AIAA SciTech Forum
8–12 January 2018, Kissimmee, Florida
2018 AIAA Information Systems-AIAA Infotech @ Aerospace

10.2514/6.2018-0505

UAS Position Estimation in GPS-Degraded and Denied Environments Via ADS-B and Multilateration Fusion

Robert Larson *, Jonathon Winde†, and Christopher W. Lum ‡
Autonomous Flight Systems Laboratory
University of Washington, Seattle, WA, 98195, USA

This paper details estimation algorithms and software/hardware infrastructure used to estimate the position of a small unmanned aerial system (sUAS) in a global positioning system (GPS) denied environment. The sUAS is equipped with an automatic dependent surveillance broadcast (ADS-B) transponder and operates in the vicinity of a ground-based local area multilateration system (LAMS). A specialized ground control station (GCS) was developed for this application to simultaneously process information gathered from ADS-B and LAMS data streams. Estimation and fusion algorithms were developed to filter information provided to the GCS by the raw ADS-B and LAMS data streams. The algorithms use this filtered information to provide consistent, fused estimates of aircraft position. Three estimation algorithms and three fusion algorithms were developed and compared in this study. This paper focuses on the development of the software package associated with this project as well as the development of the estimation and fusion algorithms. In addition, this paper describes a simulation infrastructure that can be used to test various scenarios before conducting actual flight. Both simulation and flight testing results are presented as test cases for the estimation algorithms.

Nomenclature

ADS-B	Automatic Dependent Surveillance-Broadcast
AFSL	Autonomous Flight Systems Laboratory
EPU	Estimated Position Uncertainty
GCS	Ground Control System
GPS	Global Positioning System
HiL	Hardware-in-the-Loop
ILS	Instrument Landing System
INS	Inertial Navigation System
KDLS	Airport ID of Columbia Gorge Regional Airport
LAMS	Local Area Multilateration System
MFOC	Mobile Flight Operations Center
NACp	Navigation Accuracy Category for Position
PIC	Pilot in Command
PWM	Pulse Width Modulation
SSR	Secondary Surveillance Radar
TRAPIS	TRAnsponder-based Positioning Information System
UA/UAS	Unmanned Aircraft/Unmanned Aerial System
UW	University of Washington

*Research Assistant, William E. Boeing Department of Aeronautics and Astronautics, University of Washington; Guggenheim Hall Room 211, Box 352400, Seattle, WA 98195-2400, AIAA member.
†Research Assistant, William E. Boeing Department of Aeronautics and Astronautics, University of Washington; Guggenheim Hall Room 211, Box 352400, Seattle, WA 98195-2400, AIAA member.
‡Research Assistant Professor, William E. Boeing Department of Aeronautics and Astronautics, University of Washington; Guggenheim Hall Room 211, Box 352400, Seattle, WA 98195-2400, AIAA member.

I. Introduction

A. Problem Statement

The application of unmanned aerial systems (UAS) is wide and varied. Missions for UAS encompass search and rescue,[1,2] aerial mapping[3,4] and surveying,[5,6] and many others. The platforms to perform these missions are equally diverse ranging from large aircraft such as the Global Hawk down to smaller vehicles such as the Raven. Despite this large variety of systems, nearly all UAS are heavily reliant on global positioning system (GPS) data to provide position information. Although GPS provides a cheap and reliable means of determining aircraft position in a variety of situations, GPS signals are subject to degradation and jamming,[7,8] both of which affect the reliability of GPS information provided to the UAS. Signal degradation occurs as a result of variations in the geometry of the GPS satellite constellation over time, while signal jamming, though illegal, can be accomplished through the use of cheap, readily-available components.[9] Both GPS signal degradation and jamming cause navigational issues for UAS which rely on such information for autonomous flight guidance. This paper examines methods of computing inertial position of an aircraft in a GPS-degraded or GPS-denied environment through the use of secondary position information and several data fusion algorithms.

B. Literature Review

A previous paper by this group focused on the hardware and flight testing of an ADS-B equipped sUAS[10] and this forms the basis of the estimation and data fusion work presented in this paper. This paper focusing on the software algorithms that are used in this framework to provide state estimates of the sUAS in GPS-denied environments. Several major studies have been conducted in similar vein as the research presented in this paper. In 2009, researchers at the University of North Dakota presented software-in-the-loop simulations for an sUAS sense and avoid algorithm which made use of ADS-B information.[11] Another 2009 study involved an ADS-B based collision avoidance system to be used by sUAS in airspace with other unmanned and manned aircraft operating simultaneously.[12] A 2013 study investigated the possibility of incorporating ADS-B transponders on sUAS and presented a case study to include recommendations for ADS-B regulations regarding sUAS aircraft.[13] More recently, researchers investigated additional sense and avoid algorithms with access to multiple data streams to include traffic collision avoidance system (TCAS) and ADS-B information.[14] While studies such as these have largely focused on future regulations and algorithms for operating sUAS in airspace shared with other sUAS and manned aircraft, the research presented in this paper was focused on demonstrating the use of an ADS-B transponder on a commercially-available sUAS and tracking the aircraft in real time with ADS-B and secondary LAMS unit for GPS-degraded and GPS-denied operations.

II. Infrastructure

This section describes the hardware and software infrastructure used to perform state estimation using both ADS-B and LAMS data streams.

A. TRAPIS Infrastructure

The core software package in this application is referred to as the TRAnsponder-based Positioning Information System (TRAPIS). TRAPIS is a C# application with a Windows Presentation Foundation (WPF) graphical user interface (GUI) (Figure 2). It utilizes an event-driven architecture to easily disseminate and update information across the core application and all extensions. TRAPIS was designed to serve two purposes. The first and primary function is to fuse information from two data streams (ADS-B and LAMS) and present to the user a more complete, accurate, and reliable representation of the surrounding airspace. TRAPIS filters and combines two independent measurements of the same property with known uncertainties. Since the uncertainties of the measurements are known, TRAPIS can present to the user the best available data at any given time. The resulting estimation is more reliable because the two streams are independent. If the data flow from one stream in interrupted, due to either a technical failure or malicious activity, TRAPIS can still process and present the data from the second stream with minimal impact to the user.

The second purpose of TRAPIS is to provide a platform that can be used by third party extensions and

plugins to further model the airspace environment in real-time. An example of such an extension is the Wake Turbulence Estimator.[15] This extension uses information gathered by TRAPIS to predict the location of dangerous wake vortex corridors.

TRAPIS was initially designed to support two specific data sources. The first data source is ADS-B packets obtained from an ADS-B In receiver such as the Sagetech Clarity ADS-B receiver. The Clarity uses a proprietary Traffic Report Output message to publish information gleaned from ADS-B transmissions. The second data source is the LAMS, developed by Advanced Navigation and Positioning Corporation (ANPC), which uses ASTERIX Category 048 messages to report the computed location and velocity of nearby aircraft.

After decoding a new message from either data source, the system uses the extracted information to construct a TRAPIS Packet. TRAPIS Packets record which data source the information came from and the time the message was received, as well as the aircraft's location, velocity, emitter category, uncertainty of the altitude measurement, and uncertainty of the horizontal position measurement. The newly constructed TRAPIS Packets are then added to a database corresponding to that aircraft's unique identifier.

A vehicle reported by the Clarity receiver is identified by its ICAO address, a unique 24-bit code assigned to the aircraft during national registration. Alternatively, a vehicle reported by the LAMS is identified by its Track Number, a unique integer assigned to each aircraft by the LAMS. Since the Clarity receiver and the LAMS use different identifiers, a pairing of the information is not automatic. The system will treat an aircraft that is reported by both data streams as separate vehicles until the user manually pairs the data. Once the data streams are paired, the system can begin fusing the information (see Section III).

TRAPIS was primarily designed to provide real time information to the user, however, several post analysis tools were also incorporated. First, every packet that TRAPIS receives from both the Clarity receiver and the LAMS are automatically logged. If the user experiences something unexpected while running TRAPIS, developers can gather these logs, use Python scripts to feed the raw packets back through TRAPIS, and recreated exactly what the user experienced. This feature became a useful debugging tool and allowed for rapid revisions during development. Second, the TRAPIS packet databases can be exported to comma-separated values (CSV) files. This includes the unaltered, the estimated, and the fused databases. Having all this information in CSV files allows for easy importation into other engineering tools. These files were used by developers to create, analyze, and improve the estimators and fusers (see Section III), but are also available to third party developers creating extensions. Third, the TRAPIS packet databases can be exported to Keyhole Markup Language (KML) files. These KML files allow all the information gathered and created by TRAPIS to be displayed natively in Google Earth. The 3D graphical representation of the data provides a clear depiction of the surrounding airspace. Many of the figures in this document were created using these KML files.

B. Simulation Infrastructure

Three simulators were used during the development of TRAPIS.

1. Clarity Simulator

Sagetech provided a simulator of the Clarity ADS-B receiver. This standalone application produced realistic flight data, packaged into Sagetech Traffic Report Output messages, and sent via UDP to the same port used by the Clarity receiver. This simulator was used primarily during early development to verify that the Traffic Report Output messages were properly ingested and stored in the TRAPIS Packet database.

2. LAMS Simulator

ANPC provided a simulator of the LAMS. It was capable of simulating multiple aircraft simultaneously, packaging the data into ASTRIX Category 048 messages, and sending the packets via UDP to the same port used by the LAMS. This simulator was used primarily during early development to verify that the Category 048 messages were properly ingested and stored in the TRAPIS Packet database.

3. TRAPIS Simulator

The Clarity and LAMS simulators were useful for early testing of the TRAPIS architecture. However, since they were created by different companies for different purposes, they were unable to simulate the same aircraft/entity with an ADS-B transponder and LAMS data stream simultaneously. Furthermore, they

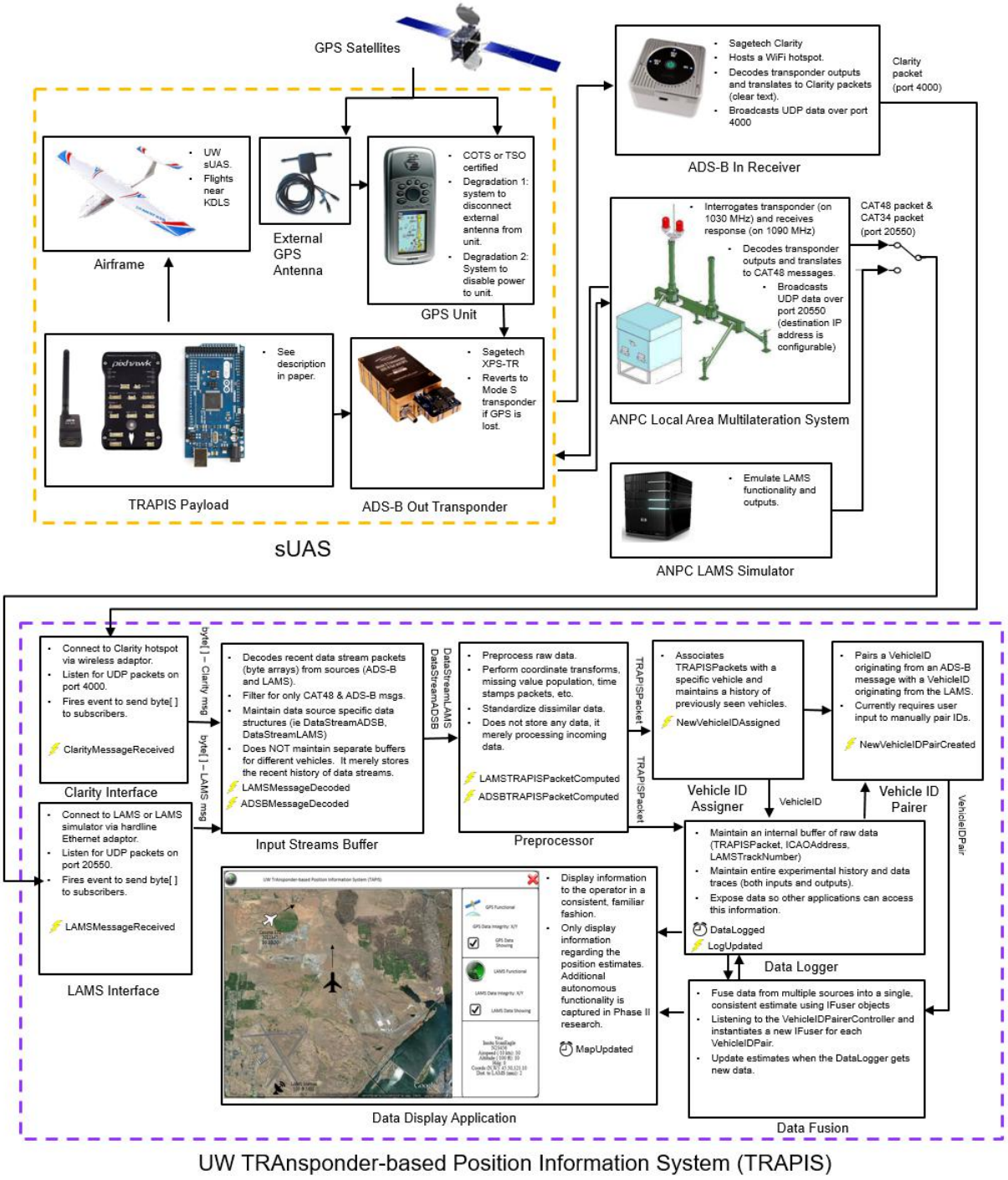


Figure 1. TRAPIS block diagram.

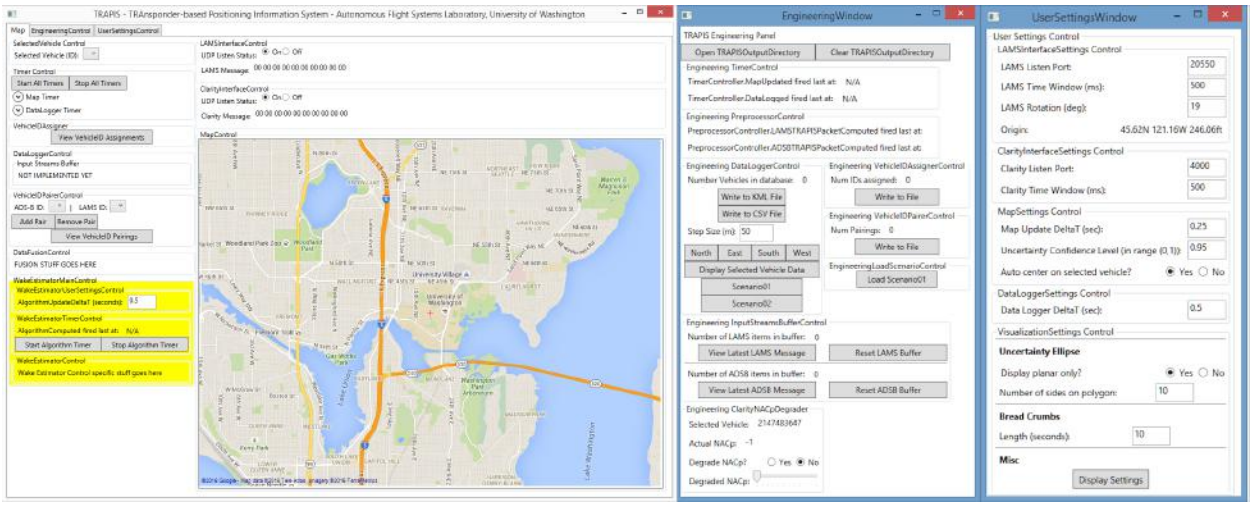


Figure 2. TRAPIS graphical user interface.

lacked the ability to degrade position measurements and were difficult to modify for custom flight paths or scenarios. Without these features, the Sagetech and ANPC simulators could not be used to test TRAPIS estimators or fusers. To overcome these obstacles, the UW AFSL created the TRAPIS Simulator.

The TRAPIS Simulator allows for multiple aircraft, with different emitter categories, and highly adjustable flight routes to be simulated at once. Encoders of both the Sagetech Traffic Report Output message and the ASTERIX Category 048 message were developed. The simulator uses these encoders to simultaneously emulate messages from both the Clarity receiver and the LAMS. It has the ability to degrade and add noise to the position measurements of both sources independently. The TRAPIS Simulator was used to test every aspect of the TRAPIS system and provided a means for flight test engineers to simulate the expected results before every test (see Section IV).

III. Estimation and Fusion

This section describes the various estimation algorithms used to filter/estimate the location of the aircraft based on a single data stream (either ADS-B or LAMS). This then describes various fusion algorithms which are used to combine outputs from two estimators into a single, consistent estimate of aircraft position. Additional details on the development and implementation of these algorithms are detailed in other works.[16]

A. Estimation Algorithms

In order to accurately estimate and fuse vehicle flight paths from ADS-B and LAMS information sources in varying states of degradation or denial, several estimation algorithms were developed. These estimation algorithms ranged in complexity and provided a multi-faceted approach to accurate estimation of vehicle flight paths. At the lowest level, the estimation algorithms passed unaltered position data through the TRAPIS framework to the user interface, and at the highest level the estimation algorithms accounted for position errors associated with ADS-B and LAMS TRAPIS information packets in real-time. The three estimators developed for this project allowed for a build-up approach to accurate vehicle position estimation, and provided TRAPIS users with a variety of estimation tools to choose from.

1. DoNothingEstimator

At the lowest level, estimation associated with TRAPIS relied on unaltered ADS-B and LAMS position information being passed directly to the TRAPIS user interface. In accordance with this description of the basic information handoff that occurred for the lowest-level estimator, this simple estimator was called the *DoNothingEstimator*. Although the *DoNothingEstimator* did not alter the position information being provided to the TRAPIS interface, it provided the basic building block from which additional estimators

were developed. Additionally, assuming that all ADS-B and LAMS information being ingested by TRAPIS is not being degraded or denied, then the *DoNothingEstimator* avoids unnecessary filtering of pre-filtered ADS-B and LAMS data.

The *DoNothingEstimator* was primarily used as an initial test of estimation algorithm implementation within the broader TRAPIS framework. Since the estimator did not involve any computations or actual state estimation, it was not included in the final version of TRAPIS used during the culminating flight demonstration at the Columbia Gorge Regional Airport (Airport ID: KDLS).

2. Kalman Filter Estimator

Once the *DoNothingEstimator* was completed, two additional estimators were developed. The additional estimators were designed in order to incrementally increase the complexity of the position estimates provided to the data fusion algorithms. During initial outlining and planning, it was determined that the advanced position estimation algorithms should include filters designed to reduce the known errors associated with position measurements in order to obtain consistent, accurate estimates of aircraft positions.

The first of these two estimators, the *KalmanFilterEstimator*, was designed to reduce the error associated with position estimates through the use of basic filtering with assumed position errors. Based on several factors including the number of available GPS satellites, the orientation of said satellites, the orientation of the GPS antenna on the receiving unit, and others, the accuracy of GPS position measurements can change significantly over the course of a measurement period. In ADS-B applications, this measurement error is reported as a navigation accuracy category for position (NACp) value, for which there are corresponding position errors and altitude errors. A table of these NACp values and their associated position errors is shown in Figure 3.[17]

NACp	95% Horizontal Accuracy Bound (EPU)	Comment
0	EPU ≥ 10 NM	Unknown accuracy
1	EPU < 10 NM	RNP-10 accuracy
2	EPU < 4 NM	RNP-4 accuracy
3	EPU < 2 NM	RNP-2 accuracy
4	EPU < 1 NM	RNP-1 accuracy
5	EPU < 0.5 NM	RNP-0.5 accuracy
6	EPU < 0.3 NM	RNP-0.3 accuracy
7	EPU < 0.1 NM	RNP-0.1 accuracy
8	EPU < 0.05 NM	e.g., GPS (with SA)
9	EPU < 30 m	e.g., GPS (SA off)
10	EPU < 10 m	e.g., WAAS
11	EPU < 3 m	e.g., LAAS

Figure 3. NACp values and associated position errors.[17]

From the information contained in Figure 3 it can be seen that for each NACp value there is an associated estimated position uncertainty (EPU), which corresponds to the horizontal position error for a 95% horizontal position accuracy bound. The range of available NACp values gives a wide range of position errors for GPS position measurement accuracy. Reported NACp values in ADS-B Out packets fall in the range from 7 to 11 during normal operations, assuming there is no degradation or jamming of the GPS signal being received by the ADS-B transceiver unit. For the purposes of this project however, the entire range of NACp values was acceptable for use since the GPS signal was simulated to be degraded or denied.

Once the NACp table shown in Figure 3 was found and initial analysis of real ADS-B Out packets was conducted, an initial choice of horizontal and vertical positions errors was required for the *KalmanFilterEstimator* position estimation algorithm. It was determined that for the requirements of the *KalmanFilterEstimator*, a NACp value of 9 would be used, correpsonding to a horizontal position error of 10 meters. In addition to the assumed horizontal position error, the vertical position error was assumed to be 7.5 meters. This vertical position error was chosen as a result of the altitude reporting capabilities of the Sagetech XPS-TR transponder unit. The transponder was only able to resolve pressure altitude differences of 25 feet, and as a result the vertical position error was selected to be 7.5 meters which corresponds to approximately 25 feet.

The same horizontal and vertical position errors that were assumed for the ADS-B position estimates were also assumed for the LAMS position estimates. Although the LAMS systems is subject to position errors that are dependent on the range of tracked aircraft from the LAMS system, all sUAS and manned aircraft testing required for the culminating flight test was performed within 10 nautical miles of the LAMS equipment site at the KDLS airfield. Furthermore, the horizontal position accuracy of the LAMS system manifests itself in both range and bearing errors. Although these horizontal position errors are not directly related to those for the GPS information used in the ADS-B packets, it was assumed that the errors were similar for the test applications required of the tracking algorithms. Since the maximum reliable tracking range of the LAMS system is a 60 nautical mile radius from the LAMS tracking station, but all flight operations were conducted within a 10 nautical mile radius from the LAMS station, it was determined that 10 meters was a reasonable assumption for the LAMS horizontal position errors. Additionally, the *KalmanFilterEstimator* only served as an interim estimation algorithm and was replaced by the *DynamicKalmanFilterEstimator* in the final iteration of TRAPIS, so any errors associated with the assumptions made for the *KalmanFilterEstimator* were taken care of in the final TRAPIS product.

As evidenced by the name associated with the filter, the *KalmanFilterEstimator* was designed to function as a basic discrete time Kalman filter. The prediction and update equations associated with the Kalman filter are shown in general form in Equations 1 through 5.[18] Since there were no systems inputs that directly affected the system state, no terms for an input vector were included in the prediction and update equations associated with the *KalmanFilterEstimator*.

$$x_{predicted} = A * x_{n-1} \tag{1}$$

$$P_{precicted} = A * P_{n-1} * A^T + F * Q * F^T \tag{2}$$

$$K = P_{predicted} * H^T * \Big(H * P_{predicted} * H^T + R\Big)^{-1} \tag{3}$$

$$x_n = x_{predicted} + K * \Big(Y - H * x_{predicted}\Big) \tag{4}$$

$$P_n = \Big(I - K * H\Big) * P_{predicted} \tag{5}$$

Using the discrete time Kalman Filter equations detailed above for both the ADS-B and LAMS information being supplied to the TRAPIS position estimation framework required a standard reference from which to compute position changes during a flight scenario. Without a standard framework and geographical reference for the position measurements made using both the LAMS and ADS-B systems, there would be no way to accurately compare the position estimates generated using the two methods. Based on this requirement, all vehicle positions, velocities, and associated errors were transformed into a local coordinate system hereafter referred to as the *UW* coordinate frame. In essence, the methods associated with the *UW* coordinate system transform coordinates and velocities associated with a vehicle system into a local North, East, Down frame in order to allow for ease of calculation. Once all required calculations are performed, the *UW* coordinates can be converted back into known distances and velocities within the original frame. An additional benefit of the *UW* coordinate system is that it allows for direct application of velocities over specified time periods culminating in changes to the latitude and longitude positions of an aircraft. The coordinate transformations involved in reconciling position and velocity changes over changes in latitudes and longitudes are numerous and can be rather complex depending on the distances and magnetic travel directions involved. By using the *UW* coordinate system, these conversions are performed directly by the code base, and it is not left to the user to define required coordinate transformations each time new code is written.

Regardless of whether a new *KalmanFilterEstimator* object was associated with ADS-B or LAMS information, the position and velocity information associated with the corresponding measurement was immediately converted into the *UW* coordinate system. Once the distances, velocities, and associated errors had been transformed into the new coordinate system, the state vector was generated according to the definition shown in Equation 6.

$$x = \begin{bmatrix} EastPosition(m) \\ EastVelocity(m/s) \\ NorthPosition(m) \\ NorthVelocity(m/s) \\ DownPosition(m) \\ DownVelocity(m/s) \end{bmatrix} \tag{6}$$

In addition to the state vector, it is necessary to define prediction and update matrices for the Kalman filter. Since the *KalmanFilterEstimator* is a discrete time Kalman filter, the time step associated with the system had to be computed and used in the matrices associated with the Kalman filter equations. Due to the nature of the data packets generated from the ADS-B and LAMS data streams, the time stamps associated with the packets are not generated at a constant rate. As such, the time between when packets are received in the estimation algorithm is different for each new packet. In order to compensate for this, the time step associated with the Kalman filter is changed each time a new observation enters the estimation algorithm. This time step is calculated as the difference between the time the new packet was received and the time the previous packet was received.

The measurement error for the *KalmanFilterEstimator* was set equal to the assumed horizontal position error of 10 meters for the North and East directions, and 7.5 meters for the Down direction in the UW coordinate frame. These assumed position errors were used to directly model the measurement noise associated with the system and form the measurement covariance matrix. Since the coordinate system required transformation of the positions and velocities to the local UW coordinate frame, the process noise associated with the system was more difficult to quantify. After many simulations were completed to analyze the effects of changes to the process noise covariance matrix, a choice was made to develop the process noise covariance matrix using the errors associated with the position and velocity measurements of the system. The matrices associated with the discrete time Kalman filter algorithm in the *KalmanFilterEstimator* are detailed in Equations 7 through 12. The time step calculated between successive observations is shown as dt, and the horizontal and vertical position errors are shown by σ_x and σ_y, respectively.

$$A = \begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$P = \begin{bmatrix} \sigma_x & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_x^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_y & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_z & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_z^2 \end{bmatrix} \tag{10}$$

$$Q = \begin{bmatrix} \sigma_x & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_x^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_y & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_z & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_z^2 \end{bmatrix} \tag{11}$$

$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\sigma_x^2}{10} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_y^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sigma_y^2}{10} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_z^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\sigma_z^2}{10} \end{bmatrix} \tag{12}$$

Using these matrices and the discrete time Kalman filter equations detailed in Equations 1 through 5, the positions and velocities of the tracked vehicle were computed at each new time step. Once the Kalman filter prediction and update equations had been run for each time step, the final velocities in the North, East, Down coordinate system were gathered directly from the output vector. The final positions were determined by offsetting the defined origin by the final North, East, Down position vectors contained within the Kalman filter output state vector.

3. DynamicKalmanFilterEstimator

After the *KalmanFilterEstimator* was completed, the third and final estimation algorithm, the *DynamicKalmanFilterEstimator*, was developed. In a similar fashion to the *KalmanFilterEstimator*, the *DynamicKalmanFilterEstimator* is a discrete time Kalman filter which uses the same prediction and update equations detailed in Equations 1 through 5. Unlike the *KalmanFilterEstimator*, the *DynamicKalmanFilterEstimator* uses the actual horizontal and vertical position errors associated with the information contained within the observation/data packets to update the required Kalman Filter matrices at each new time step. While the *KalmanFilterEstimator* assumed that the horizontal position error was constant at 10 meters and the vertical position error was constant at 7.5 meters, the *DynamicKalmanFilterEstimator* uses the appropriate position error associated with the data packets for either the ADS-B or the LAMS data stream, thereby ensuring that the appropriate position error is used to generate the position estimates for the next time step.

After simulations were conducted using all three of the estimation algorithms, it was shown that the *DynamicKalmanFilterEstimator* provided the best estimates of aircraft position regardless of the integrity of the ADS-B and LAMS data streams. As a result of these simulation results, the *DynamicKalmanFilterEstimator* was selected for use with the final iteration of TRAPIS, and this estimation algorithm was used for all subsequent simulation and testing including the final flight demonstration.

B. Fusion Algorithms

Once the three estimators were created, three data fusion algorithms were created as well. A data fusion algorithm is simply a method to take the outputs of two estimators (one for the ADS-B signal and one for the LAMS signal) and fuse these into a single, consistent estimate. In similar fashion to the estimation algorithms, these data fusion algorithms were created with varying levels of complexity in order to provide a multi-tiered

approach to fusion of vehicle position estimates. At the lowest level, the fusion of vehicle position estimates operates as a simple switch between the ADS-B position estimates and the LAMS position estimates, allowing either the ADS-B estimates or the LAMS estimates to be used for the fused position estimate. At the highest level, the fusion algorithms determine the weighted average of ADS-B and LAMS position estimates based on the reported error associated with the ADS-B and LAMS data streams. Ultimately, the three data fusion algorithms complement the three estimation algorithms and allow for varying levels of complexity for position estimation, fusion, and reporting in the TRAPIS interface.

1. *SimpleFuser*

The simplest fusion algorithm offers a selection of either the ADS-B position estimates or the LAMS position estimates. In a similar fashion to the *DoNothingEstimator* which passed either the unmodified ADS-B or LAMS data stream through to the fusion algorithm, the *SimpleFuser* passed either the ADS-B or LAMS estimate through as the fused estimate seen by the user through the TRAPIS user interface.

The *SimpleFuser* works with any of the three possible position estimation algorithms to provide a fused position estimate to the user of TRAPIS. If the *SimpleFuser* is being used, then the TRAPIS user can select which estimate is desired, namely the ADS-B or LAMS estimate. By viewing the raw data streams being consumed by TRAPIS in real-time, the user can determine which of the two data streams should be trusted over the other. Using the *SimpleFuser*, the user can then select which data stream should be used as the fused estimate. Although the position information provided to the fusion algorithm within the two separate data streams is not combined in any manner, the *SimpleFuser* is well-suited for situations in which one of the two data streams is unaltered while the other data stream is being degraded or denied. The primary drawback of this method is that it requires manual selection of a data stream by the user who must be familiar enough with the system to make an informed decision.

2. *WeightedFuser*

After the *SimpleFuser* algorithm was developed, two additional fusion algorithms were developed, the *WeightedFuser* and the *KalmanFuser*. The first of these two fusion algorithms, the *WeightedFuser*, provides increased estimate fusion functionality over the *SimpleFuser* by averaging the position estimates provided to TRAPIS by the ADS-B and LAMS data streams. In order to accomplish this weighted averaging, the *WeightedFuser* computes the weighted geographic midpoint of the most recent positions provided to TRAPIS by the ADS-B and the LAMS data streams by means of weighted averaging methods. By using a weighted average as opposed to a direct calculation of the unweighted geographic midpoint, the error associated with the ADS-B and LAMS packets provided to TRAPIS can be included in the calculation. Depending on which of the two data streams is providing more accurate position information, the geographic midpoint can be corrected appropriately.

After the most recent state estimates are received by the *WeightedFuser*, the weighted geographic midpoint is computed. In order to compute this midpoint, both position estimates are first converted to a local Cartesian coordinate system. The equations required for this conversion are shown in Equations 13 through 15, where *lat* corresponds to the latitude of the estimate, and *lon* corresponds to the longitude of the estimate.[19]

$$X = \cos(lat) * \cos(lon) \tag{13}$$

$$Y = \cos(lat) * \sin(lon) \tag{14}$$

$$Z = \sin(lat) \tag{15}$$

Once these Cartesian coordinates are calculated for both the ADS-B and LAMS estimates, the weighting factors are determined by using the planar standard deviations reported in the TRAPIS packets associated with the most recent observations. These weighting factors are determined by Equation 16 where σ_{planar} represents the planar standard deviation of the corresponding ADS-B or LAMS estimate. Using the inverse ensures that the measurements with larger associated planar standard deviations carry less weight in the final fused estimate.

$$w = \frac{1}{\sigma_{planar}} \tag{16}$$

Using the Cartesian coordinates computed for both the ADS-B and the LAMS estimates along with the weighting factors determined by the planar standard deviations associated with each of the estimates, the weighted Cartesian coordinates are determined using Equations 17 through 19. The subscripts on the variables indicate the Cartesian coordinates and weighting factors associated with the ADS-B and LAMS estimates, respectively.

$$X_{fused} = \frac{\left(X_{ADSB} * w_{ADSB} + X_{LAMS} * w_{LAMS}\right)}{\left(w_{ADSB} + w_{LAMS}\right)} \tag{17}$$

$$Y_{fused} = \frac{\left(Y_{ADSB} * w_{ADSB} + Y_{LAMS} * w_{LAMS}\right)}{\left(w_{ADSB} + w_{LAMS}\right)} \tag{18}$$

$$Z_{fused} = \frac{\left(Z_{ADSB} * w_{ADSB} + Z_{LAMS} * w_{LAMS}\right)}{\left(w_{ADSB} + w_{LAMS}\right)} \tag{19}$$

After the fused Cartesian positions are calculated, the fused coordinates are converted back into a fused latitude and longitude estimate using Equations 20 through 22.

$$Lon_{fused} = \text{atan2}(Y_{fused}, Xfused) \tag{20}$$

$$Hyp = \sqrt{X_{fused}^2 + Y_{fused}^2} \tag{21}$$

$$Lat_{fused} = \text{atan2}(Z_{fused}, Hyp) \tag{22}$$

The latitude and longitude coordinates obtained from Equations 20 through 22 result in the weighted geographic midpoint of the ADS-B and LAMS position estimates. Once these coordinates are calculated, the weighted average of the aircraft velocity vector in terms of North, East, Down components is determined using Equations 17 through 19. The same planar deviation weighting factors are used for the fused velocity calculations. Once the fused North, East, and Down velocity components are determined, they are placed into a *UW* velocity vector and included in the *TRAPISPacket* generated for the fused estimate.

The weighted average of the aircraft altitude is calculated by standard methods using Equations 23 and 24. In these equations, the subscripted altitudes correspond to the altitudes reported in the most recent ADS-B and LAMS observations provided to the fusion algorithm, and the variable σ_{alt} corresponds to the reported altitude standard deviation in the most recent ADS-B or LAMS observation.

$$w_{Alt} = \frac{1}{\sigma_{alt}} \tag{23}$$

$$Alt_{fused} = \frac{\left(Alt_{ADSB} * w_{Alt:ADSB} + Alt_{LAMS} * w_{Alt:LAMS}\right)}{\left(w_{Alt:ADSB} + w_{Alt:LAMS}\right)} \tag{24}$$

The fused planar and altitude standard deviations are calculated using a direct average, and these fused standard deviations are included in the appropriate fields of the data associated with the fused estimate. Each time new observations are added to both of the estimators associated with the fusion algorithm, the *WeightedFuser* computes a new fused estimate which is then provided to the TRAPIS user interface to be viewed by the user. The *WeightedFuser* was used for preliminary testing of the weighted geometric midpoint calculations before the third fusion algorithm was developed.

3. KalmanFuser

Once the *WeightedFuser* was completed, the *KalmanFuser* was developed. The *KalmanFuser* was designed to further streamline the fusion of ADS-B and LAMS estimates by directly weighting the estimated state vectors associated with both estimation algorithms.

In order to perform this data fusion, Equation 25 was used as detailed in.[20] The equation provides a means of finding the weighted average of the state vectors, X, associated with two Kalman filters by means of their error covariance matrices, P.

$$X_{fused} = \frac{\left(X_{ADSB}/P_{ADSB} + X_{LAMS}/P_{LAMS}\right)}{\left(1/P_{ADSB} + 1/P_{LAMS}\right)} \tag{25}$$

Equation 25 assumes that the Kalman filters used are associated with redundant streams of information, namely sensors that are providing the same state measurements to the system. For this application, since the ADS-B and LAMS data streams provide the same vehicle information to TRAPIS, the equation proved useful.

If this direct weighting of the estimated state vectors failed, the *KalmanFuser* used the same methods as the *WeightedFuser* to compute an appropriate fused estimate of the vehicle position and velocity. In the event that one of the data streams was no longer providing information to the fusion algorithm, in the event of GPS-denial for example, the *KalmanFuser* was designed to provide the most recent estimate of the non-denied data stream as the fused estimate. Upon re-acquisition of the denied signal, the *KalmanFuser* would once again weight the positions and velocities based on the standard deviations associated with the most recent estimates from both data streams. Accordingly, the *KalmanFuser* was used with the *DynamicKalmanFilterEstimator* for the final iteration of TRAPIS to provide the most appropriately-fused vehicle data to the user.

IV. Results

This section details results of the aforementioned estimation and fusion algorithms. Results from both simulation flight testing are presented and discussed.

A. Simulation Results

Before flight test data were gathered with the ADS-B payload flown on an AFSL sUAS, a variety of simulations were run. These simulations were designed to ensure that all TRAPIS code worked as planned, specifically as it related to the implementation of the estimation and fusion algorithms. Although the final version of TRAPIS only made use of the *DynamicKalmanFilterEstimator* estimation algorithm and the *KalmanFuser* data fusion algorithm, all of the estimation and fusion algorithms were tested in simulation. Through simulation, it was shown that all estimation and fusion algorithms performed as desired within the TRAPIS software framework.

Simulation Scenario

In order to accurately model the flight test that was planned as the culminating event of the TRAPIS research, a simulation scenario was developed in which three aircraft were flying in geographically-separate airspaces around the KDLS airfield. These airspaces were based on the initial flight test design, before additional factors and land-use agreements prevented the original test airspaces from being used as-planned. The original test design involved an AFSL sUAS carrying the TRAPIS payload flying in an airspace 3.5 nautical miles North-Northeast of the KDLS airfield, with two manned aircraft flying 5.2 nautical miles to the North and 5.2 nautical miles to the East of the KDLS airfield, respectively. The locations of these original test areas are shown in Figure 4.

In Figure 4 the test airspaces are marked by the black rectangles with yellow pin identifiers. These approximate locations were used for the simulated aircraft in order to closely match the planned flight test airspaces and aircraft flight paths.

After the simulation aircraft and routes were created, the simulations were completed in order to test all of the estimation and fusion algorithms. In order to ensure that the estimation and fusion algorithms worked

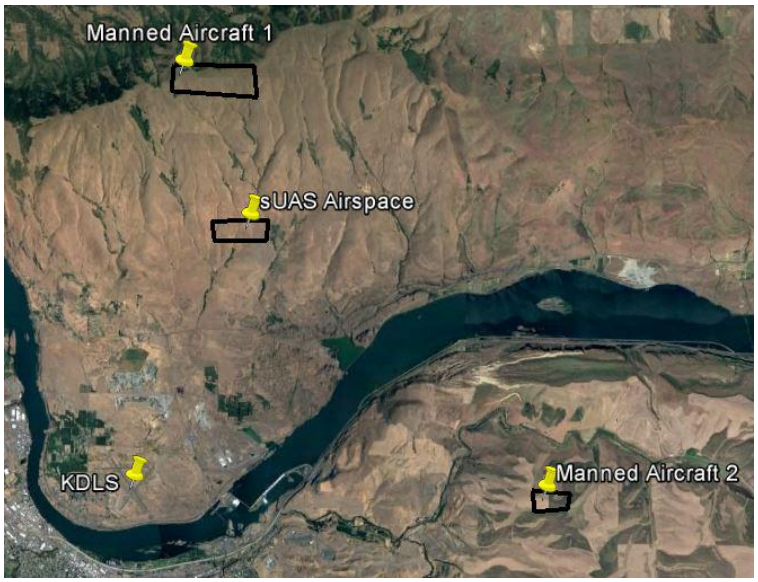


Figure 4. TRAPIS simulation scenario airspaces.

Pair Number	Estimation Algorithm	Fusion Algorithm
1	DoNothingEstimator	SimpleFuser
2	DoNothingEstimator	WeightedFuser
3	DoNothingEstimator	KalmanFuser
4	KalmanFilterEstimator	SimpleFuser
5	KalmanFilterEstimator	WeightedFuser
6	KalmanFilterEstimator	KalmanFuser
7	DynamicKalmanFilterEstimator	SimpleFuser
8	DynamicKalmanFilterEstimator	WeightedFuser
9	DynamicKalmanFilterEstimator	KalmanFuser

Table 1. Simulation run matrix.

as designed from the lowest to highest levels, a build-up testing approach was used. By using this build-up approach, low-level code problems could be identified in the less-complex algorithms and fixed before continued testing with the advanced algorithms. Based on this desired build-up method, the simulations were conducted in accordance with the framework shown in Table 1. Using this run structure all estimation and fusion algorithms were tested with one another. Each of the runs were conducted multiple times, with variation in the length and scope of ADS-B and LAMS data stream degradation and denial. Simulation results showed that all estimation and fusion algorithms worked as-desired, even in the presence of ADS-B and LAMS data stream degradation or denial.

In order to accurately summarize the simulation results and the performance of each of the algorithms, several of the simulation runs are summarized in the following sections as detailed in Table 2 and Table 3. For the runs conducted using the *SimpleFuser*, the LAMS data stream was not degraded. This was done in order to accurately simulate the LAMS data stream being chosen over a degraded or denied ADS-B data stream. Furthermore, for all ADS-B denied operations, the NACp value was set to 8, with corresponding standard deviations applied to the TRAPIS packets as detailed in Figure 3.

Run Number	Estimation Algorithm	Fusion Algorithm
1	DoNothingEstimator	SimpleFuser
2	KalmanFilterEstimator	WeightedFuser
3	DynamicKalmanFilterEstimator	KalmanFuser
4	DynamicKalmanFilterEstimator	KalmanFuser

Table 2. Abbreviated simulation run matrix.

Run Number	ADS-B Degraded	LAMS Degraded	ADS-B Denied
1	Yes	No	No
2	Yes	Yes	No
3	Yes	Yes	No
4	Yes	Yes	Yes

Table 3. Simulation ADS-B and LAMS degradation states.

1. DoNothingEstimator and SimpleFuser

At the lowest-level, position estimation involved the use of the raw, unaltered ADS-B and LAMS data streams by means of the *DoNothingEstimator*. Since the *DoNothingEstimator* did not alter the position information gathered from the data streams in any manner, the scope of acceptable applications was necessarily limited. For the purposes of simulation, the only application of the *DoNothingEstimator* that provided realistic results was one in which one of the two data streams was unaffected by degradation or denial. In order to create such a situation, the simulated ADS-B data stream was assumed to be degraded from an optimal NACp value of 10 to a NACp value of 8. The ADS-B data stream was degraded during the entire simulation time span of 350 seconds, but the LAMS data stream was assumed to be unaltered, thereby providing the TRAPIS user with true aircraft positions.

In order to check the performance of the *DoNothingEstimator*, two .kml files were generated at the end of the simulation. These .kml files contained the information from the raw ADS-B and LAMS data streams as well as the information from the estimated ADS-B and LAMS data streams. Once these .kml files were generated, they were opened in Google Earth and the data from the raw streams were compared to the data from the estimated streams. For the *DoNothingEstimator* the estimated data were expected to be the same as the raw data, a result which was confirmed by the simulation. The data associated with the ADS-B and LAMS streams for the simulated sUAS are shown in Figure 5. The raw position data are shown by blue aircraft markers, while the estimated position data are shown by red aircraft markers.

From the figure, it can be seen that the *DoNothingEstimator* performed as anticipated, where the estimated positions are equivalent to the raw positions for both the ADS-B and LAMS data streams. Figures 5(a) and 5(b) show the raw and estimated position information corresponding to the ADS-B data, while Figures 5(c) and 5(b) show the same information for the LAMS data during one circuit of the planned sUAS flight path. The simulated sUAS traveled in a counter-clockwise direction, and began its circuit in the northwest corner of the flight path. Since the estimation algorithm did not start providing estimates until at least three TRAPIS Packets were received, it can be seen that the initial raw positions provided to TRAPIS were not associated with corresponding estimated positions.

While the *DoNothingEstimator* was being used to generate the estimated positions associated with the ADS-B and LAMS data streams, the *SimpleFuser* was being used to return the most accurate of the two estimated streams. In the case of this initial simulation, the ADS-B data stream was degraded but the LAMS data stream was not degraded. As a result of this, the *SimpleFuser* provided the LAMS estimates as the fused vehicle position estimates. This result is shown in Figure 6, where the green aircraft markers indicate the fused position estimates. From Figure 6 it can be seen that the fused estimates correspond to the LAMS estimated positions. The *SimpleFuser* allows the TRAPIS user to choose which estimate (ADS-B or LAMS) to use for the fused estimate, and therefore proved useful for scenarios in which one of the two data streams was degraded while the other was not.

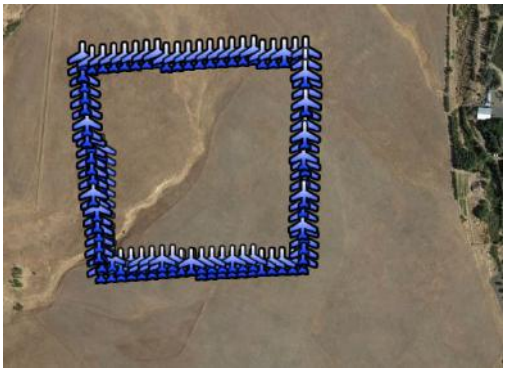
2. KalmanFilterEstimator and WeightedFuser

Although the *DoNothingEstimator* and the *SimpleFuser* proved desirable for providing accurate fused estimates in the presence of an unaltered data stream, these algorithms were not designed for use in the unlikely event that both data streams provide unreliable information to TRAPIS. In order to ensure that this case could be handled, the higher-level estimation and fusion algorithms were tested.

For the simulation in which the *KalmanFilterEstimator* was tested, the ADS-B data stream remained degraded to a NACp value of 8 for the entire simulation time span. The LAMS data stream was degraded to a NACp-equivalent of 6, thereby presenting position estimates to TRAPIS that were less-accurate than the ADS-B estimates. Although this scenario presented an unlikely case in which both data streams were degraded, the degradation of both data streams allowed the *KalmanFilterEstimator* to be used effectively,

(a) ADS-B Raw Positions.

(b) ADS-B Estimated Positions.



(c) LAMS Raw Positions.

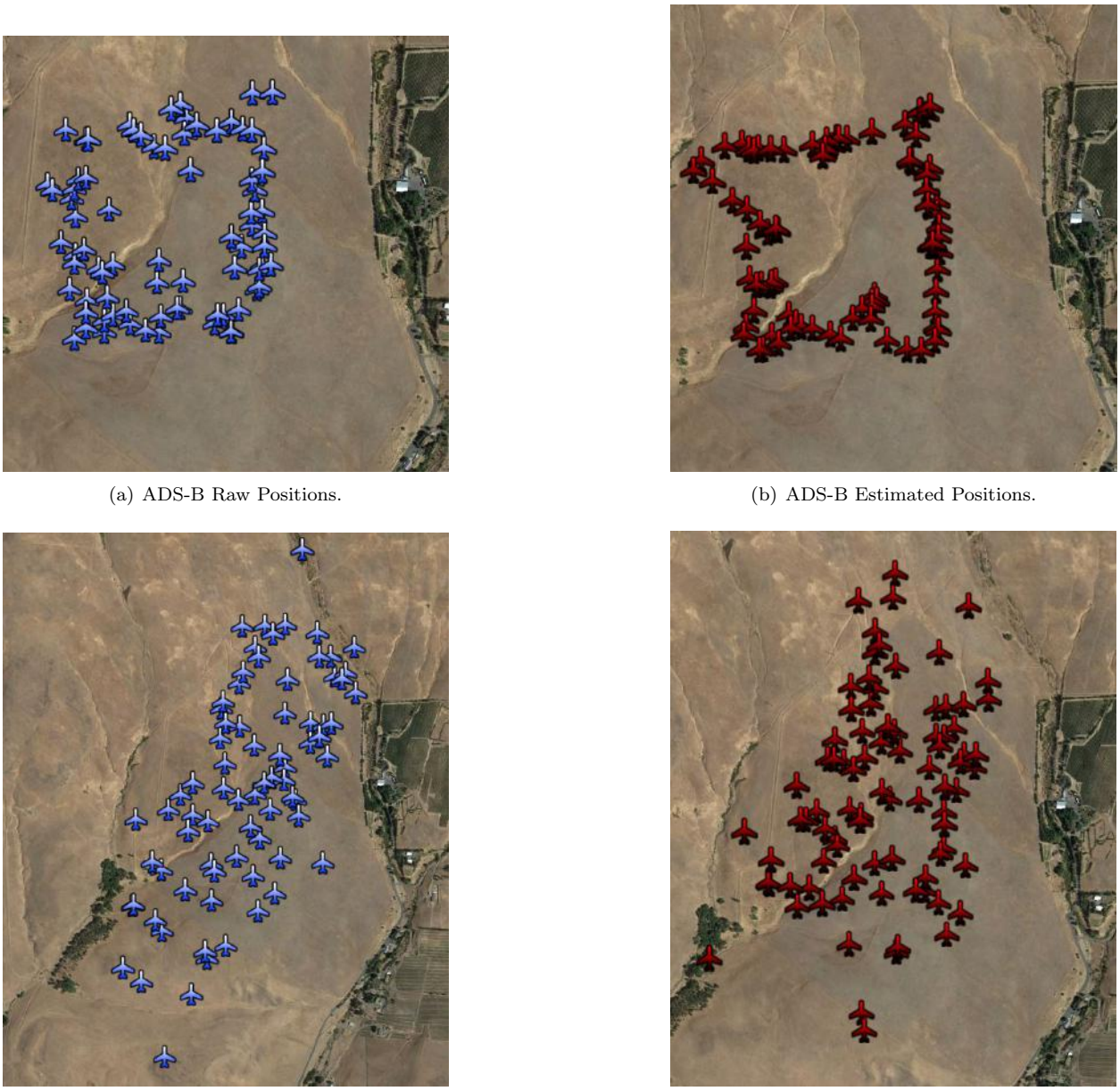
(d) LAMS Estimated Positions.

Figure 5. *DoNothingEstimator* simulated ADS-B and LAMS raw and estimated positions.

Figure 6. *SimpleFuser* simulated fused estimates.

and presented the *WeightedFuser* with disparate error values so that the ADS-B and LAMS estimates could be effectively fused into a final, single estimate.

After the simulation was conducted, the raw and estimated positions from both data streams were compared to one another as shown in Figure 7. Based on the images shown in the figure, it can be seen that the *KalmanFilterEstimator* successfully reduced the error associated with the raw data streams to create more-accurate estimates of the sUAS position. Since the *KalmanFilterEstimator* was designed to assume a



(a) ADS-B Raw Positions.

(b) ADS-B Estimated Positions.

(c) LAMS Raw Positions.

(d) LAMS Estimated Positions.

Figure 7. *KalmanFilterEstimator* simulated ADS-B and LAMS raw and estimated positions.

standard data stream error equivalent to a NACp value of 9, the estimates generated for the ADS-B data stream proved to be more accurate than the estimates generated for the LAMS data stream. This result was expected based on the design and implementation of the *KalmanFilterEstimator* algorithm.

Using the estimates generated for the ADS-B and LAMS data streams, the *WeightedFuser* weighted the position estimates associated with each data stream based on the true error of each stream in order to fuse the estimates. The fused estimates generated for the sUAS during a single orbit of the simulated flight path are shown in Figure 8. Based on the position estimates shown in the figure, it can be seen that the *WeightedFuser* accurately weighted the contributions of the ADS-B and LAMS estimates to provide a consistent, fused estimate of the aircraft position. Since the error associated with the ADS-B data stream was less than the error associated with the LAMS data stream for this simulation, the fused estimates generated by the *WeightedFuser* more closely resembled the ADS-B position estimates than the LAMS estimates.

From the results of the second simulation, it was shown that the *KalmanFilterEstimator* provided

Figure 8. *WeightedFuser* simulated fused estimates.

reasonably-accurate estimates of true aircraft positions in the presence of signal degradation. Since the *KalmanFilterEstimator* assumed a standard error corresponding to a NACp value of 9, the estimation algorithm more-accurately filtered the error out of the raw ADS-B data stream than the raw LAMS data stream. In addition to the estimation algorithm, the *WeightedFuser* data fusion algorithm provided a consistent, reasonable estimate of the sUAS position that represented the weighted average of the ADS-B and LAMS position estimates.

3. DynamicKalmanFilterEstimator and KalmanFuser

Once the initial estimation and fusion algorithms were tested through simulation, the final pair was tested. Using the same simulation scenario presented in the previous section, the *DynamicKalmanFilterEstimator* estimation algorithm and the *KalmanFuser* fusion algorithm were tested. These two algorithms were the final algorithms used for TRAPIS during the culminating flight demonstration, and thus extensive simulation testing was performed using these algorithms before flight tests began.

The raw and estimated positions from the ADS-B and LAMS data streams are shown in Figure 9. From the results presented in the figure, it can be seen that the *DynamicKalmanFilterEstimator* properly filtered the ADS-B positions and generated an accurate estimate of the rectangular sUAS flight path. Furthermore, it can be seen that the *DynamicKalmanFilterEstimator* generated a more accurate estimate from the provided LAMS positions than the *KalmanFilterEstimator*. Although error remained in the estimated LAMS positions, the magnitude of the error was significantly reduced, and the rectangular flight path can begin to be resolved from the filtered data stream.

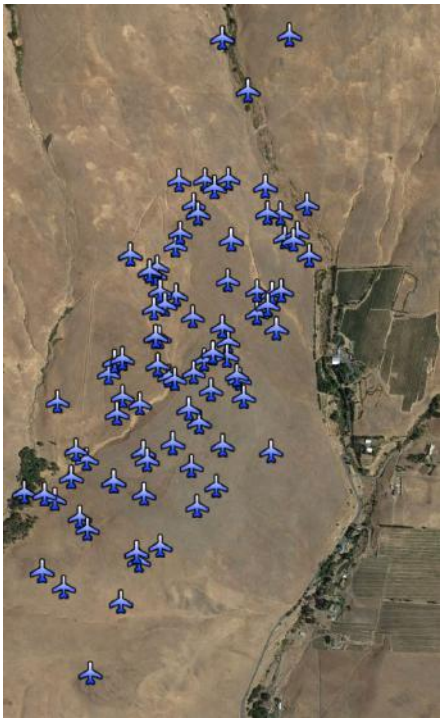
Once the position estimates were generated for the ADS-B and LAMS data streams, the *KalmanFuser* provided fused estimates to TRAPIS. Figure 10 shows the fused estimates from the simulation. The fused data stream closely resembles the filtered ADS-B estimated data stream shown in Figure 9(b), a result that was expected based on the weighting associated with the *KalmanFuser* algorithm. During simulations in which the ADS-B and LAMS data streams were degraded, the *DynamicKalmanFilterEstimator* provided the most accurate estimates of sUAS position, and the *KalmanFuser* provided the most accurate fused position estimates. This result was expected, and validated the decision to use the *DynamicKalmanFilterEstimator* and the *KalmanFuser* in the final iteration of TRAPIS.

4. GPS-Denied Operation

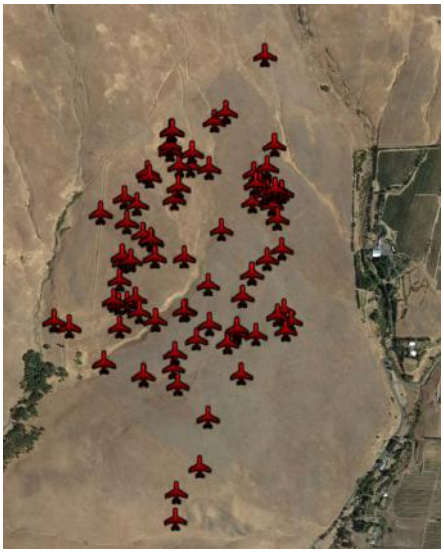
The final simulation run was designed in order to test the functionality of the estimation and fusion algorithms during GPS-denied operations. For purposes of simulating GPS denial, packets associated with the ADS-B data stream were not sent to TRAPIS during a period of 60 seconds from a simulation time of 40 seconds

(a) ADS-B Raw Positions.

(b) ADS-B Estimated Positions.



(c) LAMS Raw Positions.



(d) LAMS Estimated Positions.

Figure 9. ***DynamicKalmanFilterEstimator*** **simulated ADS-B and LAMS raw and estimated positions.**

to 100 seconds. The *DynamicKalmanFilterEstimator* and *KalmanFuser* were used for the estimation and fusion, respectively, and the total simulation time remained constant at 350 seconds.

The estimated positions from the ADS-B and LAMS data streams are shown in Figure 11. Both of the images within Figure 11 show the position data associated with the first full lap of the flight path. It can be seen that the ADS-B data stream was only available for the first quarter-lap of the flight path before the GPS was artificially denied. Although the ADS-B data stream did not continue after this point, the LAMS data stream continued for the full lap of the flight path.

As the position estimates were being generated for the GPS-denied simulation, the *KalmanFuser* data fusion algorithm generated the fused position estimates as shown in Figure 12. The fused positions shown in Figure 12 represent the fused position estimates for the first one and a half laps of the flight path. From the figure, it can be seen that fused position estimates were present for the entire first lap of the flight path,

Figure 10. *KalmanFuser* simulated fused estimates.

(a) ADS-B Estimated Positions.

(b) LAMS Estimated Positions.

Figure 11. GPS-Denied simulated ADS-B and LAMS estimated positions.

despite the loss of GPS signal. Once GPS signal was regained and the ADS-B data stream continued, the fused estimates seamlessly transitioned back to incorporating the position estimates from both the ADS-B and LAMS data streams.

The results of all simulations associated with TRAPIS showed that all of the estimation and fusion algorithms performed as expected and provided reasonable position estimates. Most importantly, the simulations showed that the data fusion algorithms provided a reasonable fused position estimate stream to TRAPIS regardless of the availability or accuracy of the ADS-B and LAMS data streams.

B. Flight Test Results

Flight testing using both manned aircraft (a Vans RV-12 and a Cessna 172SP) and unmanned aircraft was performed at the Columbia Gorge Regional Airport (KDLS). Information regarding the flight test including hardware, test cards, execution, and other details were documented in a previous publication.[10] As such, the analysis contained within this report will focus on errors between the raw and estimated data streams, to include results for both the RV-12 and the sUAS.

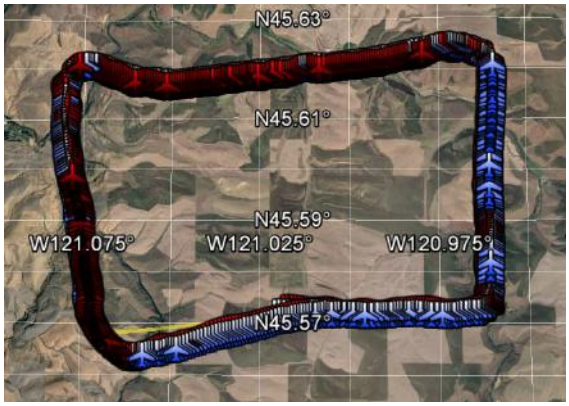
Figure 12. GPS-Denied simulated fused estimates.

1. *Manned Aircraft*

Technical problems prevented transponder information from the Cessna 172SP to be collected and as such, this section details results from the RV-12 only.

In the final iteration of TRAPIS, the *DynamicKalmanFilterEstimator* was used with the *KalmanFuser* to generate the position estimates and fused estimates, respectively. Since the ADS-B and LAMS information associated with the RV-12 was not degraded or denied, the position estimates closely resembled the raw position data. The ADS-B and LAMS estimated positions are shown with the raw positions in Figure 13. The estimated positions are shown by the red aircraft icons, while the raw positions are shown by the blue aircraft icons and the secondary GPS positions are shown by the continuous yellow lines. From the figure, it can be seen that the estimated positions closely match the raw positions for both the ADS-B and LAMS data streams. This result was expected since the data streams were not corrupted in any manner.

(a) RV-12 ADS-B raw (blue aircraft) and estimated (red aircraft) positions overlaid on data flash log track (yellow line).



(b) RV-12 LAMS raw (blue aircraft) and estimated (red aircraft) positions overlaid on data flash log track (yellow line).

Figure 13. RV-12 raw and estimated ADS-B and LAMS positions.

As the estimated ADS-B and LAMS positions were being generated, the fused estimate was simultaneously generated. Since both the estimated ADS-B and LAMS positions closely matched the raw positions, the fused estimate closely matched the raw positions as well. The fused position estimates are shown in Figure 14. The estimated positions are shown by the green aircraft icons, while the secondary GPS positions are shown by the continuous yellow line. From the data presented in the figure, it can be seen that the fused position estimates closely matched the raw and estimated positions associated with the ADS-B and LAMS

data streams.

Figure 14. RV-12 fused (green aircraft) position estimates overlaid on data flash log track (yellow line).

Based on the results associated with the RV-12 for the final flight demonstration, it was shown that the aircraft could be tracked using the TRAPIS system. Furthermore, the estimation and fusion results showed that the estimation and fusion algorithms provided accurate estimates of aircraft position based on the data associated with the ADS-B and LAMS data streams. Although the ADS-B and LAMS data streams were not degraded or denied, the results showed that the TRAPIS system performed as-desired during the final flight demonstration.

The largest error between the true aircraft track and the fused position estimates occurred at the southwest corner of the flight path, where the contribution from the wind to flight path deviations was greatest. A representation of all data streams for a single lap of the RV-12 flight path is shown in Figure 15. Based on the information shown in this figure, it can be seen that the largest difference between the ADS-B and LAMS position information occurred at the southwest corner of the flight path. At this point, the difference between the true aircraft position reported by the GPS information and the aircraft position reported by the LAMS data stream was approximately 300 meters. When compared to the total length of the flight path at 5 NM, this represents a 3.2% error. Additional differences between the ADS-B and LAMS position data were seen at the southeast corner of the flight path, where the difference between the GPS and LAMS positions was approximately 100 meters, which represents an error of 1.1% when compared to the total length of the flight path.

In order to compare the differences between the raw and estimated data streams at the southwest and southeast corners of the flight path, the corners of the plot shown in Figure 15 were expanded to generate Figure 16 and 17. Based on the results shown in Figure 16, the differences between the raw and estimated ADS-B and LAMS data streams can be seen clearly. The pink line shows the raw GPS data stream information from the HiL unit carried on-board the RV-12, while the red line and the green line show the ADS-B raw and estimated data stream information, respectively. The ADS-B results show that there was very little difference between the raw and estimated data streams, with the maximum error being approximately 10 meters. When compared to the total flight path length this represents a 0.1% error. The LAMS data, however, shows a noticeable difference between the raw and estimated data streams, with the estimated positions clearly oscillating about the raw data stream. These oscillations in the estimated data stream indicate that the covarince matrices require further tuning to ensure that the *DynamicKalmanFilterEstimator* accurately tracks aircraft with non-degraded position information. These overshoots could also be associated with differences between expected LAMS errors and what was experienced during actual flight testing. Additionally, the LAMS initially filters and smooths the incoming raw data, so the LAMS data stream represents prefiltered position information. Depending on the error that gets reported in the CAT048 messages from the LAMS station after filtering, the performance of the *DynamicKalmanFilterEstimator* could be negatively impacted, thereby affecting the TRAPIS-filtered position results. The fused estimate is shown by the dark blue line, and it can be seen that in this case the *KalmanFuser* weighted the ADS-B information much more

heavily than the LAMS information. In permissive environments where the GPS signal is not being degraded or denied, it makes sense that the ADS-B information would be used as the primary aircraft position data source, so this result was expected.

Similar results are seen in Figure 17. The raw and estimated ADS-B data streams are clearly seen to match with one another to within 10 meters. Furthermore, these data streams match the GPS information associated with the HiL unit. The raw and estimated LAMS data streams show the same trends seen in Figure 16, with the estimated data stream oscillating about the raw data stream. The fused estimate closely matches the ADS-B data stream as expected, since GPS information was not being degraded or denied. Overall, these results indicate that the covariance matrices associated with the Kalman filters will need to be tuned further to ensure that aircraft are tracked appropriately.

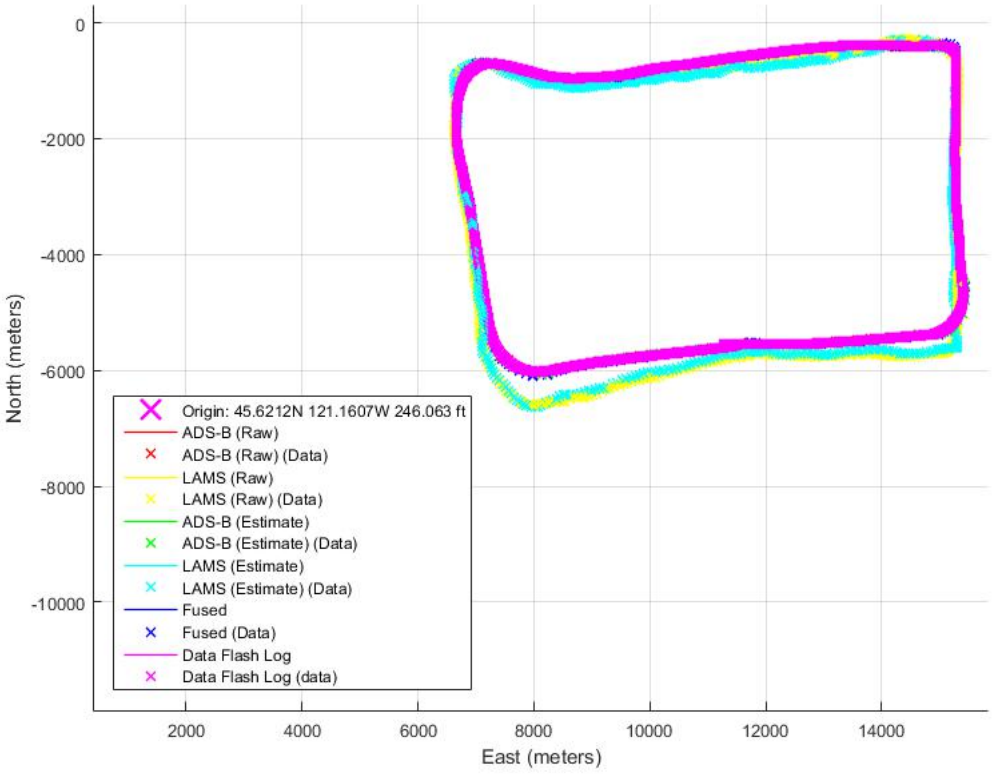


Figure 15. RV12 position data overlay for single lap of flight path.

2. Unmanned Aircraft

Although much thought went into the initial test plan for the sUAS flights, weather prevented the original plan from being flown. As a result, the sUAS was flown in a primarily east-west pattern over a total distance of approximately 100 meters. The ADS-B output remained unmodified for the beginning of the flight but was then artificially degraded for latter portions of the flight.

The estimated positions based on the ADS-B and LAMS data streams are shown in Figure 18. The estimated positions are shown by the red aircraft icons, while the raw positions are shown by the blue aircraft icons and the on-board GPS positions are shown by the continuous yellow lines.

From the figure, it can be seen that the estimated positions for the sUAS did not match the raw ADS-B and LAMS positions as closely as they did for the RV-12. The ADS-B estimated positions initially followed the raw positions, but began to diverge from the raw positions when the GPS signal was artificially degraded. This result was most likely attributed to the Kalman filter weighting associated with the *DynamicKalmanFilterEstimator* coupled with the sharp direction and velocity changes associated with the modified sUAS flight path. These sharp changes likely did not allow enough time for the filter matrices to update, and as a result the estimated positions did not closely match the raw positions. The LAMS results show similar discrepancies, and it can be seen that the estimated LAMS positions do not closely match the raw positions.

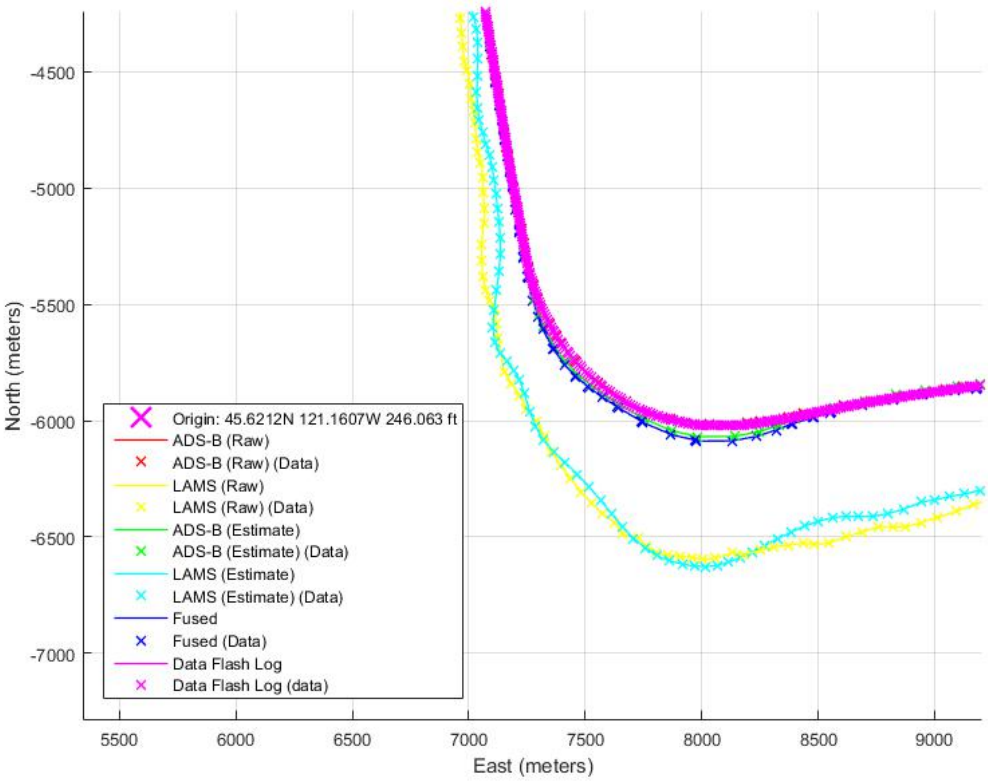


Figure 16. RV12 position data overlay for southwest corner of flight path.

The presence of position estimate errors with the LAMS data offered further information to support the conclusion that the estimation algorithm could not update quickly enough to follow the sharp direction and velocity changes associated with the updated sUAS flight path. Overall, the estimated positions associated with the LAMS data were more accurate than the estimated positions associated with the ADS-B data, but neither of the estimates accurately reflected the true aircraft positions.

The fused position estimates for the sUAS are shown in Figure 19. The figure shows that the fused position estimates more-closely followed the LAMS estimated positions than the ADS-B estimated positions. This result was expected due to the nature of the GPS unit accuracy associated with the sUAS and the presence of GPS degradation and denial during sUAS flight testing. Overall, the fused estimates captured the east-west travel of the sUAS with reasonable accuracy, especially considering the nature of the modified flight path and the associated rapid changes in direction and velocity of the aircraft.

Although the estimation and fusion results associated with the sUAS did not match the true aircraft positions as closely as the results associated with the RV-12, the fused estimates accurately captured the positions of the sUAS during the flight demonstration. In the future, additional flight tests would be valuable in which the aircraft could be flown around a flight path similar to the planned path detailed in.[10] Overall, during the flight demonstration, the estimation and fusion algorithms proved that reasonably-accurate aircraft position estimates could be generated from the provided ADS-B and LAMS data.

Another representation of all these data streams overlaid on top one another for a single lap of this modified flight path is shown in Figure 20. The average error between the ADS-B and LAMS data streams during the entire flight was approximately 100 meters, which represents a 100% error when compared to the approximate flight path length of 100 meters.

In Figure 20, the pink markers show the GPS data provided to the Sagetech XPS-TR transponder using the same type of GPS unit used in the RV-12 HiL unit as gathered from telemetry logs. The data shown by the pink markers match the ADS-B raw data shown by the red markers. Since the ADS-B information was received at longer discrete time intervals than the telemetry log reported, there are fewer raw ADS-B data points. The estimated ADS-B data stream is shown by the green markers, and it can be seen that the estimated data stream differed from the raw ADS-B stream by as much as 60 meters. Since the aircraft

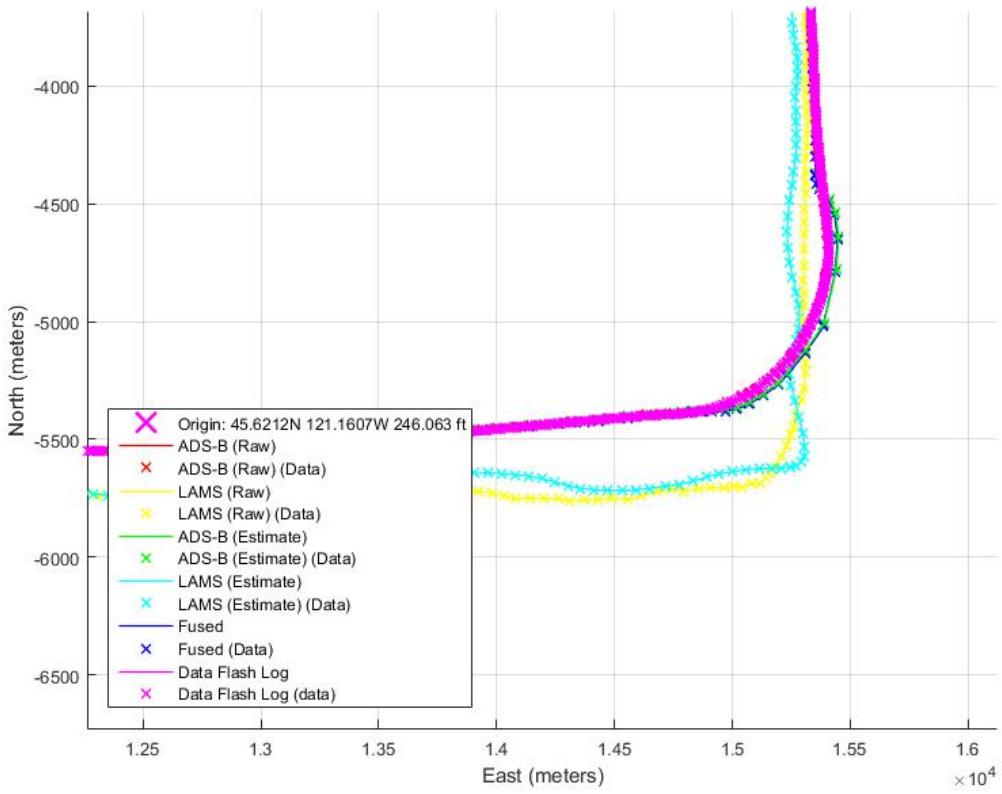
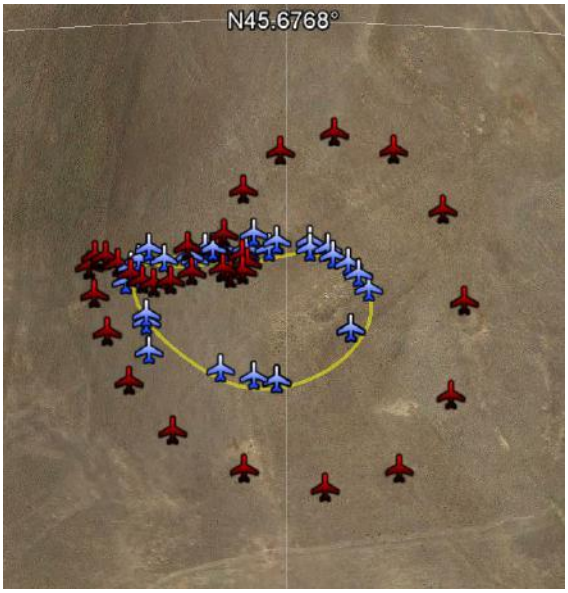
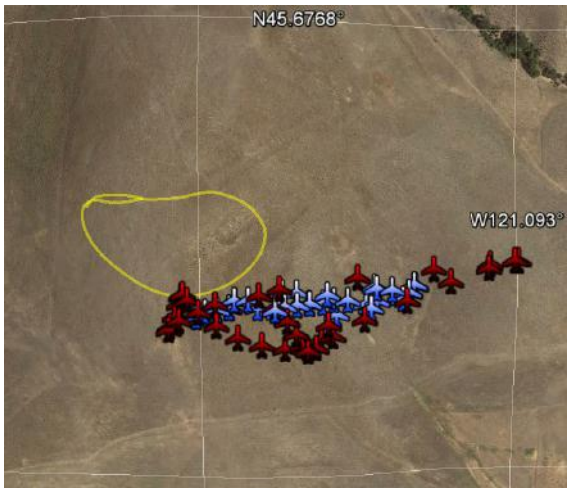


Figure 17. RV12 position data overlay for southeast corner of flight path.



(a) sUAS ADS-B raw (blue aircraft) and estimated (red aircraft) positions overlaid on data flash log track (yellow line).



(b) sUAS LAMS raw (blue aircraft) and estimated (red aircraft) positions overlaid on data flash log track (yellow line).

Figure 18. sUAS raw and estimated ADS-B and LAMS positions.

was transiting the modified flight path in a counter-clockwise manner, the estimation algorithm began to re-converge once the aircraft was moving west on the north side of the flight path. This result can likely be

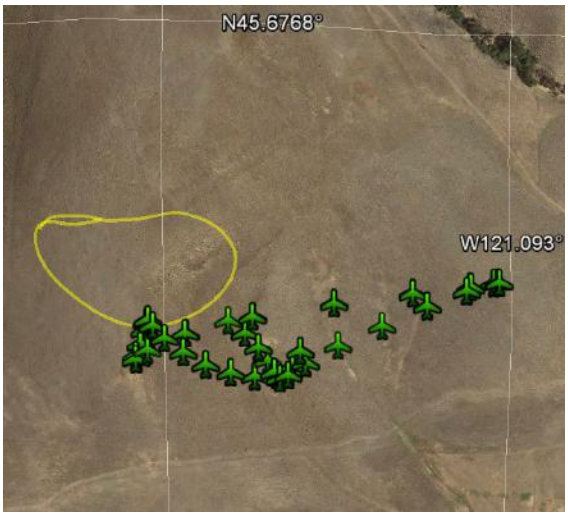


Figure 19. sUAS fused (green aircraft) position estimates overlaid on data flash log track (yellow line).

attributed to the fact that the winds during testing were blowing from the west at approximately 20 mph. As the aircraft speed increased during the eastbound portions of the flight, the estimator used the eastbound velocity. Once the aircraft turned to the west and slowed down, it took several time steps for the filter to re-converge to the slower speed and match the raw ADS-B data stream. On a larger flight path with lower winds such errors should be reduced, since additional time steps would be experienced on each leg of the flight path before the aircraft changed direction. This result ultimately indicates that the filter covariance matrices need to be tuned further to ensure that TRAPIS can accurately track both manned and unmanned aircraft.

The LAMS raw results shown in yellow show a significant bias when compared to the ADS-B results, with a constant offset of approximately 100 meters during the entire flight. This could be a result of the rotation applied to the LAMS data stream, the value of which is dependent upon antenna array orientation, magnetic variation, and other factors. Additionally, it can be seen that the estimated LAMS data stream varied from the raw data stream. This behavior was noticed for the RV-12 data, and could be a result of the way error is reported by the LAMS system after internal filtering and smoothing is applied, or could indicate the need to further adjust the TRAPIS estimation algorithm covariance matrices.

The fused data for the sUAS is shown by the dark blue markers, and it can be seen that unlike the RV-12 data, the fused data stream closely follows the LAMS data stream. Unlike the RV-12, which used an on-board WAAS GPS unit to report GPS positions to its ADS-B transponder, the sUAS used a 3DR uBlox GPS unit to report GPS positions, a unit which is not WAAS capable. It is unclear whether the error information provided by the uBlox GPS unit was different from that provided by the RV-12 WAAS GPS, but such a difference could account for the higher weight of the LAMS data stream positions for the sUAS data.

Ultimately, the RV-12 results showed that although variations between the ADS-B and LAMS data were noticed at the southwest and southeast corner of the flight path, the estimation algorithms provided reasonable estimates of the aircraft flight path, and the fusion algorithm appropriately weighted the GPS information associated with the ADS-B data stream over the information associated with the LAMS data stream. It is not known whether internal LAMS filtering and smoothing algorithms affected the reported error and hence the TRAPIS-estimated LAMS positions. The differences between the raw and estimated data streams for the sUAS indicate that additional covariance tuning must be conducted on the TRAPIS estimation algorithms if the algorithms are to be used reliably for both manned and unmanned aircraft. Furthermore, it is suspected that the uBlox GPS unit did not provide sufficient error information to TRAPIS to appropriately weight the ADS-B data stream against the LAMS data stream. Additional analysis will be required with larger sUAS test flight paths to ensure that estimation algorithm covariance matrices have been tuned properly and to ensure that appropriate GPS errors are being reported with associated ADS-B packets.

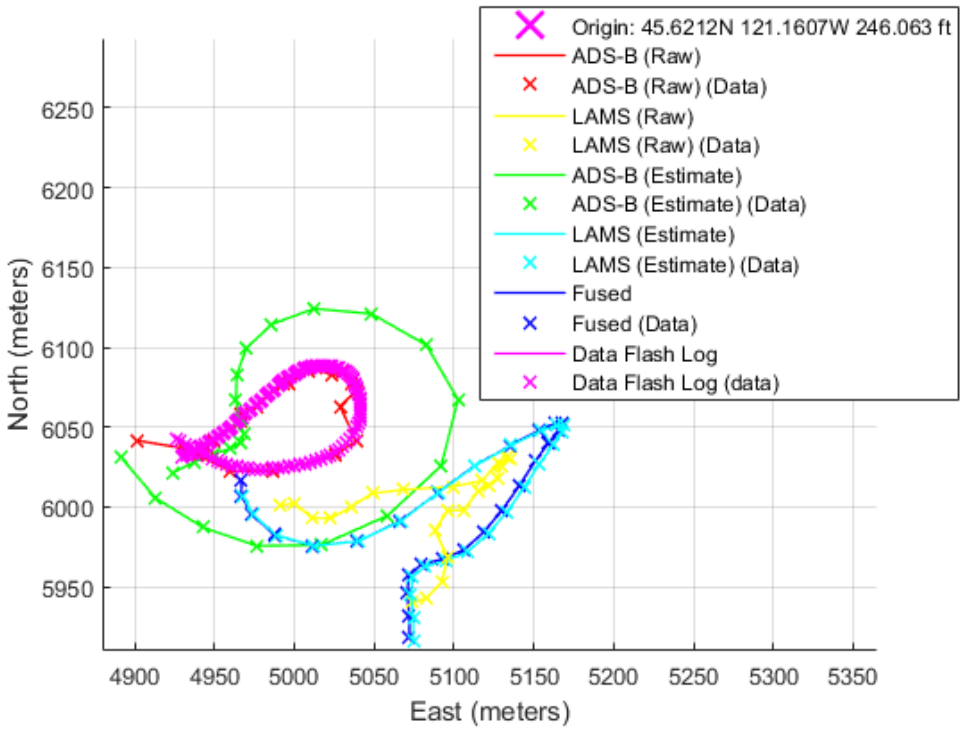


Figure 20. Leia sUAS position data overlay for single lap of flight path.

V. Conclusions and Further Research

Based on the results of the simulations and flight testing, several main conclusions can be drawn. Although the algorithms performed as-desired in simulation and in initial flight testing, the flight demonstration proved that additional work should be done to further develop and validate the performance of these algorithms.

Simulations with all of the estimation and fusion algorithms showed that each of the algorithms was best-suited to certain conditions on the ADS-B and LAMS data provided to TRAPIS. In applications where no GPS signal degradation or denial is expected, and under the assumption that the LAMS will perform without significant errors, the *DoNothingEstimator* and *SimpleFuser* should be used. As-expected, simulation showed the best results for aircraft tracking when these algorithms were used in the presence of non-degraded position information from both sources. In cases where GPS degradation or denial is experienced, the *DynamicKalmanFilterEstimator* and *KalmanFuser* should be used. Simulation results showed that in the presence of degraded or denied GPS information, these algorithms provided the most accurate estimates of aircraft position. Under no circumstances should the *KalmanFilterEstimator* be used for tracking of actual aircraft, as it was designed as an incremental step between the *DoNothingEstimator* and the *DynamicKalmanFilterEstimator* and assumes a static GPS error. Similarly, although the *WeightedFuser* provided reasonable fused estimates of aircraft position, the *KalmanFuser* provides increased functionality over the *WeightedFuser* and therefore should be used as the primary data fusion algorithm.

Initial flight testing demonstrated the utility of the *DynamicKalmanFilterEstimator* when used with an actual ADS-B data stream. Although a collocated LAMS signal was not available during initial testing, the performance of the estimation algorithm verified the accuracy of position estimates generated for an sUAS flying around a rectangular flight path similar to the one tested in simulation and planned for the final flight demonstration. Future testing should ensure that GPS-degradation is appropriately scaled to the size of sUAS flight paths to ensure that reasonable estimates can be generated when GPS signal is artificially degraded. Furthermore, code associated with artificial GPS degradation should ensure that aircraft velocities correspond to time intervals associated with degraded positions to more-accurately model actual GPS function.

Based on the results of the final flight demonstration, additional testing should be accomplished where the

sUAS is able to fly around a large, pre-defined flight path similar to the one that was originally planned. The data show that the sUAS was able to be tracked simultaneously by both the Clarity ADS-B In receiver and the LAMS, however the estimates generated from the manual sUAS flight do not realistically demonstrate the full capabilities of the estimation and fusion algorithms due to their erratic nature. While it is encouraging to know that the sUAS was tracked by both position technologies and that the LAMS track remained reasonably accurate in the presence of artificial GPS degradation and denial, further research is required to demonstrate the full functionality of the estimation and fusion algorithms.

Acknowledgments

The authors would like to thank Andy von Flotow from Hood Technology for sponsorship of the research, guidance, and contributions to the research vision.

Advanced Navigation and Positioning Corporation (ANPC) was instrumental during the progression of the project. Michael Van Dooren provided invaluable technical support and assistance navigating ANPC software. Karl Winner provided technical support and assistance interfacing UW software with the ANPC LAMS during flight testing. Jeff Mains also provided support of the project.

Sagetech Corporation, particularly Kelvin Scribner, James Davis, and Tom Furey, contributed technical expertise and equipment.

Rolf Rysdyk from Insitu assisted with guidance and flight testing support.

Many members of the UW's Autonomous Flight Systems Laboratory contributed to the research including Zhenzhen Su, Alec Bueing, Andrew Jacobson, Emil Caga-anan, Henry Qin, Marissa Reid, Ryan Valach, Scott An, Shida Xu, Zach Williams.

This research was sponsored by the Joint Center for Aerospace Technology Innovation (JCATI) as part of the project entitled "Specialization, Testing, and Integration of NextGen Technologies on Unmanned Aerial Systems".

References

[1]Lum, C. W. and Vagners, J., "A Modular Algorithm for Exhaustive Map Searching Using Occupancy Based Maps," *Proceedings of the 2009 Infotech@Aerospace Conference*, Seattle, WA, April 2009.

[2]Lum, C. W., Vagners, J., and Rysdyk, R. T., "Search Algorithm for Teams of Heterogeneous Agents with Coverage Guarantees," *AIAA Journal of Aerospace Computing, Information, and Communication*, Vol. 7, January 2010, pp. 1–31.

[3]Lum, C. W., Gardner, S., Jordan, C., and Dunbabin, M., "Expanding Diversity in STEM: Developing International Education and Research Partnerships in a Global Society," *Proceedings of the American Society for Engineering Education 123rd Annual Conference & Exposition*, June 2016.

[4]Lum, C. W., Mackenzie, M., Shaw-Feather, C., Luker, E., and Dunbabin, M., "Multispectral Imaging and Elevation Mapping from an Unmanned Aerial System for Precision Agriculture Applications," *Proceedings of the 13th International Conference on Precision Agriculture*, St. Louis, MO, August 2016.

[5]Lum, C. W., Summers, A., Carpenter, B., Rodriguez, A., and Dunbabin, M., "Automatic Wildfire Detection and Simulation Using Optical Information from Unmanned Aerial Systems," *Proceedings of the 2015 SAE Aerotec Conference*, Seattle, WA, September 2015.

[6]Lum, C. W., Rysdyk, R. T., and Pongpunwattana, A. A., "Autonomous Airborne Geomagnetic Surveying and Target Identification," *Proceedings of the 2005 Infotech@Aerospace Conference*, AIAA, Arlington, VA, September 2005.

[7]Volpe, J. A., "Vulnerability Assessment of the Transportation Infrastructure Relying on the Global Positioning System," Tech. rep., National Transporation Systems Center, 2001.

[8]Hu, H. and Wei, N., "A Study of GPS Jamming and Anti-Jamming," *2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS)*, 2009.

[9]Brandon, J., "GPS Jammers Illegal, Dangerous, and Very Easy to Buy," *Fox News Technology*, 2010.

[10]Lum, C. W., Larson, R. S., Handley, W., Lui, S., and Caratao, Z., "Flight Testing an ADS-B Equipped sUAS in GPS-Denied Environments," *Proceedings of the AIAA Flight Testing Conference*, Denver, CO, June 2017.

[11]Martel, F., Shultz, R., W, S., Wang, Z., and Czarnomski, M., "Unmanned Aircraft Systems Sense and Avoid Avionics Utilizing ADS-B Transceiver," *AIAA Aerospace Conference*, 2009.

[12]Lai, C., Ren, Y., and Lin, C., "ADS-B Based Collision Avoidance Radar for Unmanned Aerial Vehicles," *Microwave Symposium Digest*, 2009.

[13]Stark, B., Stevenson, B., and Chen, Y., "ADS-B for Small Unmanned Aerial Systems: Case Study and Regulatory Practices," *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2013.

[14]Ramasamy, S., Sabatini, R., and Gardi, A., "Avionics Sensor Fusion for Small Unmanned Aircraft Sense-and-Avoid," *Meteorology for Aerospace*, 2014.

[15]Handley, W., *Two NextGen Air Safety Tools: An ADS-B Equipped UAV and a Wake Turbulence Estimator*, Master's thesis, University of Washington, Seattle, WA, June 2016.

[16]Larson, R. S., *sUAS Position Estimation and Fusion in GPS-Degraded and GPS-Denied Environments using an ADS-B Transponder and Local Area Multilateration*, Master's thesis, University of Washington, Seattle, WA, March 2017.

[17]Mohleji, S. and Wang, G., "Modeling ADS-B Position and Velocity Errors for Airborne Merging and Spacing in Interval Management Application," Tech. rep.

[18]Zarchan, P., Musoff, H., and Lu, F., *Fundamentals of Kalman Filtering: A Practical Approach*, AIAA, February 2009.

[19]GeoMidpoint, "Geographic Midpoint Calculation Methods," 2017.

[20]Drolet, L., Michaud, F., and Cote, J., "Adaptable Sensor Fusion Using Multiple Kalman Filters," *Intelligent Robots and Systems*, February 2000.