# EnAcT: Generating Aircraft Encounters using a Spherical Earth Model

James A. Ritchie III
*Modeling and Simulation*
*Federal Aviation Admin.*
Atlantic City, NJ 08405
james.ritchie@faa.gov

Andrew J. Fabian
*Modeling and Simulation*
*Federal Aviation Admin.*
Atlantic City, NJ 08405
andrew.fabian@faa.gov

Nidhal C. Bouaynaya
*Elec. and Comp. Eng.*
*Rowan University*
Glassboro, NJ 08028
bouaynaya@rowan.edu

Mike M. Paglione
*Software & Systems*
*Federal Aviation Admin.*
Atlantic City, NJ 08405
mike.paglione@faa.gov

***Abstract*—There is ongoing research at the Federal Aviation Administration (FAA) and other private industries to examine a concept for delegated separation in multiple classes of airspace to allow unmanned aircraft systems (UAS) to remain well clear of other aircraft. Detect and Avoid (DAA) capabilities are one potential technology being examined to maintain separation. To evaluate these DAA capabilities, input traffic scenarios are simulated based on either simple geometric aircraft trajectories or recorded traffic scenarios and are replayed in a simulator. However, these approaches are limited by the breadth of the traffic recordings available. This paper derives a new mathematical algorithm that uses great circle navigation equations in an Earth spherical model and an accurate aircraft performance model to generate realistic aircraft encounters in any airspace. This algorithm is implemented in a program called Encounters from Actual Trajectories (EnAcT) and uses a number of user inputs defining the encounter events, called encounter properties. Given these encounter properties, the program generates two 4-dimensional flight trajectories that satisfy these properties. This encounter generator could be used to evaluate DAA systems as well as initiate research in automation for conflict detection and resolution.**

***Index Terms*—Aerospace testing, algorithm design and analysis, systems modeling, unmanned aircraft systems.**

## I. Introduction

The Federal Aviation Administration (FAA) and private industry collectively within RTCA Special Committe (SC) 228 are working to define the minimum operational performance standards (MOPS) for unmanned aircraft systems (UAS) to allow for safe integration of UAS into the National Airspace System (NAS). Part of defining the MOPS is ensuring the Detect and Avoid Algorithms (DAA) on UAS function properly to adhere to the See and Avoid requirement of the 14 CFR 91 - General Operating and Flight Rules imposed by the FAA. The FAA Federal Aviation Regulation (FAR) requirement of See and Avoid is essential to safe flight in manned aircraft. See and Avoid refers to the ability of the pilot to visually scan the surrounding airspace and determine possible risks. It is then the pilot's responsibility to avoid these risks by following the rules defined in the FAR or by any means in the case of an emergency. Thus, UASs without a pilot on board the aircraft are required to carry DAA systems that comply with 14 CFR 91.

This work was supported by an FAA Cooperative Agreement No. 15-G-001 awarded to Rowan University.

The DAA system provides preventive, corrective, and warning guidance to the UAS pilot to assist in preventing loss of separation with other aircraft. These systems have significant technical challenges in establishing and maintaining the relative position of one or more external threats (i.e. aircraft) during an encounter. The entire DAA system consists of surveillance sensors used to detect an intruder, tracker to fuse and filter multiple sources of information to provide one single track to the DAA algorithm.

To ensure DAA systems are capable of detecting other aircraft, the systems need to be tested. Evaluating these systems requires scenarios in which two or more aircraft have a close encounter with each other. An encounter event can be an event that has the two or more aircraft lose legal separation with each other, called a conflict, or can be close to loss of legal separation, which is called an encounter. Encounter/Conflict definitions change with the different types of airspaces in the NAS. In EnRoute (Class A) airspace, separation distances are usually greater than those in terminal (Class B, C, D, E) airspaces. The DAA system must be tested with cases of encounters and conflicts in all airspace types to ensure they can correctly identify when the UAS may stray too close to another aircraft. This work focuses on encounters with two aircraft.

In the NAS, no conflicts exist since the controllers keep the traffic separated thus preventing the use of unmodified recorded air traffic. Scenarios need to be generated to be able to simulate these desired encounters. Current capabilities of generating rely on simulated input traffic scenarios based on either simple linear aircraft trajectories or recorded traffic scenarios [1]–[5]. However, these approaches are either limited by the simplicity of their models or by the breadth of the traffic recordings available. Thus, there is a need for more realistic and complex algorithms to generate a large set of user specified traffic scenarios.

In the literature, the generation of conflicting aircraft trajectories can be divided into three categories: (1) fitting probabilistic distributions and statistical models over the range of encounter variables [6], [7]; (2) using recorded air-traffic data through time-shifting the flights [1]–[5]; and (3) ad-hoc generation of aircraft trajectories for the main purpose of studying conflict detection. [8]–[12]. Each of these approaches considers different assumptions and addresses different concerns with regard to what type of conflicts are generated and why they are needed.

The encounter model in [6], developed by the Massachusetts Institute of Technology's (MIT) Lincoln Laboratory, describes a probabilistic aircraft encounter model. This model is currently limited to generating encounters in the EnRoute environment, where flights are typically cruising and not changing altitude often. Also, the encounters are all generated at random (according to the probabilistic properties), and you cannot specify what kind of encounter you would like to have. MIT Lincoln Laboratory is currently working on a new encounter model, but it is not ready for generating encounters at the time of this publication.

The FAA has traditionally generated aircraft conflicts and encounters through the use of time-shifting the flights in a recorded air traffic scenario [1], [2]. These algorithms consider the recorded flight data of aircraft that have flown in the NAS, and shift the position of the aircraft in time to induce encounters. The resulting trajectories contain the same physical flight position data as the original recorded traffic data, but the aircraft fly these tracks at new times. A genetic algorithm was implemented to determine the optimal time-shift values for each flight in the scenario [3]. Even though more encounter criteria have been considered [4] and the algorithm has been improved since its original version [5], current algorithms only generate encounters from flights that have existed in the NAS. This prevents the user from specifying the exact parameters for each conflict, which is useful for testing specific conditions. These algorithms have only been tested on EnRoute (Class A) airspace, and not on data from the terminal environment. These methods would need to be evaluated for use in terminal environment and have had limited use beyong en route airspace.

Other studies did not focus on the problem of aircraft conflict generation, but considered the issue of conflict detection or resolution [8]–[12]. Aircraft conflicts in these studies are typically fixed, and have only examined a subset of conflict types. The trajectories of these aircraft are typically straight, and use coordinates based on a flat-Earth system. While the encounter properties are typically specified, the method in which they are created is not conducive to generating millions of encounters in a reasonable time span.

This paper addresses the need to simulate encounter traffic events for the ultimate aim of testing the DAA capabilities of UAS in all airspace types. This algorithm output will cover gaps that exist in the types of encounters that are needed for testing. Specifically, we propose a new encounter generator algorithm that uses a spherical Earth model and performance parameters from EuroControl's Base of Aircraft Data (BADA) [13] to generate pairs of 4D trajectories that satisfy a specified set of encounter event properties. Phase 2 MOPS is looking at the incorporation of UAS in the terminal airspace, which this paper intends to address. The generator is formulated as a constrained optimization problem in a spherical coordinate system. A program was developed to use this algorithm, called Encounters from Actual Trajectories (EnAcT).

This paper is organized as followed: In Section II, the encounter generator is mathematically formulated as a constrained optimization problem in a spherical Earth model. Section III discusses the implementation of the algorithm within the EnAcT program. Section IV discusses the process in which the input for the algorithm was set up, and Section V discusses the results of testing the algorithm. Finally, concluding remarks are given in Section VI.

## II. Mathematical Formulation and Algorithm

We define a conflict between two aircraft: an ownship and an intruder. The time-varying latitude, longitude, altitude, and heading of each aircraft are, respectively, denoted by $\phi_i(t)$, $\lambda_i(t)$, $h_i(t)$, and $\theta_i(t)$, where $i \in \{0, 1\}$ represents the ownship when zero and the intruder when one. We also define $H(t)$ and $V(t)$ to be the horizontal spherical distance and the vertical distance, respectively, between the aircraft. $\Delta\theta(t)$ is the difference in headings of the aircraft at time instant $t$.

Tangential and normal velocities of the ownship and intruder are looked up from the Base of Aircraft Data (BADA) aircraft performance model, which is an aircraft characteristics table based on aircraft type and altitude. They are represented here as tangential velocity $v_{i,\mathrm{H}}$ and normal velocity $v_{i,\mathrm{V}}$, and are assumed constant for the duration of the conflict model.

The radius of the spherical Earth is denoted by $R$. It is sometimes more convenient to refer to distances and velocities in their angular forms by dividing through by $R$. Let $\delta(t) = \frac{H(t)}{R}$ and $\omega_i = \frac{v_{i,\mathrm{H}}}{R}$.

For simplicity and without loss of generality, we will assume that the closest point of approach (CPA) occurs at time $t = 0$. The constraints of a given conflict, specified by the user, are: i) the minimum horizontal and vertical distances at CPA, denoted as $H_0$ and $V_0$, respectively; ii) the encounter angle at CPA, which is the difference in initial headings, denoted by $\Delta\theta_0$; and iii) the location and heading of the ownship at CPA. Thus, we have the following conflict requirements: $\phi_{00}$, $\lambda_{00}$, $h_{00}$, and $\theta_{00}$. Observe that we can uniquely compute $\theta_{10}$, the initial heading of the intruder at CPA, from $\theta_{00}$ and $\Delta\theta_0$. Similarly, knowing $h_{00}$ and $V_0$ uniquely defines $h_{10}$. The aim is to construct pairs of four dimensional trajectories, $(t, \phi_i(t), \lambda_i(t), h_i(t))$, representing the ownship and intruder, that produce the conflict event specified by the three CPA constraints.

In the sequel, we will drop the time-dependency notation for clarity, i.e., we will write $\phi$ to denote the time-varying latitude $\phi(t)$, and similarly for all other time-dependent variables. We parameterize the trajectories using the unit normal vector,

$$\boldsymbol{N} = (\cos\phi\,\cos\lambda,\ \cos\phi\,\sin\lambda,\ \sin\phi). \qquad (1)$$

The North and East unit vectors at N are given, respectively, by

$$\boldsymbol{\nu} = (-\sin\phi\,\cos\lambda,\ -\sin\phi\,\sin\lambda,\ \cos\phi), \qquad (2)$$

and

$$\boldsymbol{\epsilon} = (-\sin\lambda,\ \cos\lambda,\ 0). \tag{3}$$

Let

$$\boldsymbol{D} = \boldsymbol{\nu}\cos\theta\ + \boldsymbol{\epsilon}\sin\theta \tag{4}$$

be the directional unit vector. For any central angle $\sigma_0(t) = \omega_0 t$ traveled, the temporal parameterization of the ownship is given by

$$\boldsymbol{N}_0(t) = \cos(\sigma_0(t))\, \boldsymbol{N}_{00} + \sin(\sigma_0(t))\, \boldsymbol{D}_{00}. \tag{5}$$

If we have a fixed start point and a fixed arc-distance to travel $\delta_0$, we have an equation for the locus of all points, in unit normal vector notation, equidistant from the ownship at CPA. We need to place the intruder on this circle at CPA. The corresponding bearing vector is:

$$\boldsymbol{B}_0(\beta_0) = (\cos\beta_0\, \boldsymbol{\nu}_{00} + \sin\beta_0\, \boldsymbol{\epsilon}_{00})\,, \tag{6}$$

and the vector representation of the intruder at CPA as a function of bearing $\beta_0$ is:

$$\boldsymbol{N}_{10}(\beta_0) = \cos\delta_0\, \boldsymbol{N}_{00} + \sin\delta_0\, \boldsymbol{B}_0(\beta_0). \tag{7}$$

The trajectory of the intruder can be formulated in a similar manner. We, then, have

$$\boldsymbol{N}_1(\beta_0, t) = \cos\sigma_1(t)\, \boldsymbol{N}_{10}(\beta_0) + \sin\sigma_1(t)\, \boldsymbol{D}_{10}(\beta_0), \tag{8}$$

where $\boldsymbol{N}_1(\beta_0, t)$ is the unit normal vector of the intruder with bearing $\beta_0$ at time $t$, $\sigma_1(t) = \omega_1 t$ is the angular distance traveled by the intruder, and $\boldsymbol{D}_{10}$ is the directional vector of the intruder at CPA found by solving $\phi_{10}$ and $\lambda_{10}$ from $\boldsymbol{N}_{10}$, using $\theta_{10}$ and substituting these values into (4).

In spherical coordinates, we can calculate the horizontal distance between the two aircraft using the unit normal vector of the intruder and the ownship. This means we will have to add $\beta_0$ to our parameterization of $H$ while still maintaining that $H(\beta_0, 0) = H_0$. The horizontal distance along a great circle is given by:

$$H(\beta_0, t) = 2R\, \sin^{-1}\left(\frac{\|\boldsymbol{N}_1(\beta_0, t) - \boldsymbol{N}_0(t)\|}{2}\right). \tag{9}$$

The vertical distance is assumed to be linear:

$$V(t) = (v_{1,\mathrm{V}} - v_{0,\mathrm{V}})\, t + V_0 = \Delta v_V t + V_0. \tag{10}$$

Knowing the horizontal and vertical distances, the squared distance between the two aircraft is:

$$H(\beta_0, t)^2 + V(t)^2. \tag{11}$$

We need to minimize this squared distance at $t = 0$. Setting the derivative to zero, we obtain

$$H_0\, \left.\frac{\partial H(\beta_0, t)}{\partial t}\right|_{t=0} = -V_0 \Delta v_{\mathrm{V}} \tag{12}$$

Computing the partial derivative $\frac{\partial H(\beta_0, t)}{\partial t}$, we find the equation that the bearing angle must satisfy to enforce all CPA constraints:

$$-\frac{V_0 \Delta v_{\mathrm{V}}}{H_0} = v_{1,\mathrm{H}} \frac{f(\beta_0)}{\sqrt{1 - g^2(\beta_0)}} - v_{0,\mathrm{H}}\, \cos(\beta_0 - \theta_{00}), \tag{13}$$

where $f(\beta_0)$ is:

$$\begin{aligned} f(\beta_0) = \cos\phi_{00}\, \cos\delta_0\, \cos\theta_{10}\, \cos\beta_0 \\ - \sin\phi_{00}\, \sin\delta_0\, \cos\theta_{10} \\ + \cos\phi_{00}\, \sin\theta_{10}\, \sin\beta_0 \end{aligned} \tag{14}$$

and $g(\beta_0)$ is:

$$g(\beta_0) = \cos\phi_{00}\, \sin\delta_0\, \cos\beta_0 + \sin\phi_{00}\, \cos\delta_0. \tag{15}$$

Observe that the above equation has only one unknown: the bearing, $\beta_0$, which satisfies all CPA constraints. This will give us the starting locations for each aircraft in the general case. It guarantees that the aircraft are at the required distances, at a specified encounter angle and single set of speeds. This constraint can be numerically solved, using an accepted method like the Newton-Raphson algorithm.

## III. EnAcT Program

The algorithm developed in Section II is implemented in a Java program known as Encounters from Actual Trajectories (EnAcT). This program can be described with the pseudo-code given in Algorithm 1:

**Algorithm 1** EnAcT Program Run Sequence

```
 1: Read in inputs
 2: while There exists defined encounters to generate or
    required number not met do
 3:     Calculate bearing at CPA based on given inputs
 4:     if Bearing exists then
 5:         Determine heading of aircraft at CPA
 6:         Build trajectories of each aircraft
 7:         Print trajectories to file
 8:     else
 9:         Log that bearing could not be found
10:         Skip this set of inputs
11:     end if
12: end while
```

This program accepts inputs in two formats: 1) each encounter is defined with each of its properties explicitly by the user; or 2) probability distributions for each encounter property is defined along with the number of encounters to generate. The probability distributions can be given so that all encounters generated will fit the given distributions. Either set of inputs can generate an unlimited number of encounter scenarios with synthetic trajectories that match the given performance model (BADA). The properties define the encounter at the closest point of approach (CPA), and are:

- Horizontal Separation Distance at CPA
- Vertical Separation Distance at CPA
- Latitude of Ownship at CPA
- Longitude of Ownship at CPA
- Encounter Angle at CPA
- Aircraft types
- Vertical Phase of flight for both aircraft
- Number of conflicts (if giving distributions)

The Java program uses Equation 13 to determine the bearing between two aircraft at the closest point of approach (CPA) based on the given inputs, which is step 3 in Algorithm 1. If a bearing can be found, the program then uses a trajectory engine, in this case BADA, to generate the trajectories of the two aircraft based on the initial starting point and the properties at CPA. The trajectories are then written out into a Comma Separated Value (CSV) file. If the bearing cannot be found, the attempt is skipped based on the given inputs, and is logged.

## IV. Process for Generating EnAcT Inputs

EnAcT requires the distributions for each conflict property to be able to generate millions of encounters that fall into these distributions. To generate these distributions that are realistic to what can be seen in the NAS, the Modeling and Simulation Branch developed a process to determine these distributions. A software tool that the branch developed, called the Trajectory Conflict Probe [14], is the first step of the process. It inputs recorded flight plans and surveillance position reports and predicts when two aircraft are on a path that could violate these separation distances. Air traffic controllers issue clearances to alter one or both of the aircraft's paths to resolve these conflicts before they occur. The tool records both the initial event and when the conflict is resolved. This allows the team to determine theoretical encounters from recorded air traffic data that had no actual encounters occuring due to air traffic controller intervention. Figure 1 shows the flow of the Trajectory Conflict Probe program.

As Fig. 1 illustrates, the Trajectory Conflict Probe performs a pairwise analysis of flights from a given air traffic scenario. It considers each aircraft's flight plans, amendments, issued controller clearances, and surveillance position reports. The software first runs a gross filter to check rapidly for approximate temporal and spatial overlap of the pair of aircraft, based on the given position reports, or track data, of the two aircraft. This filter removes flight pairs that have no potential spatial overlap, cutting down the processing needed. If the pair passes this initial gross filter, the tool calls the Trajectory Predictor, which generates a predicted trajectory at each incremental time step that both aircraft have available track data. These predicted trajectories pass through a sequence of filters in order to determine if any predicted conflicts (violations of separation standards) or encounters (larger separation events) occur.

Once an encounter is detected, the algorithm checks to see if the encounter occurs within the user defined predicted warning time. If it does, it immediately records the encounter. If the detected encounter is predicted to occur outside of the warning time, the algorithm checks the conflict detection frequency. The conflict detection frequency is the number of times the conflict probe predicts the conflict for each track instantiated trajectory before the predicted event. The user defines a percentage of the track instantiated trajectories that have a positive detection of the predicted event for it to be recorded as an encounter. The program continues to track the prediction event for each track instantiated trajectory. Once

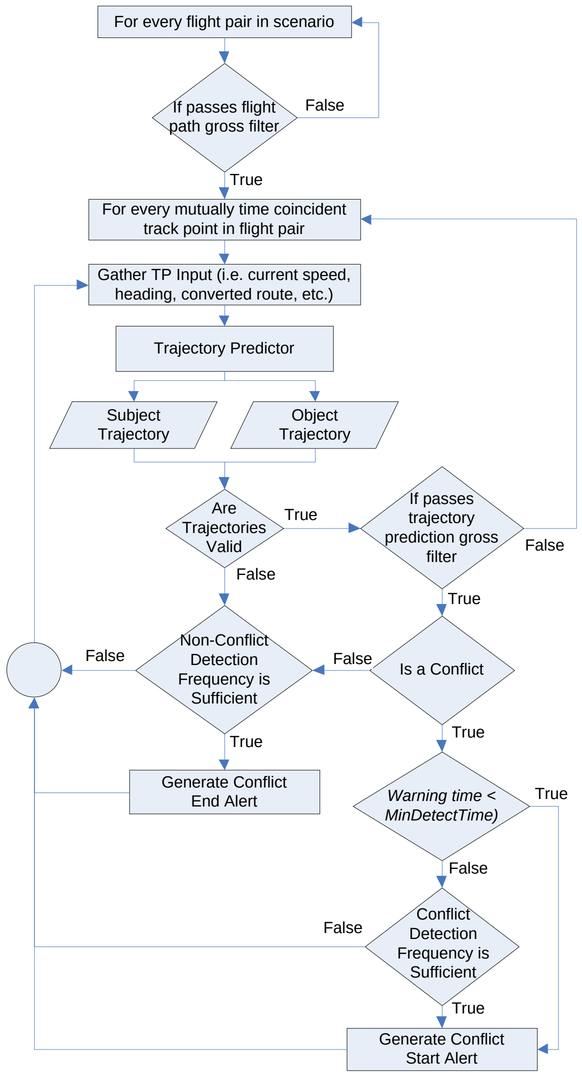
For every flight pair in scenario
If passes flight path gross filter
False
True
For every mutually time coincident track point in flight pair
Gather TP Input (i.e. current speed, heading, converted route, etc.)
Trajectory Predictor
Subject Trajectory
Object Trajectory
Are Trajectories Valid
True
False
If passes trajectory prediction gross filter
False
True
False
Non-Conflict Detection Frequency is Sufficient
False
Is a Conflict
True
True
Generate Conflict End Alert
Warning time < MinDetectTime)
True
False
False
Conflict Detection Frequency is Sufficient
True
Generate Conflict Start Alert

Fig. 1. Flowchart of the Trajectory Conflict Probe algorithm [14]. This program analyzes each flight pair that has paths that are close enough to potentially cause a conflict. Using a trajectory engine and the relevant clearance information, this program can predict where a conflict could have occurred if it were not for controller intervention, as long as the generated trajectories are valid.

the program detects the encounter has ended, it starts to count these events from each new trajectory. If this percentage is met, the encounter has ended and the results are recorded. These checks are the frequency checks referenced in Fig. 1 flowchart and act as stability filters for the encounter predictions. The program outputs the resulting encounters it detects as alert messages stored in a relational database. These alert messages include the time of detection, the location of the predicted event, and the location of the aircraft when the even was predicted.

Using the Trajectory Conflict Probe tool, the characteristics of potential NAS conflicts and encounters can be estimated and matched by the conflict generation algorithm. The conflict properties that were considered in this study are: Conflict Location, Horizontal Separation Distance, Vertical Separation Distance, Encounter Angle, Aircraft Type, and Vertical Phase of Flight.

The FAA's Operations Network (OPSNET) was to utilized to select a day in the NAS with above average traffic count but minimal delays, including from weather [15]. Conflict count is proportional to traffic count, so a large traffic count provides a significant sample for the measurement of the conflict properties. By minimizing the delay as well, the traffic and associated distributions are expected to exhibit minimal reroutes thus reducing the variability in the generated trajectories. The study resulted in selection of January 31, 2016 representing a peak winter traffic day with minimal delays

NASQuest was utilized to retrieve the 24-hour flight data from ERAM for each of the 20 Air Route Traffic Control Centers (ARTCCs) in the NAS. Analysts processed the Common Message Set (CMS) for each ARTCC using the Modeling and Simulation Branch's software suite. This suite analyzes the CMS messages and inserts the data into a series of relational tables within their database. A set of related tables is referred to as a scenario.

With the historic flight data retrieved, the Trajectory Conflict Probe software predicts the conflict events that would have occurred without controller intervention. Alert messages are created based on the predicted conflict events. Each message created contains detailed information describing the specific aircraft predicted to be in a specified conflict or encounter, the location, and other properties of the event. This data is stored in a set of tables in a relational database.

Upon completion of the Trajectory Conflict Probe software, predicted conflict alerts and their properties are stored within a database. Trajectory Conflict Probe produces multiple messages for a predicted event throughout the duration of the event. The conflict property information is contained in the first message of the alert. The information in this message is collected and used to create distributions for each conflict property.

The following subsections discuss the conflict properties studies, the K-means process used for determining the distribution of the conflict location property, and the process used for determining the empirical distributions of the other conflict properties.

### A. *Conflict Location*

The location of a conflict event is described by three parameters: latitude, longitude, and altitude. This three-dimensional location of the predicted start of the conflict event is examined in a two-step process: the horizontal position and the vertical position of the aircraft during the conflict.

First, the horizontal position of each aircraft in an event, consisting of latitude and longitude, is investigated. A K-means clustering algorithm is used to analyze the horizontal location. Clusters are created based on the latitude and longitude of the aircraft involved in a predicted conflict event. A cluster represents a section of airspace where a group of

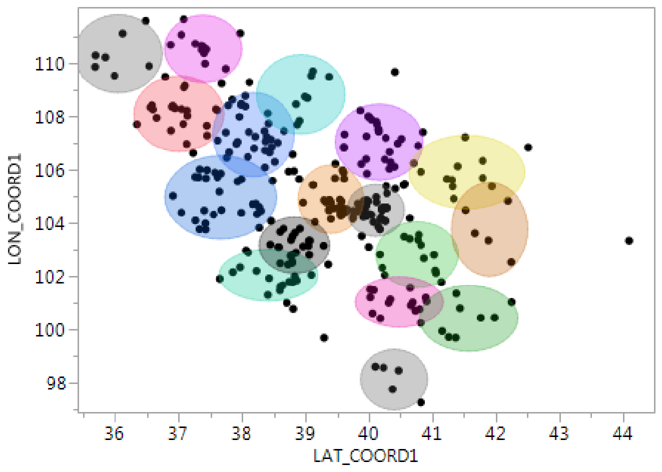

Fig. 2. *K*-means cluster diagram representing the horizontal location of flight conflicts in ZDV. The colored circles represent the clusters generated by the *K*-means cluster algorithm, and each dot represents the location of an aircraft conflict.

predicted conflict events occurs. A two-dimensional normal distribution is used to model the center and spread of each cluster. The centroid location (mean) is estimated by the mean latitude and longitude of the aircraft at the beginning of the conflict. The standard deviation represents the spread of the locations for the given cluster.

For this study, the latitude and longitude of the subject aircraft in each predicted conflict event is used. The *K*-means clustering algorithm is set to produce 20 clusters for a given airspace. Each cluster's mean and standard deviation is stored for input to the conflict generator software. Figure 2 shows the results of the *K*-means cluster algorithm of the horizontal position of conflicts in the Denver Center (ZDV).

Each colored circle in Fig. 2 forms a cluster produced by the K-means algorithm. The solid dots are the locations of predicted conflict events that the K-means algorithm uses to generate these clusters. Latitude coordinates are on the ordinate while the longitude coordinates are on the abscissa. Each cluster is modeled by a two-dimensional normal distribution with point estimates for the mean latitude and longitude dimensions and associated standard deviations. These models, in combination with the altitude distribution, estimated separately, represent the distribution of the location of conflict events throughout ZDV.

The altitude of the conflict was measured separately from the horizontal position as an empirical distribution. The altitude data was split into bins of 2500 ft. The results from ZDV are shown in Fig. 3. A three normal mixture model is the best fit for this distribution of conflict event altitudes in ZDV. The other ARTCCs' altitudes are also best modeled using a three normal mixture model.

### B. *Separation Distance*

Minimum separation between two aircraft is typically 5 nmi horizontally and 1,000 ft. (or 2,000 ft. if the aircraft is not

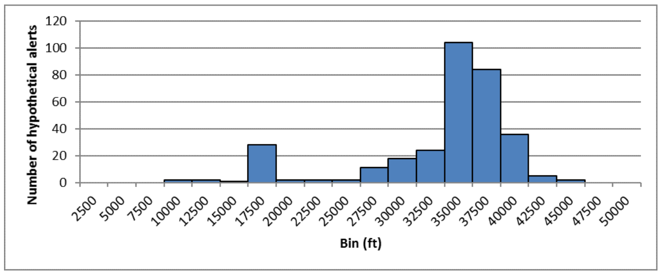

Fig. 3. Total number of hypothetical conflict alerts for each altitude bin in ZDV. The bins are in 2500 ft increments, with the lower bound inclusive, and the outer bound exclusive.

RVSM capable) vertically when in Class A (en route) airspace. When examining the predicted conflict events produced by Trajectory Conflict Probe, the distance between the aircraft is used to categorize the flights into bins. Each bin's lower bound is inclusive, while the upper bound is exclusive.

For horizontal separation, there is a bin for every nautical mile between 0 and 5. Figure 4 presents an example of the binned predicted conflict events for ZDV. In this example, the number of flights in the first four bins is similar, and the number of flights in the 4 to 5 nmi bin is lower.

The vertical separation between the aircraft is binned in 500-foot intervals, from 0 to 2000 ft. As with the horizontal separation bins, the bins for vertical separation have an inclusive lower bound and an exclusive upper bound. Figure 5 displays the results from examining ZDV's vertical separation between aircraft during conflict events. In ZDV, most of the predicted conflict events occurred between 0 and 500 ft. Some airspaces had different distributions, including near-uniform, while others were similar to ZDV.

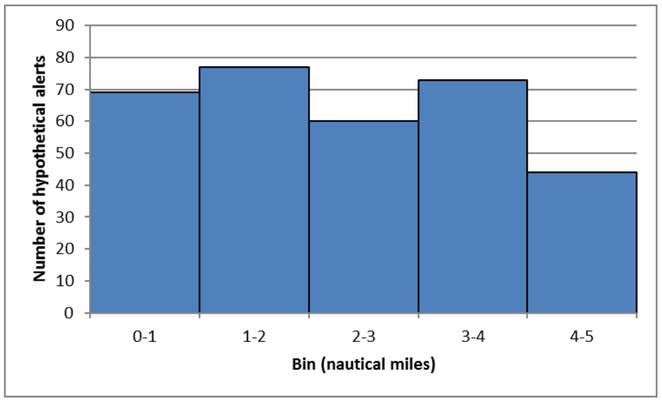

Fig. 4. Total number of hypothetical conflict alerts for each horizontal separation bin in ZDV. The bins are in one nmi increments, with the lower bound inclusive, and the upper bound exclusive.

### C. Encounter Angle

The encounter angle is the difference between the aircraft headings at the closest point of approach, measured in degrees (°). Predicted events are counted and grouped into four bins: 0° to 15°, 15° to 90°, 90° to 165°, and 165° to 180°. The first

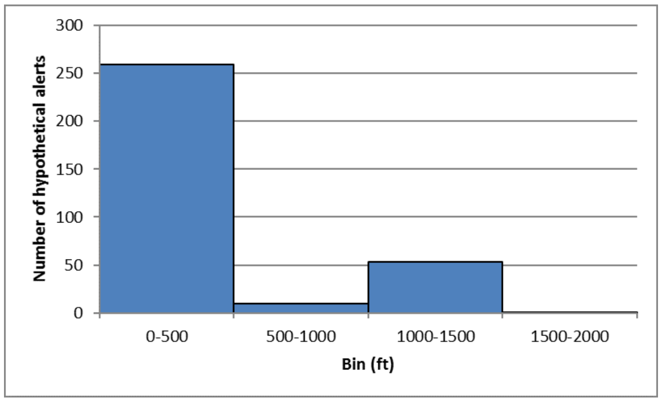

Fig. 5. Total number of hypothetical conflict alerts for each vertical separation bin in ZDV. The bins are in 500 ft increments, with the lower bound inclusive, and the upper bound exclusive.

bin, 0° to 15°, represents a predicted conflict where the two aircraft are either in-trail (following the same horizontal path) or are on similar paths and closing at a very shallow angle. A special case of these events, referred to as overtake, is when the aircraft are on the same path and the trailing aircraft is faster than the leading aircraft. The second and third bins, 15° to 90° and 90° to 165°, represent predicted conflicts that have aircraft crossing paths from the right or left. The last bin from 165° to 180° represents predicted conflict events from aircraft trajectories that are approaching one another either head-on or near head-on.

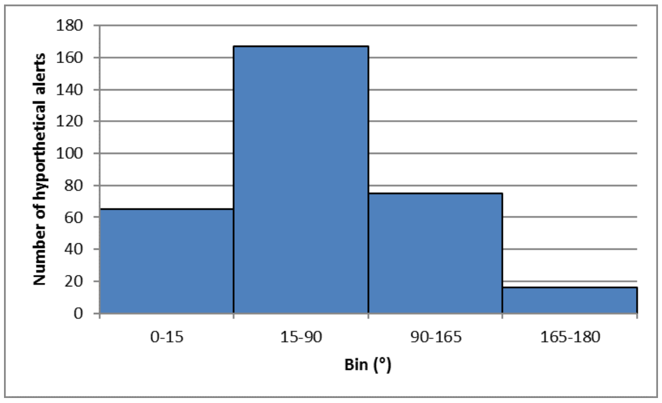

Fig. 6. Total number of hypothetical conflict alerts for each encounter angle bin in ZDV. The bins are separated into 0° to 15° for in-trail conflict, 15° to 90° for crossing angle conflict from one side, 90° to 165° for a crossing conflict on the mirror side, and 165° to 180° for a head on conflict. The lower bound is inclusive, and the upper bound is exclusive.

These different categories of encounter angles are likely to present different challenges to systems that need to predict and resolve them. For example, in-trail encounter angles are particularly sensitive to errors in aircraft speed calculations, while head-on events may have high closure rates requiring quick application of conflict resolutions.

Looking at the results from Trajectory Conflict Probe, the majority of the predicted events in ZDV happened in crossing, while very few head-on conflict events were predicted. Figure

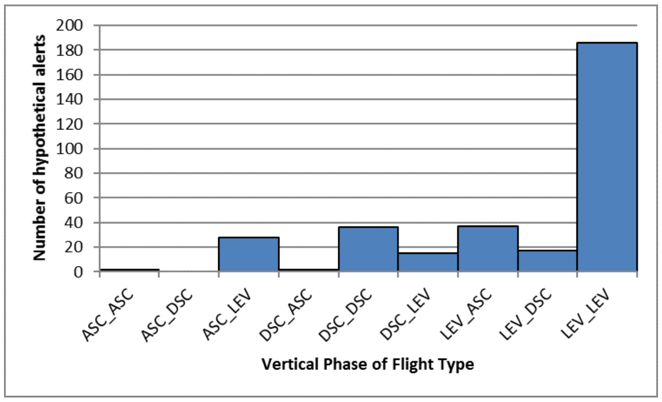

Fig. 7. Total number of vertical phase of flight pairs for aircraft in hypothetical conflict alerts in ZDV. Each bin is the pairwise combination of vertical phase of flight of both the ownship and intruder during a conflict. ASC stands for ascending, DSC stands for descending, and LEV stands for level.

6 illustrates the distribution of encounter angle between aircraft in ZDV's conflict events.

## D. Vertical Phase of Flight

The vertical phase of flight describes whether the aircraft is climbing, descending, or level. An aircraft is ascending (ASC) when it is increasing in altitude and descending (DEC) when it is decreasing in altitude. An aircraft is level (LEV) when the aircraft remains at a constant altitude. This metric captures information about the vertical profiles of the flights during the predicted conflict event.

In this study, the vertical phase of flight for both aircraft is considered, creating a vertical phase of flight pair. There are nine combinations of vertical phase of flight: ASC_ASC, ASC_DSC, ASC_LEV, DSC_ASC, DSC_DSC, DSC_LEV, LEV_ASC, LEV_DSC and LEV_LEV. The first code before the underscore denotes the vertical phase of flight of the ownship, while the code after the underscore denotes the vertical phase of flight of the intruder.

For the results from Trajectory Conflict Probe, each conflict event is placed into a bin that corresponded to the vertical phase of flight of the pair. Figure 7 presents the results of examining this property in ZDV. In ZDV, most of the flights are level during a conflict event, while some occurred while one aircraft was ascending or descending and the other was level.

## E. Aircraft Type

The last property examined was the aircraft type. This parameter is a two- to four-character ICAO aircraft type designator representing the type of aircraft involved in the conflict event. Each aircraft has its own code. For example, Boeing 737-700 aircraft have ICAO code B737, and Airbus A321 aircraft have ICAO code A321. The choice of aircraft type naturally affects the nominal aircraft performance data retrieved from BADA, such as speed profile, climb and descent rates, and weight. This aircraft performance information

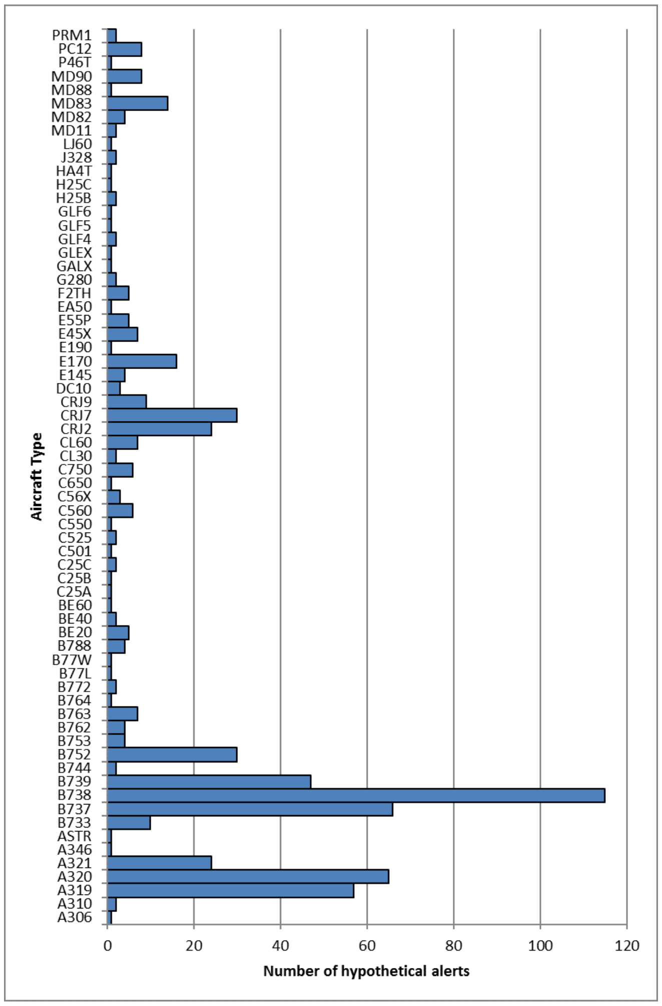

Fig. 8. Total number of aircraft types involved in hypothetical conflict alerts in ZDV. The bins are separated by aircraft type.

allows the conflict generator program to generate customized trajectories for the aircraft involved in a conflict.

The results for the aircraft type parameter are simply binned by each aircraft type, involved in the predicted conflict event, that was generated by the Trajectory Conflict Probe software. Both aircraft involved in the event are counted toward the total for each aircraft type. Figure 8 describes the number of each aircraft type present in the predicted events for ZDV. This ARTCC had 66 different aircraft types involved in conflict events, with at least two aircraft in each bin. The frequencies in this graph mirror the representation of aircraft types found in the airspace.

This section only described the results from one of twenty ARTCCs. The same analysis was applied to every ARTCC for the days worth of data that was collected. The results from all of the ARTCCs are packaged into the Avro file, and used as inputs to the conflict generator algorithm.

## V. Results of Algorithm

The algorithm was tested in two ways: by generating a million conflicts and comparing them to their expected values

TABLE I
MEAN ABSOLUTE ERROR FOR EACH CONFLICT PROPERTY

| Property Name at CPA | Mean Error | St. Dev. |
|---|---|---|
| Horizontal Separation (ft) | -3.72E-13 | 2.77E-9 |
| Vertical Separation (ft) | -3.05E-18 | 1.90E-14 |
| Encounter Angle (°) | -2.41E-13 | 1.97E-10 |
| Ownship Latitude (°) | -3.55E-17 | 4.83E-15 |
| Ownship Longitude (°) | -1.82E-15 | 1.06E-14 |
| Ownship Altitude (°) | 0 | 0 |

at the closest point of approach (CPA) and visual inspection of a small subset of the conflicts. This section describes the results of both of these methods.

## *A. Simulations*

The first test performed to validate the algorithm was to compare the parameters of the generated trajectories to the user-specified CPA properties: horizontal separation distance, vertical separation distance, encounter angle, ownship latitude, ownship longitude, and ownship altitude at CPA.

We considered the mean absolute error between the user-input conflict properties (ground truth) and the properties of the generated conflicting trajectories. The mean errors, averaged over 100,000 trajectories, are displayed in Table I for each conflict property. Observe that these errors are of the order of the numerical and rounding errors of the Java virtual machine (JVM). This result is expected because the algorithm adopts the user-specified conflict properties in an exact mathematical formulation of the problem.

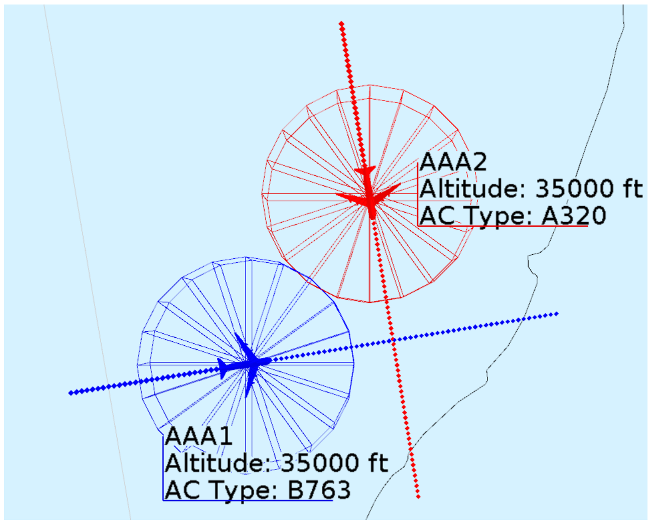

Fig. 9. Overhead view of the crossing conflict with both aircraft flying level. The legal En Route separation distance is shown as circles around the aircraft, and the small spheres represent a generated track point. Note: the aircraft models are exaggerated in scale.

## *B. Visual Inspection*

We used a visualization tool to visually inspect the generated conflicting trajectories for different conflict events.

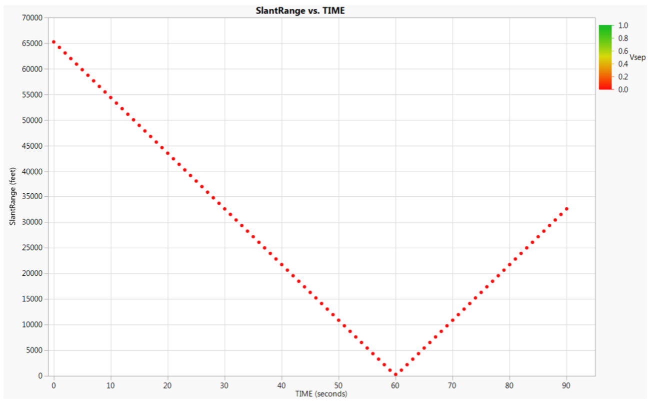

Fig. 10. Distance between the aircraft over time. The $x$-axis represents time in seconds, and the $y$-axis is the distance between the two aircraft in feet. The color of the dots represents the vertical separation between the aircraft, with red being the smallest distance and green being the furthest.

For instance, Table II shows three different conflict events: a crossing conflict with both flights level, a shallow angle conflict with one of the flights ascending and the other level, and a crossing conflict with one flight ascending and the other level.

The first conflict is a collision event set to occur with both aircraft approaching each other at a 90° angle. Figure 9 shows an overhead view of the two aircraft flying their generated trajectories for this conflict. The circles around the aircraft are 2.5 nmi in diameter and are 1000 ft tall. These circles represent the legal separation distance of two aircraft in En Route airspace. If the circles overlap, it means that the two aircraft have lost legal separation, i.e. conflict. This visualization shows that the aircraft conflict is as expected at 90° angle with a collision event. Figure 10 shows the separation distance between the aircraft over time. This chart confirms that the closest point of approach occurred at the expected time of 60 seconds.

The second conflict spaced each other with a minimum distance of 2 nmi which resulted in a non-collision conflict event. In this event, the ownship was ascending and approaching the intruder at a shallow angle ($0° - 15°$). Figure 11 displays the overhead view of the conflict event, while Fig. 12 shows a side view. The trajectories generated matched what was expected, with one flight ascending and the other level. Figure 15 illustrates the distance between both aircraft, proving that the closest point of approach occurred around the expected time of 60 seconds.

The third conflict was a combination of the first and second events, with the aircraft trajectories forming a collision conflict at a crossing angle of 90°, but with one aircraft ascending and the other level. Figure 13 and Fig. 14 illustrate the overhead and side views of the conflict, respectively. It can be seen that the aircraft are following the expected trajectories based on the given conflict properties. The distance between the aircraft over time is shown in 16, which proves that the aircraft reach minimum separation at the expected time of 60 seconds.

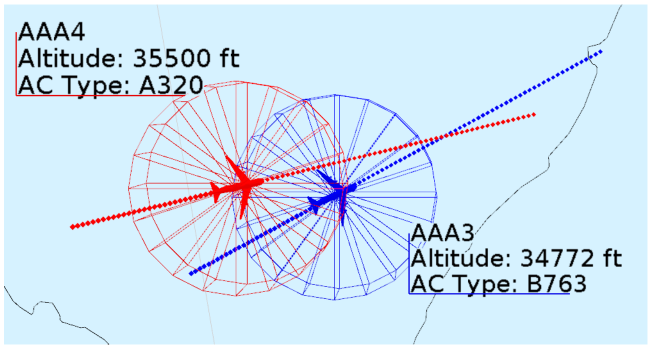

Fig. 11. Overhead view of the shallow angle conflict with one aircraft flying level and the other ascending. The legal En Route separation distance is shown as circles around the aircraft, and the small spheres represent a generated track point. Note: the aircraft models are exaggerated in scale.

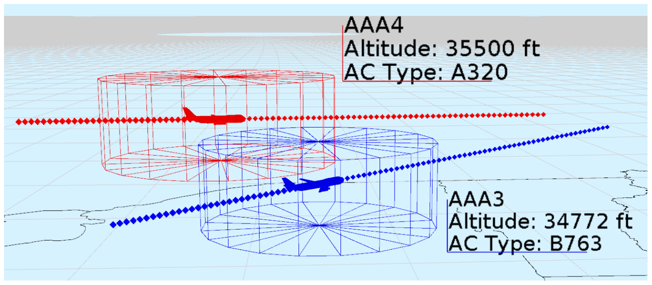

Fig. 12. Side view of the shallow angle conflict with one aircraft flying level and the other ascending. The legal En Route separation distance is shown as circles around the aircraft, and the small spheres represent a generated track point. The Aircraft denoted at AA3 is ascending, while the other is level. Note: the aircraft models are exaggerated in scale.

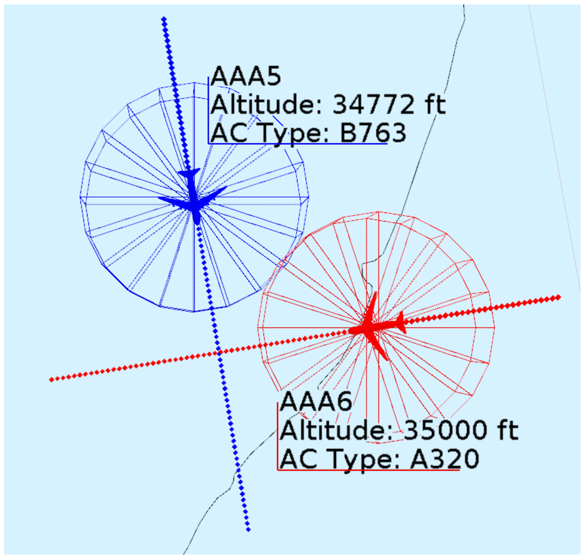

Fig. 13. Overhead view of the crossing conflict with one aircraft flying level and the other ascending. The legal En Route separation distance is shown as circles around the aircraft, and the small spheres represent a generated track point. Note: the aircraft models are exaggerated in scale.

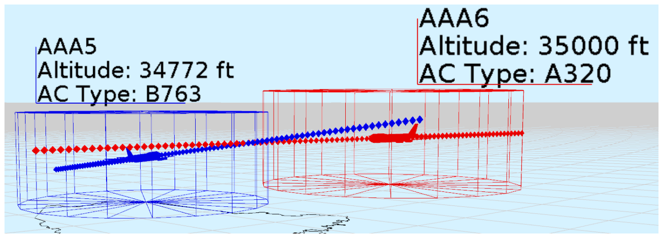

Fig. 14. Side view of the crossing conflict with one aircraft flying level and the other ascending. The legal En Route separation distance is shown as circles around the aircraft, and the small spheres represent a generated track point. The Aircraft denoted at AA5 is ascending, while the other is level. Note: the aircraft models are exaggerated in scale.

## VI. Conclusion

For unmanned aircraft systems (UAS) to be integrated into the National Airspace System (NAS), a detect and avoid (DAA) system used to assist the ground pilot in preventing loss of separation between other aircraft. To accurately train and validate DAA systems for UAS, realistic encounter events must be used. These types of conflicting events do not exist in the NAS since air traffic controllers prevent them from happening. To use conflict events to train DAA systems, one must first generate these conflict events. The FAA has partnered with Rowan University to develop a conflict generator program, which uses realistic flight properties to generate conflicts.

This paper describes an algorithm developed to generate these realistic flight conflicts using a spherical coordinate system. The generator is formulated mathematically as a constrained optimization problem, the constraints being the user-specified conflict properties at the Closest Point of Approach (CPA). The algorithm outputs a pair of conflicting 4D trajectories that satisfy the given properties.

The algorithm was implemented in a Java program. The program can generate the aircraft pair encounters using the algorithm, based on user defined parameters for the encounters. The encounters can be defined manually by the user or the user can input probability distributions for each encounter property, with the desired number of flights. This method allows the random generation of the encounters that fit the given distributions.

TABLE II
Conflict Properties for each Specified Conflict Event in FliteViz4D

| Ownship VPoF | Intruder VPoF | Horizontal Sep (nmi) | Vertical Sep (ft) | Encounter Angle (°) | Altitude of Ownship at CPA (ft) |
|---|---|---|---|---|---|
| Level | Level | 0.05 | 0 | 90 | 35000 |
| Ascending | Level | 2 | 500 | 15 | 35000 |
| Ascending | Level | 0.05 | 0 | 90 | 35000 |

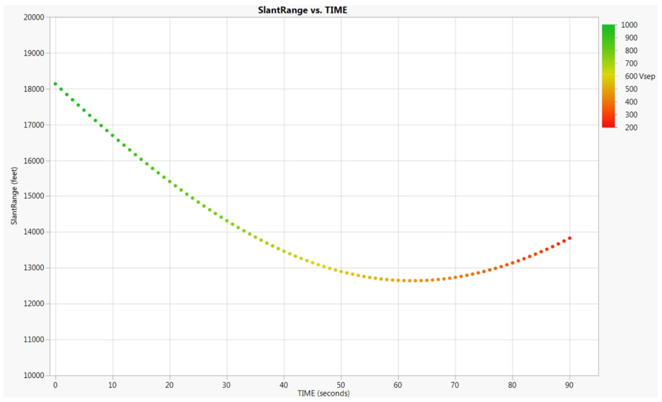

Fig. 15. Distance between the aircraft over time. The $x$-axis denotes time in seconds, while the $y$-axis denotes the distance between the two aircraft in feet. The color of the dots represents the vertical separation between the aircraft, with red being the smallest distance and green being the furthest.

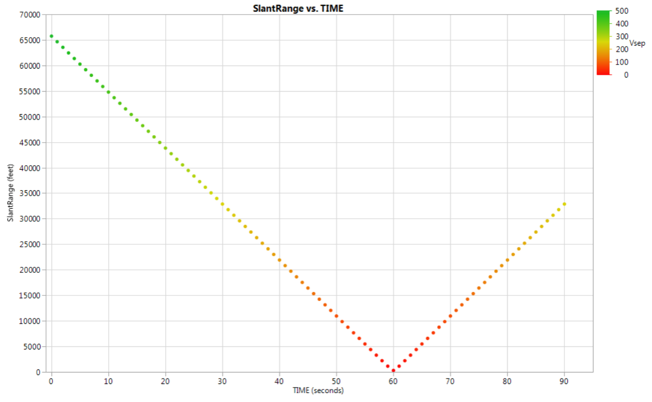

Fig. 16. Distance between the aircraft over time. The $x$-axis denotes time in seconds, while the $y$-axis denotes the distance between the two aircraft in feet. The color of the dots represents the vertical separation between the aircraft, with red being the smallest distance and green being the furthest.

We conducted Monte Carlo simulations to test the algorithm's accuracy. As expected from an exact mathematical derivation, the mean absolute error between the properties of the generated trajectories at CPA is found to be within the bounds of machine error. Also, visual inspection using our visualization tool confirms that the trajectories generated match what is expected based on the given input.

In the future, the algorithm can be upgraded to include changing horizontal position of the two aircraft to simulate a change in the horizontal phase of flight of the aircraft. This will simulate the aircraft turning during the duration of the conflict, which is a more likely scenario when the aircraft are UAS. For the conflict properties study, more recorded traffic will be examined to determine the empirical distributions for the NAS conflict properties throughout the year. Also, conflicts within terminal airspace could be studied and used as input to the algorithm. This will allow more testing for UAS in these areas with more realistic trajectories.

## Acknowledgment

The authors would like to thank Mr. Deepak Chaunhan and Mr. Michael McNeil with the FAA's Engineering Development Services Division, Surveillance Branch, for their financial and technical support.

The authors' thanks also extend to Mr. Evan Gallo, Junior engineering student at Rowan University, for the initial implementation in Matlab.

## References

[1] R. D. Oaks and M. Paglione, "Generation of realistic air traffic scenarios based on recorded field data," in *Air Traffic Control Association (ATCA) 46th Annual Conference Proceedings*, vol. 46, Washington, DC, October 2001, p. 142.

[2] M. Paglione, R. D. Oaks, and J. S. Summerill, "Time shifting air traffic data for quantitative evalution of a conflict probe," in *American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference*, Austin, TX, August 2003, p. 5343.

[3] R. D. Oaks, "A study on the feasibility of using a genetic algorithm to generate realistic air traffic scenarios based on recorded field data," in *American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference*, Monterey, CA, August 2002, p. 4767.

[4] ——, "Generation of realistic air traffic scenarios using a genetic algorithm," in *21st Digital Avionics Systems Conference (DASC)*, vol. 1, Irvine, CA, October 2002, pp. 2A1–1–2A1–2.

[5] J. A. Ritchie III, A. J. Fabian, C. M. Young, and M. Paglione, "Design and performance of an improved genetic algorithm implementation for time-shifted air traffic scenario generation," Federal Aviation Administration, Atlantic City, NJ, Tech. Rep. DOT/FAA/TC-TN16/3, January 2016.

[6] M. J. Kochenderfer, L. J. P. Espindle, J. K. Kuchar, and J. D. Griffith, "Correlated encounter model for cooperative aircraft in the national airspace system, version 1.0," MIT Lincoln Laboratory, Tech. Rep. ATC-344, October 2008.

[7] ——, "Uncorrelated encounter model of the national airspace system, version 1.0," MIT Lincoln Laboratory, Tech. Rep. ATC-345, November 2008.

[8] S. M. Malaek, A. Alaeddini, and D. S. Green, "Optimal maneuvers for aircraft conflict resolution based on efficient genetic webs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 47, no. 4, pp. 2457–2472, 2011.

[9] C. MingQiang, "Flight conflict detection and resolution based on neural network," in *2011 International Conference on Computational and Information Sciences*, Chengdu, China, October 2011, pp. 860–862.

[10] G. Meng and F. Qi, "Flight conflict resolution for civil aviation based on ant colony optimization," in *Fifth International Symposium on Computational Intelligence and Design*, Hangzhou, China, January 2012.

[11] W. Liu and I. Hwang, "Probabilistic aircraft midair conflict resolution using stochastic optimal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 37–46, 2014.

[12] Y. Yang, J. Zhang, K.-Q. Cai, and M. Prandini, "Multi-aircraft conflict detection and resolution based on probabilistic reach sets," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 309–316, 2017.

[13] Eurocontrol Experimental Centre. User manual for the base of aircraft data (BADA), revision 3.13. [Online]. Available: http://www.eurocontrol.int/services/bada

[14] M. Paglione, C. Santiago, A. Crowell, and R. D. Oaks, "Analysis of the aircraft to aircraft conflict properties in the national airspace system," in *2008 American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, August 2008.

[15] Federal Aviation Administration. The Operations Network (OPSNET). [Online]. Available: https://aspm.faa.gov/opsnet/sys/main.asp