	260.54	388
3/3/2015	77	120	38	7	11	16	26	24	25.69	6.14	255.80	242
3/4/2015	169	291	87	13	25	13	69	57	26.23	9.08	299.26	555
3/5/2015	174	368	129	15	21	19	59	82	27.82	9.40	299.01	693
3/6/2015	218	491	105	20	24	20	58	69	23.49	5.50	299.20	787
3/7/2015	168	317	109	12	32	25	64	64	26.74	5.60	299.20	623
6/1/2015	163	222	80	8	24	14	51	38	25.36	5.54	289.65	437
6/2/2015	125	301	22	3	4	9	21	8	13.23	5.50	281.28	368
6/3/2015	162	435	142	11	37	24	84	76	23.61	5.70	284.67	809
6/4/2015	239	573	176	20	44	37	86	92	25.49	8.70	299.26	1028
6/5/2015	216	620	102	10	24	15	56	59	20.13	5.50	285.39	886
6/6/2015	105	166	54	4	7	13	44	49	31.16	9.29	294.25	337
6/7/2015	101	155	45	2	13	5	27	24	23.72	9.16	251.09	271
9/1/2015	244	771	141	19	50	28	68	81	20.34	5.50	284.93	1158
9/2/2015	212	427	118	11	25	20	66	66	25.11	5.60	284.92	733
9/3/2015	168	354	46	8	9	6	38	24	17.27	5.50	282.80	485
9/4/2015	127	190	58	7	16	11	40	29	25.01	9.44	260.71	351
9/5/2015	140	307	35	3	10	6	18	19	15.49	5.50	279.81	398
9/6/2015	60	91	26	2	6	6	13	10	21.45	9.14	294.32	154
9/7/2015	176	543	117	10	33	22	64	58	21.29	5.50	299.29	847
12/1/2015	53	93	8	1	0	3	5	2	12.01	5.50	254.32	112
12/2/2015	172	311	83	15	33	16	64	53	26.24	8.04	289.62	575
12/3/2015	192	345	105	19	19	27	75	59	26.05	5.51	299.25	649
12/4/2015	190	306	87	9	34	18	108	48	26.26	9.36	284.84	610
12/5/2015	180	415	71	12	18	11	54	32	19.26	5.50	265.44	613
12/6/2015	127	262	94	14	14	26	66	58	28.67	5.50	299.20	534
12/7/2015	157	296	91	11	20	18	61	67	29.03	5.57	299.30	564

Finley Drop Outs												
Date	# of Aircraft	<24 sec	24-36 sec	36-48 sec	48-60 sec	60-90 sec	90-120 sec	>120 sec	Avg.	Min	Max	# of Instances
3/1/2015	518	361	463	48	11	49	12	24	33.16	22.12	347.98	968
3/2/2015	715	505	673	114	27	88	38	77	38.93	22.81	395.63	1522
3/3/2015	619	1289	393	52	11	30	10	25	23.02	13.00	396.09	1810
3/4/2015	668	568	652	100	18	60	21	67	36.83	13.13	372.10	1486
3/5/2015	615	434	555	123	24	78	29	125	47.95	23.31	396.07	1368
3/6/2015	596	374	563	133	30	114	48	183	57.59	13.20	396.24	1445
3/7/2015	482	260	418	81	19	79	33	121	55.41	23.59	396.29	1011
6/1/2015	949	1188	1301	244	49	166	53	73	34.19	22.45	395.69	3074
6/2/2015	854	858	989	254	39	168	46	113	38.41	17.76	395.86	2467
6/3/2015	820	654	843	176	30	138	42	65	38.25	13.42	383.94	1948
6/4/2015	1042	1142	1566	351	69	297	116	267	45.02	22.35	396.57	3808
6/5/2015	1139	1320	1696	405	69	335	131	254	43.52	23.58	396.40	4210
6/6/2015	798	690	851	154	33	104	41	73	37.22	13.06	395.71	1946
6/7/2015	949	1103	1348	254	58	197	99	171	40.74	22.67	396.24	3230
9/1/2015	1234	1961	2396	620	108	436	176	368	42.97	13.00	396.34	6065
9/2/2015	1299	2288	3021	755	117	486	210	392	41.50	13.02	396.44	7269
9/3/2015	1323	2809	3576	816	141	629	214	466	41.20	13.28	396.36	8651
9/4/2015	1160	1758	2397	587	82	406	159	254	40.10	13.17	396.63	5643
9/5/2015	871	1752	1530	264	50	195	87	95	32.36	13.00	383.55	3973
9/6/2015	890	1370	1541	310	43	193	72	94	34.27	13.46	359.80	3623
9/7/2015	1033	1538	1720	315	81	265	112	295	43.59	13.17	395.97	4326
12/1/2015	723	634	810	155	44	93	49	140	44.92	13.04	399.17	1925
12/2/2015	1126	1299	1717	548	113	496	165	399	50.11	13.06	396.50	4737
12/3/2015	1044	1896	1422	413	119	402	194	343	44.92	13.00	397.92	4789
12/4/2015	1029	1070	1614	442	73	428	184	410	52.26	23.63	396.19	4221
12/5/2015	626	413	552	122	33	106	29	126	48.55	13.01	384.21	1381
12/6/2015	766	531	787	219	40	254	113	232	55.36	23.70	395.95	2176
12/7/2015	904	888	1066	228	52	206	79	219	46.29	22.89	396.41	2738

Altitude Outliers

Fargo Altitude Outliers		
Date	Number of Aircraft	Number of Instances
3/1/2015	31	54
3/2/2015	44	69
3/3/2015	34	49
3/4/2015	74	111
3/5/2015	82	135
3/6/2015	71	140
3/7/2015	83	166
6/1/2015	53	83
6/2/2015	19	30
6/3/2015	94	213
6/4/2015	107	206
6/5/2015	72	165
6/6/2015	41	67
6/7/2015	40	58
9/1/2015	88	199
9/2/2015	101	205
9/3/2015	34	64
9/4/2015	54	105
9/5/2015	27	34
9/6/2015	13	13
9/7/2015	67	173
12/1/2015	11	29
12/2/2015	78	157
12/3/2015	111	209
12/4/2015	74	133
12/5/2015	58	107
12/6/2015	60	103
12/7/2015	89	177

Finley Altitude Outliers		
Date	Number of Aircraft	Number of Instances
3/1/2015	110	135
3/2/2015	230	317
3/3/2015	142	218
3/4/2015	189	293
3/5/2015	223	343
3/6/2015	262	484
3/7/2015	177	244
6/1/2015	240	364
6/2/2015	240	347
6/3/2015	247	312
6/4/2015	406	662
6/5/2015	425	726
6/6/2015	229	279
6/7/2015	339	525
9/1/2015	493	928
9/2/2015	553	1109
9/3/2015	608	1191
9/4/2015	446	749
9/5/2015	286	398
9/6/2015	285	409
9/7/2015	423	744
12/1/2015	214	349
12/2/2015	510	1020
12/3/2015	453	1038
12/4/2015	489	933
12/5/2015	226	378
12/6/2015	336	599
12/7/2015	337	580

Prolonged Altitude Failures

Fargo Prolonged Altitude Failures											
Date	<10	10-15	15-20	20-25	25-30	30-60	>60	Avg.	Min	Max	Number of Aircraft
3/1/2015	1	0	0	0	0	0	0	9.67	9.67	9.67	1
3/2/2015	0	0	0	0	0	0	0	0.00	0.00	0.00	0
3/3/2015	0	0	0	0	1	0	5	134.07	28.98	357.22	6
3/4/2015	3	0	1	0	0	0	3	83.47	4.86	333.53	7
3/5/2015	1	1	0	0	0	0	1	51.59	9.66	130.65	3
3/6/2015	3	0	0	1	0	0	6	240.76	9.49	1056.26	10
3/7/2015	1	1	0	0	2	2	12	398.94	9.70	1159.05	18
6/1/2015	1	0	0	0	0	3	6	436.38	9.75	2471.76	10
6/2/2015	0	0	0	0	0	0	1	72.30	72.30	72.30	1
6/3/2015	0	1	0	0	0	1	1	166.19	14.44	440.71	3
6/4/2015	1	0	0	0	0	1	3	80.17	9.66	135.23	5
6/5/2015	1	0	0	0	0	1	7	174.86	9.77	694.99	9
6/6/2015	1	0	0	1	0	0	2	83.26	9.66	154.48	4
6/7/2015	1	1	0	0	0	1	0	27.39	9.60	58.11	3
9/1/2015	1	0	0	0	0	1	5	112.61	9.90	260.48	7
9/2/2015	4	1	0	1	0	3	2	44.34	9.54	159.39	11
9/3/2015	0	1	0	0	0	2	3	152.86	14.44	637.17	6
9/4/2015	0	0	0	1	0	3	4	89.27	24.14	236.56	8
9/5/2015	1	0	0	0	0	1	1	258.88	9.61	728.42	3
9/6/2015	0	0	0	0	0	0	0	0.00	0.00	0.00	0
9/7/2015	0	0	0	0	0	0	5	96.49	72.12	168.92	5
12/1/2015	0	0	0	0	0	0	0	0.00	0.00	0.00	0
12/2/2015	0	1	0	1	1	1	3	164.09	14.49	584.96	7
12/3/2015	1	1	3	0	0	5	6	112.83	9.61	713.04	16
12/4/2015	0	0	0	0	1	2	6	179.78	28.82	526.33	9
12/5/2015	2	1	0	2	0	0	0	16.38	9.55	24.11	5
12/6/2015	0	0	0	1	0	1	6	155.49	24.17	432.82	8
12/7/2015	0	0	0	0	3	2	12	157.57	28.82	405.03	17

Finley Prolonged Altitude Failures											
Date	<24 sec	24-36 sec	36-48 sec	48-60 sec	60-90 sec	90-120 sec	>120 sec	Avg.	Min	Max	Number of Instances
3/1/2015	0	1	0	0	0	0	0	35.98	35.98	35.98	1
3/2/2015	1	1	0	1	3	0	2	265.53	23.98	984.93	8
3/3/2015	1	0	0	0	0	1	1	100.05	23.96	156.38	3
3/4/2015	0	1	0	0	0	1	3	148.78	35.92	264.36	5
3/5/2015	1	1	1	0	1	1	2	101.22	23.92	252.50	7
3/6/2015	1	9	5	2	3	0	4	114.01	23.89	888.48	24
3/7/2015	1	0	0	0	1	0	2	482.60	23.89	1546.65	4
6/1/2015	1	0	0	0	0	0	3	609.29	23.86	2053.38	4
6/2/2015	2	1	1	1	0	0	7	155.07	23.98	419.90	12
6/3/2015	0	0	0	0	1	0	2	238.10	72.06	498.18	3
6/4/2015	2	0	0	1	1	1	3	129.03	23.95	335.73	8
6/5/2015	2	0	0	0	1	0	7	169.19	23.98	312.32	10
6/6/2015	0	0	0	0	0	0	2	276.21	264.24	288.19	2
6/7/2015	0	0	0	0	0	0	1	156.15	156.15	156.15	1
9/1/2015	1	1	2	1	1	2	4	110.96	23.98	443.70	12
9/2/2015	3	0	0	3	1	0	4	192.02	23.92	971.96	11
9/3/2015	0	3	2	0	0	0	1	70.00	24.03	275.61	6
9/4/2015	0	1	0	0	0	0	0	24.17	24.17	24.17	1
9/5/2015	0	1	0	0	0	0	0	24.02	24.02	24.02	1
9/6/2015	1	0	0	0	0	1	0	60.02	12.06	107.97	2
9/7/2015	1	1	0	1	2	0	6	249.85	23.96	660.01	11
12/1/2015	1	1	0	1	0	2	7	232.43	23.79	603.37	12
12/2/2015	1	5	0	3	2	1	7	130.13	23.91	395.97	19
12/3/2015	1	1	1	1	3	4	12	199.84	23.93	612.30	23
12/4/2015	4	0	1	2	1	0	6	150.82	23.85	588.07	14
12/5/2015	1	4	3	3	1	1	2	60.03	23.88	179.98	15
12/6/2015	1	0	1	2	3	2	4	128.27	23.87	347.49	13
12/7/2015	1	5	0	1	2	2	8	150.32	23.94	444.04	19

Multiple Aircraft & Repeated Data

Fargo Multiple Aircraft/ Repeated Data		
Date	Number Multiple Aircraft w/ Same ID Aircraft	Number of Repeated Data Aircraft
3/1/2015	2	5
3/2/2015	2	10
3/3/2015	3	7
3/4/2015	3	14
3/5/2015	3	16
3/6/2015	28	3
3/7/2015	4	22
6/1/2015	2	15
6/2/2015	53	5
6/3/2015	4	12
6/4/2015	2	22
6/5/2015	48	3
6/6/2015	3	6
6/7/2015	2	14
9/1/2015	18	2
9/2/2015	3	19
9/3/2015	24	5
9/4/2015	2	6
9/5/2015	28	4
9/6/2015	2	1
9/7/2015	25	8
12/1/2015	70	1
12/2/2015	4	25
12/3/2015	11	23
12/4/2015	6	14
12/5/2015	24	3
12/6/2015	5	13
12/7/2015	5	19

Finley Multiple Aircraft/ Repeated Data		
Date	Number Multiple Aircraft w/ Same ID Aircraft	Number of Repeated Data Aircraft
3/1/2015	2	15
3/2/2015	5	36
3/3/2015	29	5
3/4/2015	5	40
3/5/2015	5	66
3/6/2015	9	73
3/7/2015	4	55
6/1/2015	4	21
6/2/2015	13	31
6/3/2015	10	30
6/4/2015	16	63
6/5/2015	11	61
6/6/2015	8	24
6/7/2015	10	16
9/1/2015	12	72
9/2/2015	17	79
9/3/2015	8	63
9/4/2015	10	33
9/5/2015	18	7
9/6/2015	6	75
9/7/2015	10	85
12/1/2015	5	11
12/2/2015	18	55
12/3/2015	49	20
12/4/2015	6	57
12/5/2015	6	54
12/6/2015	12	51
12/7/2015	8	58

APPENDIX D: OVERLAPPING ANALYSIS RESULTS

Drop Outs

Date	Total Number of Aircraft	Number of radar Failures	Number of Transponder Failures	Percentage of Radar Failures	Percentage of Transponder Failures
3/1/2015	54	51	3	94%	6%
3/2/2015	97	94	3	97%	3%
3/3/2015	51	46	5	90%	10%
3/4/2015	111	100	11	90%	10%
3/5/2015	107	103	4	96%	4%
3/6/2015	112	105	7	94%	6%
3/7/2015	90	83	7	92%	8%
6/1/2015	109	105	4	96%	4%
6/2/2015	86	81	5	94%	6%
6/3/2015	115	110	5	96%	4%
6/4/2015	169	160	9	95%	5%
6/5/2015	141	133	8	94%	6%
6/6/2015	56	54	2	96%	4%
6/7/2015	87	80	7	92%	8%
9/1/2015	194	174	20	90%	10%
9/2/2015	161	156	5	97%	3%
9/3/2015	143	140	3	98%	2%
9/4/2015	101	95	6	94%	6%
9/5/2015	102	97	5	95%	5%
9/6/2015	48	44	4	92%	8%
9/7/2015	144	122	22	85%	15%
12/1/2015	32	32	0	100%	0%
12/2/2015	118	110	8	93%	7%
12/3/2015	112	103	9	92%	8%
12/4/2015	135	122	13	90%	10%
12/5/2015	94	90	4	96%	4%
12/6/2015	80	64	16	80%	20%
12/7/2015	118	105	13	89%	11%

Altitude Outliers

Date	Total Number of Aircraft	Number of radar Failures	Number of Transponder Failures	Percentage of Radar Failures	Percentage of Transponder Failures
3/1/2015	5	3	2	60%	40%
3/2/2015	16	8	8	50%	50%
3/3/2015	11	4	7	36%	64%
3/4/2015	16	8	8	50%	50%
3/5/2015	25	17	8	68%	32%
3/6/2015	33	11	22	33%	67%
3/7/2015	27	14	13	52%	48%
6/1/2015	8	4	4	50%	50%
6/2/2015	5	3	2	60%	40%
6/3/2015	23	12	11	52%	48%
6/4/2015	51	27	24	53%	47%
6/5/2015	33	13	20	39%	61%
6/6/2015	10	7	3	70%	30%
6/7/2015	15	9	6	60%	40%
9/1/2015	39	17	22	44%	56%
9/2/2015	48	25	23	52%	48%
9/3/2015	16	8	8	50%	50%
9/4/2015	21	14	7	67%	33%
9/5/2015	4	2	2	50%	50%
9/6/2015	3	3	0	100%	0%
9/7/2015	29	15	14	52%	48%
12/1/2015	2	0	2	0%	100%
12/2/2015	36	20	16	56%	44%
12/3/2015	39	20	19	51%	49%
12/4/2015	32	16	16	50%	50%
12/5/2015	18	7	11	39%	61%
12/6/2015	24	13	11	54%	46%
12/7/2015	32	15	17	47%	53%

APPENDIX E: CLIMATE EFFECTS SUMMARY

Fargo

Date	3/1/2015	3/2/2015	3/3/2015	3/4/2015	3/5/2015	3/6/2015	3/7/2015	6/1/2015	6/2/2015	6/3/2015	6/4/2015	6/5/2015	6/6/2015	6/7/2015
Temp. (°F)	high 24	33	28	7	25	40	43	74	80	71	75	76	75	86
	avg 11	17	14	0	7	26	36	63	71	64	66	66	66	71
	low -2	1	0	-7	-11	11	28	52	61	56	56	56	57	55
Dew Point (°F)	high 13	19	22	-6	8	29	31	48	63	62	58	58	68	61
	avg 4	9	5	-10	-2	16	25	44	55	56	54	55	62	56
	low -8	-4	-8	-14	-16	6	22	37	45	53	49	49	51	49
Humidity (%)	high 80	87	81	75	72	80	85	66	73	97	93	93	90	100
	avg 71	63	67	65	59	60	67	52	59	85	73	70	79	66
	low 62	39	53	54	45	40	49	38	45	73	53	46	67	31
Sea Level Press. (in)	high 30.46	30.47	30.14	30.52	30.52	30.15	30.23	30.18	29.89	29.96	30.1	30.22	30.07	29.72
	avg 30.32	30.2	29.9	30.36	30.35	30.05	30.11	30.05	29.79	29.83	30.03	30.15	29.81	29.67
	low 30.22	29.82	29.73	30.14	30.09	29.91	29.88	29.9	29.64	29.66	29.96	30.07	29.65	29.63
Visibility (mi)	high 10	10	10	10	10	10	10	10	10	10	10	10	10	10
	avg 10	10	8	10	10	10	10	10	10	7	10	10	8	8
	low 2	10	2	7	10	10	10	10	10	1	10	10	1	0
Wind (mph)	high 23	32	33	21	24	20	21	29	37	18	20	15	26	23
	avg 11	16	22	13	14	11	11	19	18	13	9	9	12	10
	gust (high) 26	42	40	29	29	23	26	38	45	22	24	22	33	36
Precip. (in)	sum T	0	0.02	0	0	0	0	0	T	0	0	0	0.58	T
Events	Snow		Snow						Rain				Rain	Fog, Rain
Drop Outs	28%	99%	75%	118%	148%	134%	135%	85%	95%	185%	181%	157%	88%	75%
Outliers	16%	18%	15%	24%	29%	24%	36%	16%	8%	49%	36%	29%	18%	16%
Instances/Total Number of Aircraft Observed	Prolonged Altitude Failure	0.3%	0.0%	1.9%	1.5%	1.7%	3.9%	2.0%	0.3%	0.7%	0.9%	1.6%	1.0%	0.8%
	Repeated Data	6%	16%	11%	49%	1%	30%	11%	1%	24%	20%	1%	7%	9%
	Multiple Aircraft	1%	1%	1%	1%	5%	1%	0%	14%	1%	0%	8%	1%	1%

Date	9/1/2015	9/2/2015	9/3/2015	9/4/2015	9/5/2015	9/6/2015	9/7/2015	12/1/2015	12/2/2015	12/3/2015	12/4/2015	12/5/2015	12/6/2015	12/7/2015
high	90	90	93	84	84	79	73	35	29	33	40	40	37	38
avg	74	79	83	76	76	66	61	32	21	23	29	32	29	34
low	57	68	72	74	67	53	49	28	12	12	18	23	20	30
high	70	69	73	73	73	72	56	31	22	25	31	33	28	30
avg	57	66	71	71	70	67	51	30	14	16	23	29	24	28
low	49	63	68	68	66	49	46	24	8	8	14	21	18	24
high	100	90	90	94	94	97	89	92	92	92	84	92	96	82
avg	62	67	70	79	82	78	68	87	80	82	74	84	83	76
low	24	44	49	63	69	58	47	82	68	72	64	76	70	69
high	29.89	29.84	29.79	29.81	29.85	29.99	30.05	29.85	30.1	30.25	30.22	30.29	30.31	30.01
avg	29.85	29.8	29.76	29.78	29.79	29.77	30	29.76	30	30.17	30.1	30.16	30.23	29.82
low	29.8	29.76	29.71	29.74	29.71	29.66	29.93	29.71	29.86	30.09	29.98	30.02	30.04	29.7
high	10	10	10	10	10	10	10	10	10	10	10	10	10	10
avg	8	9	8	10	8	7	10	4	8	10	10	10	10	10
low	0	6	6	7	2	0	10	1	7	8	9	1	3	10
high	15	21	24	24	12	20	14	20	16	14	29	18	20	23
avg	7	12	16	11	5	7	6	13	7	9	17	9	9	13
gust (high)	19	26	33	32	15	25	17	24	19	16	38	23	26	31
sum	0	0	0	0.06	0.14	0.18	T	0.19	0	0	0	0	0	0
Events	Fog			Rain	Rain	Fog, Rain		Snow						
Drop Outs	205%	133%	106%	76%	108%	67%	190%	36%	117%	118%	121%	134%	121%	110%
Outliers	35%	37%	14%	23%	9%	6%	39%	9%	32%	38%	26%	23%	23%	35%
Prolonged														
Altitude Failure	1.2%	2.0%	1.3%	1.7%	0.8%	0.0%	1.1%	0.0%	1.4%	2.9%	1.8%	1.1%	1.8%	3.3%
Repeated Data	0%	163%	1%	5%	1%	1%	2%	0%	45%	6%	15%	3%	23%	47%
Multiple Aircraft	3%	1%	5%	0%	8%	1%	6%	22%	1%	2%	1%	5%	1%	1%

Finley

Date	3/1/2015	3/2/2015	3/3/2015	3/4/2015	3/5/2015	3/6/2015	3/7/2015	6/1/2015	6/2/2015	6/3/2015	6/4/2015	6/5/2015	6/6/2015	6/7/2015
Temp. (°F)	high 21	29	24	3	26	40	41	73	68	62	73	76	69	82
	avg 11	15	12	-4	6	26	32	60	62	56	60	63	62	68
	low 1	1	-1	-10	-13	11	24	47	56	51	48	51	53	53
Dew Point (°F)	high 10	18	18	-10	10	30	29	52	61	60	60	58	64	61
	avg 5	9	-1	-14	-2	16	21	45	56	52	50	52	60	57
	low -5	-4	-11	-18	-20	6	16	36	48	46	43	49	52	52
Humidity (%)	high 77	80	77	70	70	78	84	67	100	96	84	96	100	100
	avg 67	69	64	60	60	60	69	54	85	89	67	66	90	81
	low 58	51	48	51	44	38	50	42	56	73	42	40	69	35
Sea Level Press. (in)	high 30.46	30.44	30.12	30.5	30.48	30.13	30.23	30.18	29.93	29.99	30.15	30.25	30.07	29.73
	avg 30.26	30.16	29.9	30.35	30.28	30	30.11	30.01	29.84	29.86	30.09	30.18	29.78	29.67
	low 30.17	29.78	29.76	30.12	30.01	29.86	29.96	29.85	29.74	29.73	29.99	30.07	29.69	29.64
Visibility (mi)	high 10	10	10	10	10	10	10	10	10	10	10	10	10	10
	avg 10	10	9	10	10	10	10	10	10	8	9	10	10	7
	low 5	10	2	9	10	10	10	10	2	2	2	10	8	0
Wind (mph)	high 18	30	26	20	20	13	16	26	20	18	13	14	18	22
	avg 12	11	18	12	10	8	11	14	12	13	13	8	6	7
	gust (high) -	36	38	26	26	21	-	34	28	23	17	24	24	30
Precip. (in)	sum 0	0	0	0	0	0	0	0	0.76	0.38	0	0	0.15	0.09
Events									Rain, Thunder	Rain			Fog, Rain	Fog, Rain, Thunder storm
Instances/Total														
Number of Aircraft Observed	Drop Outs 98%	132%	180%	124%	107%	111%	90%	234%	192%	154%	260%	278%	155%	267%
	Outliers 14%	28%	22%	24%	27%	37%	22%	28%	27%	25%	45%	48%	22%	43%
	Prolonged Altitude Failure 0.1%	0.7%	0.3%	0.4%	0.5%	1.8%	0.4%	0.3%	0.9%	0.2%	0.5%	0.7%	0.2%	0.1%
	Repeated Data 2%	4%	0%	5%	8%	14%	7%	2%	3%	3%	8%	10%	2%	1%
	Multiple Aircraft 0%	0%	3%	0%	0%	1%	0%	0%	1%	1%	1%	1%	1%	1%

Date	9/1/2015	9/2/2015	9/3/2015	9/4/2015	9/5/2015	9/6/2015	9/7/2015	12/1/2015	12/2/2015	12/3/2015	12/4/2015	12/5/2015	12/6/2015	12/7/2015
high	86	88	91	82	73	77	70	34	30	43	39	44	42	45
avg	69	74	80	72	68	65	58	30	20	30	28	34	32	34
low	52	60	69	64	62	50	47	23	11	16	17	25	21	23
high	57	69	74	74	71	68	57	30	20	30	32	33	31	33
avg	53	65	70	70	65	64	50	28	14	22	22	30	26	28
low	49	57	67	64	63	48	45	18	8	12	15	24	18	22
high	90	100	100	100	100	100	96	100	88	88	88	93	89	93
avg	60	86	85	90	97	95	74	90	80	76	79	81	80	82
low	28	48	51	72	88	42	46	71	61	59	69	65	63	62
high	29.88	29.84	29.8	29.86	29.91	29.99	30.03	29.92	30.04	30.18	30.13	30.25	30.27	29.91
avg	29.86	29.82	29.76	29.8	29.85	29.77	29.99	29.81	29.99	30.11	30.02	30.11	30.18	29.75
low	29.82	29.77	29.69	29.77	29.74	29.71	29.96	29.77	29.91	30.02	29.89	29.95	29.99	29.61
high	10	10	10	10	10	10	10	10	10	10	10	10	10	10
avg	10	6	8	9	4	5	10	5	10	10	10	10	10	10
low	10	0	2	1	0	0	9	1	10	10	10	10	10	10
high	13	16	28	13	14	15	14	13	8	14	23	15	13	14
avg	6	6	12	9	9	6	6	9	7	8	13	9	7	11
gust (high)	-	20	34	-	-	21	20	-	-	-	34	-	-	-
sum	0	0	0	0	0.5	0.04	0.06	0.02	0	0.11	0	0	0	0
Events				Rain , Thunder storm	Fog , Rain , Thunder storm	Fog , Rain , Thunder storm	Rain	Snow						
Drop Outs	404%	480%	571%	398%	373%	350%	340%	170%	331%	322%	309%	123%	180%	217%
Outliers	62%	73%	79%	53%	37%	40%	58%	31%	71%	70%	68%	34%	50%	46%
Instances/Total Number of Aircraft Observed														
Prolonged Altitude Failure	0.8%	0.7%	0.4%	0.1%	0.1%	0.2%	0.9%	1.1%	1.3%	1.5%	1.0%	1.3%	1.1%	1.5%
Repeated Data	8%	9%	5%	3%	1%	8%	15%	1%	10%	2%	9%	10%	9%	9%
Multiple Aircraft	1%	1%	1%	1%	2%	1%	1%	0%	1%	3%	0%	1%	1%	1%

REFERENCES

1. "New Radar Being Constructed at Hector." *Fargoairport.com*. Hector International Airport, 17 Oct. 2005. Web. 15 Feb. 2017.
2. Image Source: Stanton, John. "Finley FAA Radar Site." *FortWiki*. FortWiki, 5 Aug. 2016. Web. 28 Feb. 2017.
3. "About Us." ASSUREuas. FAA, n.d. Web. 26 Apr. 2017.
4. United States. Department of Transportation. William J. Hughes. Task A6: Surveillance Criticality Final Report. By Nicholas Allen, Evan Arnold, J.W. Bruce, Matthew McCrink, Mohammad Moallemi, William Semke, Kyle Snyder, Dawson Stott, Asma Tabassum, and Micheal Wing. Washington D.C.: Federal Aviation Administration, 2016. Print.
5. Richards, Mark A., James A. Scheer, and William A. Holm. Principles of modern radar. SciTech Pub., 2010.
6. "Introduction to TCAS II Version 7.1." Federal Aviation Administration. U.S. Department of Transportation, 28 Feb. 2011. Web. 28 Feb. 2016.
7. "NextGen Automatic Dependent Surveillance-Broadcast (ADS-B)." Federal Aviation Administration. U.S. Department of Transportation, 26 Oct. 2016. Web. 24 Jan. 2017.
8. 14, §§ 91-113 (Federal Aviation Administration 2014). Print.
9. Wolff, Christian. "Radar Sets." Radar Basics. Radartutorial.eu, n.d. Web. 15 Feb. 2017.
10. Oster, Thomas. "Primary surveillance radar (PSR)." EuroControl. EuroControl, n.d. Web. 15 Apr. 2017.
11. Roy, L. "Radar." Department of Electronics. Carleton University, 14 Feb. 2011. Web. 17 Mar. 2017.
12. "Fundamentals of Radar." FAA Academy Training: Air Traffic Basics. Washington D.C.: Federal Aviation Administration, 2012. 1-34. Print.

13. United States. Department of Transportation. Aeronautical Information Manual. Washington D.C.: Federal Aviation Administration, 2014. Print
14. United States. Department of Transportation. Order 7110.66E: National Beacon Code Allocation Plan. Washington D.C.: Federal Aviation Administration, 2015. Print.
15. 14, §§ 91-215 (Federal Aviation Administration 2014). Print.
16. Jain, Preeti. "Classification of RADARs." EngineersGarage. EngineersGarage, 21 Mar. 2017. Web. 29 Apr. 2017.
17. Kennedy, Chris. "UND Cessna 172S N604ND 'Sioux 604'." *Flickr*. Yahoo!, 18 Sept. 2013. Web. 30 Jan. 2017.
18. "Learn to Fly Helicopters with UND Aerospace." *AMP Helicopters RSS*. AMP Helicopters, 3 Nov. 2015. Web. 30 Jan. 2017.
19. Van Hassel, Erwin. "N803SK Delta Connection Canadair CL-600-2D24 Regional Jet CRJ-900LR." *PlaneSpotters.net*. Plane Spotters, 11 June 2012. Web. 5 May 2017.
20. Parkhouse, Richard. "Allegiant Air Airbus A320." *Airplane-Pictures.net*. Airplane Pictures, 29 May 2016. Web. 05 May 2017.
21. "Airport Surveillance Radar (ASR-11)." Federal Aviation Administration. U.S. Department of Transportation, 15 Jan. 2014. Web. 2 Feb. 2017.
22. United States. Department of Transportation. Order 1812.8: System Requirements for the Air Route Surveillance Radar - Model 4. Washington D.C.: Federal Aviation Administration, 1986. Print.
23. Weber, Mark E. "FAA Surveillance radar data as a complement to the WSR-88D network." *Preprints, 9th Conference on Aviation Range and Aerospace Meteorology*. Vol. 150. 2000.
24. Hardy, Quentin. "A Silicon Valley for Drones, in North Dakota." *The New York Times*. The New York Times, 25 Dec. 2015. Web. 25 Aug. 2016.
25. 14, §§ 107 (Federal Aviation Administration 2016). Print.
26. Atherton, Kelsey D. "The FAA Says There Will Be 7 Million Drones Flying Over America By 2020." *Popular Science*. Bonnier Corporation, 24 Mar. 2016. Web. 16 Oct. 2016.

27. United States. Government Accountability Office. UNMANNED AERIAL SYSTEMS: FAA Continues Progress toward Integration into the National Airspace. Washington D.C.: n.p., 2015. Print.
28. Nikoleris, Anastasios, et al. "Performance of an Automated System for Control of Traffic in Terminal Airspace." *16th AIAA Aviation Technology, Integration, and Operations Conference*. 2016.
29. Ong, Hao Yi, and Mykel J. Kochenderfer. "Short-term Conflict Resolution for Unmanned Aircraft Traffic Management." IEEE Xplore Digital Library. Proc. of Digital Avionics Systems Conference, Prague, Czech Republic. IEEE, 29 Oct. 2015. Web. 10 May 2017.
30. Aubert, Miles C., Serhat Üzümcü, Andrew R. Hutchins, and M. L. Cummings. "Toward the Development of a Low-Altitude Air Traffic Control Paradigm for Networks of Small, Autonomous Unmanned Aerial Vehicles." Aerospace Research Central. Proc. of AIAA Infotech @ Aerospace, Kissimmee, FL. American Institute of Aeronautics and Astronautics, 2015. Web. 12 May 2017.
31. Mahboubi, Zouhair, and Mykel J. Kochenderfer. "Autonomous Air Traffic Control for Non-Towered Airports." ATM Seminar US. Proc. of Eleventh USA/Europe Air Traffic Management Research and Development Seminar, Lisbon, Portugal. ATM Seminar, 26 June 2016. Web. 12 May 2017.
32. Hoffman, William. "Drones Will Need an Autonomous Air Traffic Control." Inverse. Inverse, 26 Oct. 2016. Web. 11 May 2017.
33. Ackerman, Evan. "NASA Developing Air Traffic Control System for Drones." IEEE Spectrum. IEEE, 03 Sept. 2014. Web. 11 May 2017.
34. Thomas, Michael L., LtCol. "Atlas of Radar Coverage of the Lower 48 Border States - Final." psugeo.org. Penn State, 19 Aug. 2002. Web. 8 Mar. 2017.
35. Healy, Thomas A., et al. Air Route Surveillance Radar Model 4 (ARSR-4) Operational Test and Evaluation (OT&E) Final Report. No. DOT/FAA/CT-TN96/26. Federal Aviation Administration Technical Center, Atlantic City NJ, 1997.
36. Weber, Ronald, and Joseph Schanne. Airport Surveillance Radar Model 11 (ASR-11) FAA Test and Evaluation Master Plan (TEMP). No. DOT/FAA/CT-TN97/27. Federal Aviation Administration Technical Center, Atlantic City NJ, 1998.
37. Mayer, Colin, and Panoe Tzanos. "Comparison of ASR-11 and ASR-9 Surveillance Radar Azimuth Error - IEEE Xplore Document." MIT Lincoln Laboratory. Federal Aviation Administration, 9 July 2011. Web. 03 Mar. 2017

38. Busch, Allen C. and Paul Bradbury. "Measurement and Analyses of ASR-4 System Error. Part I. Overview." 44.10 (2007): n. pag. National Technical Information Service. US. Department of Commerce, Nov. 1974. Web. 06 Mar. 2017.
39. "Air Traffic Control Radar Beacon System (ATCRBS)." The Story of Mode S: An Air Traffic Control Data Link Technology. MIT, 15 Dec. 2000. Web. 27 Jan. 2017.
40. Phillips, Darryl. "Mode A and Mode C: The Straight Scoop on How it Works." *Mode A and Mode C Codes*. AeroElectric, 15 Dec. 2015. Web. 30 Jan. 2017.
41. United States. Department of Commerce. Carlos M Gutierrez. *Assessment of the Effects of Wind Turbines on Air Traffic Control Radars*. By John J. Lemmon, John E. Carroll, Frank H. Sanders, and Doris Turner. N.p.: U.S. Department of Commerce, 2008. Print
42. Immoreev, I. J., and J. D. Taylor. "Future of Radars." IEEE Xplore Digital Library. Proc. of IEEE Conference on Ultra Wideband Systems and Technologies, Baltimore, MD. IEEE, 7 May 2002. Web. 15 May 2017.
43. Buckler, L. M. "The Use of Phased Array Radars at Civilian Airports." IEEE Xplore Digital Library. Proc. of Phased Array Systems and Technology, Boston, MA. IEEE, 06 Aug. 2002. Web. 15 May 2017.
44. United States. Department of Transportation. Multifunction Phased Array Radar (MPAR) Notional Functional Requirements Document. 1st ed. Washington D.C.: Federal Aviation Administration, 2012. Print.
45. Petrochilos, N., G. Galati, L. Mene, and E. Piracci. "Separation of Multiple Secondary Surveillance Radar Sources in a Real Environment by a Novel Projection Algorithm." IEEE Xplore Digital Library. Proc. of Signal Processing and Information Technology, Athens, Greece. IEEE, 23 Jan. 2006. Web. 15 May 2017.
46. Baud, O., N. Honore, and O. Taupin. "Radar / ADS-B Data Fusion Architecture for Experimentation Purpose." IEEE Xplore Digital Library. Proc. of Information Fusion, Florence, Italy. IEEE, 12 Feb. 2007. Web. 15 May 2017.
47. Castle, Michael W., Tan Trinh, Colin Mayer, and Christine Parry. "Evaluation of Separation Performance with ADS-B at the Philadelphia Key Site." IEEE Xplore Digital Library. Proc. of IEEE/AIAA 29th Digital Avionics Systems Conference, Salt Lake City, UT. IEEE, 3 Dec. 2010. Web. 15 May 2017.

48. United States. Department of Transportation. Center Radar Presentation (CENRAP). Washington D.C.: Federal Aviation Administration, 2014. Print.
49. United States. Department of Transportation. Altitude Reporting Equipment and Transponder System Maintenance and Inspection Practices. Washington D.C.: Federal Aviation Administration, 2017. Print.
50. DeSoto, Clinton B. "Radar Techniques - Primer Principles, April 1945 QST." Radar Techniques - Primer Principles. RF Cafe, 21 Feb. 2011. Web. 1 May 2017.
51. "Quick Reference Handbook: Cessna 172S." Airborne Aviation. Airborne Aviation Pty. Ltd., 14 July 2016. Web. 31 May 2017.
52. "Bombardier CRJ700 Flight Notes." Bombardier CRJ700. N.p., n.d. Web. 31 May 2017.
53. "Flying the Airbus A320." Fly WestWind. WestWind Virtual Airlines, 2014. Web. 31 May 2017.
54. Bar-Yehuda, Zohar. "Zoharby/plot_google_map." File Exchange - MATLAB Central. MathWorks, 10 Mar. 2017. Web. 10 Apr. 2017.
55. Altman, Yair. "Export_fig." File Exchange - MATLAB Central. MathWorks, 10 Apr. 2017. Web. 10 Apr. 2017.
56. Long, David. "Ellipse.m." File Exchange - MATLAB Central. MathWorks, 09 Oct. 1998. Web. 10 Apr. 2017.
57. United States. Department of Transportation. Michael Huerta. Order 8040.4A: Safety Risk Management Policy. Washington D.C.: Federal Aviation Administration, 2012. Print.
58. Skolnik, Merrill I. "Radar: Factors Effecting Radar Performance." Encyclopædia Britannica. Encyclopædia Britannica, Inc., 24 Oct. 2016. Web. 1 May 2017.
59. "Weather History." Weather Underground. The Weather Company, LLC, n.d. Web. 5 May 2017.
60. "VFR Weather Minimums." Federal Aviation Administration. U.S. Department of Transportation, n.d. Web. 10 May 2017.
61. "WindFarm." USGS Energy Resources Program Map. United States Geological Survey, 2014. Web. 25 Apr. 2017.