A Simulated Environment for Testing 4D Detect See and Avoid Scenarios for UAVs

Jonathan D. Stevenson
Memorial University of Newfoundland
St. John's, NL, Canada

Oihane Cereceda
Memorial University of Newfoundland
St. John's, NL, Canada

***Abstract*-A synthetic environment has been developed which permits the realistic simulation of encounter scenarios of two small aircraft. This may be used to develop and test Detect Sense and Avoid (DSA) strategies for Unmanned Aerial Vehicles (UAVs). A novel maneuver has been invented that simplifies the generation of air-to-air encounters. This "Phi Maneuver" guarantees a variety of encounter geometries, and at least one encounter for each pass of the Intruder aircraft through the loiter zone of the Target aircraft. This provides an effect way to test multiple DSA scenarios and generate statistics on the relative merits of a particular DSA method.**

Key Words: UAV, 4D Synchronization, Detect See and Avoid, Simulation, FlightGear Multiplayer

I.INTRODUCTION

The current simulation environment is the result of over 8 years of development within Project RAVEN. The simulation began as a way to model typical ship inspection scenarios by small UAVs in the Newfoundland offshore environment (2005-2006). It uses the AeroSIM v2.0 toolkit for MATLAB/Simulink, as developed by Unmanned Dynamics Inc. [1]. The toolkit includes interface module which may be used to drive an external flight simulator such as FlightGear (FG), a free open-source flight simulator. In the initial simulation, the MATLAB and FlightGear were hosted on separate computers which communicated via Transmission Control Protocol/ User Datagram Protocol (TCP/UDP), to permit real-time simulations.

The simulation was upgraded and used to assist in the development of Automated Takeoff and Landing (ATOL) control algorithms from 2006-2008 for the Aerosonde Mk4.2 and the GiantStik at the Clarenville Airfield (CCZ3). This simulation included a model of the Piccolo II Autopilot and also featured manual override controls using either a joystick or radio control (R/C) controller. Custom ATOL control algorithms were added to the basic 2D and 3D waypoint following methods used on the Piccolo II. This representation of the autopilot and manual controls are still present in the current 4D Simulation. From 2008-2010 the simulation was used to develop the initial autopilot gains to use on the GiantStik and Senior Telemaster R/C aircraft.

Finally, the ATOL simulations were revived and used as the basis for the development of the current simulation (2012-present day). The Multi-player feature in FlightGear was used to support the simulation of multiple aircraft in the same airspace. This final form of the simulation is the basis of the 4D Simulation Environment presented here.

II. SIMULATION STRUCTURE

The basic structure of the AeroSIM-based simulation may be seen in Fig. 1. The major components as indicated are:

- **AUTOPILOT (AP)**, a representation of the autopilot on the UAV.
- **Aircraft SIM**: A detailed Aerodynamic simulation of a small airframe in MATLAB/Simulink built using the AeroSIM toolkit [1].
- **Inputs from FG**: AeroSIM interface module which allows the interception of FG output information in the form of User Data File (UDF) packets.
- **Outputs to Flight Dynamics Model (FDM)**: AeroSIM interface module which outputs the current aircraft states to a UDP port, to drive the flight data model in FG.
- **Input from Target SIM:** Another AeroSIM interface module which obtains position information from another aircraft FDM in the FlightGear multi-player environment. This may be used to mimic the function of an Automatic dependent surveillance-broadcast (ADS-B) avionics box (i.e. getting the GPS coordinates of another detected aircraft at 4-5 Hz rate).

III. MATHEMATICAL BASIS OF SIMULATION

The AeroSIM aircraft block is an implementation of a 6 Degrees of Freedom (6-DOF) FDM. Consider the typical diagram of an aircraft such as shown in Fig. 2 which shows the three principal position axis: Longitudinal, Lateral and Vertical (X,Y,Z), and the three corresponding angles around each axis, namely the Roll, Pitch and Yaw (φ, θ, ψ).

The FDM of this aircraft can be represented as a system of first order non-linear differential equations. The simulation results are calculated by solving these for all the state variables, based on body coordinates, with respect to time. Response in other coordinate systems of interest can be obtained by transformation matrices. The 6-DOF flight model consists of six fundamental state variables: u v w (body velocities along the x, y and z axes), calculated from the total forces along each axes, and p q r (body angular velocities about the x, y and z axes), calculated from the moments of inertia about its center of gravity and the aircraft's mass [2].

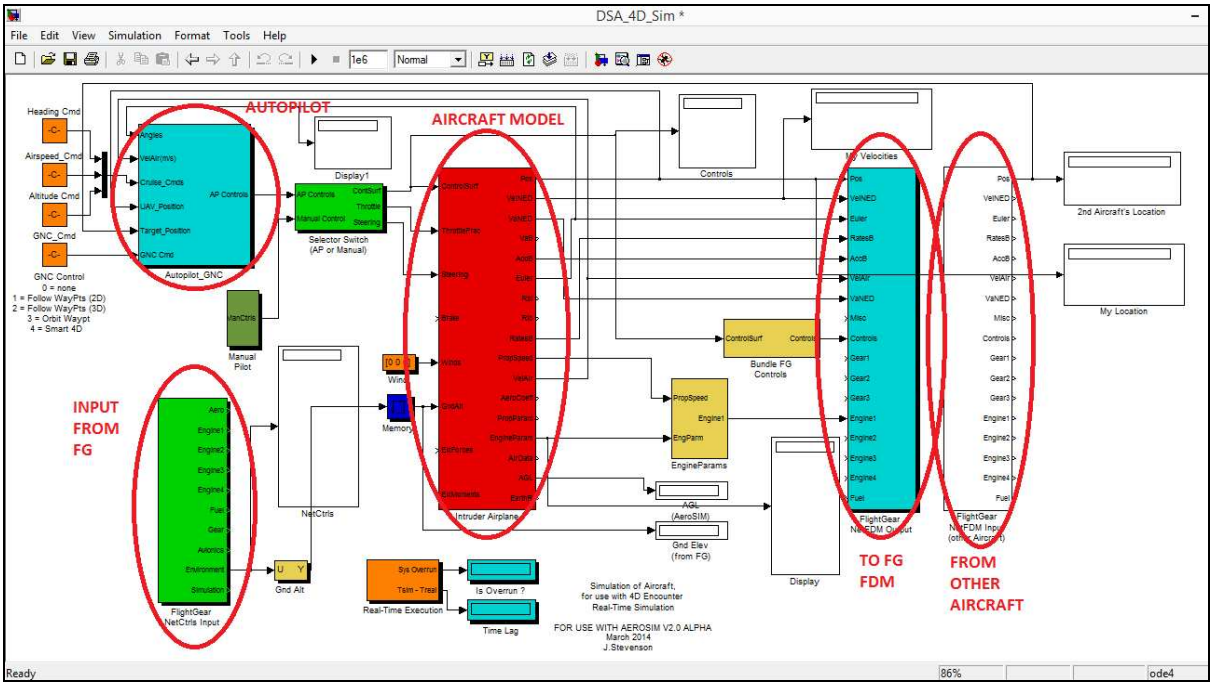

Fig. 1: Top level of 4D DSA simulation

The AeroSIM aircraft block provides a numerical solution method (in Simulink) which uses the principles described above to perform integrations in time. The change of the body angles [4] is calculated using the velocities, as well as the aircraft position in terms of the fixed-earth. Once the body velocities are known, the angle of attack α, and side-slip angle β, both which are critical to determining the aerodynamic coefficients, are determined.

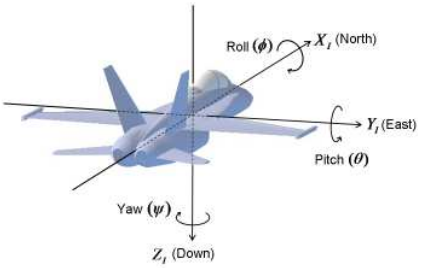

Fig. 2: 6-DOF diagram of hypothetical aircraft [3]

The AeroSIM does these calculations for each time step, marching forward in time. The size of this time step is critical, as it needs to be fine enough for numerical stability. For the types of small UAVs being simulated in this project, a base time step of 20 msec appears to be a reasonable compromise.

IV. 4D ENCOUNTER SIMULATIONS USING MULTIPLAYER

The multiplayer feature built-in to FlightGear [5] uses IP ports to output the local aircraft FDM data to a remote server. A corresponding input port is used to accept FDM data for any other aircraft entities in the local area of the user's aircraft. Several international server sites have been created to support the FG Multiplayer mode [5]. Given the open source nature of FG, it was quickly determined that any computer could be used as a server, requiring only that the IP address be known by the other computer(s). An experiment was conducted where the server address for computer 1 was defined as that of a second nearby computer 2. Likewise, the server address for computer 2 was set to that of computer 1. Once the right port numbers were assigned the experiment was successful and the two aircraft could see each other.

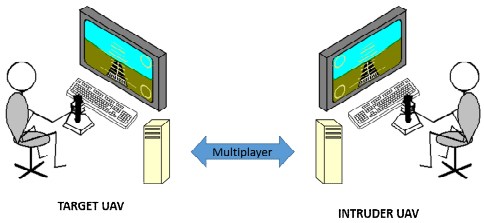

Fig. 3: Multiplayer-based 4D simulation environment

Each computer was then setup to host one complete instance of the AeroSIM simulation, driving the local instance of FG as the visualization tool. With the multiplayer settings also active as previously described, this created a dual-computer 4D simulation environment as shown in Fig. 3.

The UDP port numbers and IP addresses for each computer become the corresponding remote settings for the other computer.

The FG multiplayer mode appears to provide a stable link, and works across any two computers theoretically anywhere on

the internet. The method also works for laptops using a wireless connection. As long as reasonably powerful computers are used the simulations run in real-time. Also, while the 4D Simulation environment uses 2 aircraft it should be possible to extend this method to host many (n) aircraft on multiple computers, either additional MATLAB-controlled or normal human-piloted FlightGear aircraft.

V. DEVELOPING 4D ENCOUNTER GEOMETRIES

The 4D Simulation Environment has been used to develop and test coordinated aircraft maneuvers, developing flight-plans and test procedures which may be practiced and perfected before attempting them using live field testing. Ironically, considering the great concern about UAVs and mid-air collisions, deliberately setting two UAVs such that they threaten to meet each other in the sky turns out not to be as easy as it would seem. Several promising methods have been developed to support DSA testing.

A. Opposing circuits

In this method, one aircraft is flown in a typical circuit, using a basic 4-waypoint autopilot flight-plan as shown in Fig. 4 in BLUE (i.e. ABCD). A second aircraft is now added, flying a similar circuit but in the opposite direction (i.e. BADC) as shown in RED. Note that an alternative method could be to define the second circuit as illustrated in GREEN. However, this GREEN flight-plan would involve one of the aircraft flying behind the Ground Control Station (GCS) every circuit. A difference in altitude is also used a safety measure since the idea is to setup an encounter situation but not a collision.

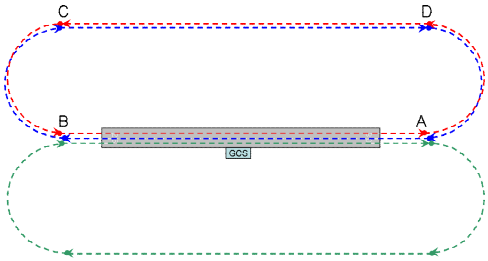

Fig. 4: 4D opposing circuits (ideal case)

Even with this simple geometry, getting the two aircraft to encounter each other at a desired location still requires careful timing and coordination by the manual pilots and the Aerial Vehicle Operator (AVO) at the GCS. Typically, one aircraft at a time is launched and flown manually, establishing a flight pattern close to the planned waypoint pattern, and it is then switched to AP mode. The same flight-plan it defined as used for the first aircraft, but in reverse order. If timed correctly, the two Aircraft will encounter each other in a head-on situation, about half-way down the runway.

With real UAV flights, winds are shown to cause the flight plans to shift from the ideal situation implied in Fig. 4. Even if assuming the ideal case of the wind direction coming straight down the runway (i.e. no cross-wind) one of the aircraft will encounter a head-wind, while for the other aircraft this will be a tail-wind. The AP will of course try to correct the effects, but not completely before the encounter occurs. In the more typical cross-wind situation, the heading corrections for the two aircraft will also not be the same, to the point where the flight-plans for each will become distorted.

The “ballooning” of one of the aircraft and the altitude dropping of the other aircraft may also set up a situation where altitude separation, defined as a safety measure, could become compromised.

B. Time-Synchronization Methods

Attempts have also been made to develop 4D algorithms which adjust for a common time of arrive of two aircraft at a defined 3D point in space [6]. This technique uses custom software which monitors the location of two UAVs, each controlled by its own GCS in real time. The software communicates with each of the two GCSs, analyzes the telemetry data from the two UAVs, and calculates adjustments needed to the flight plans with the aim to establish 4D time synchronization.

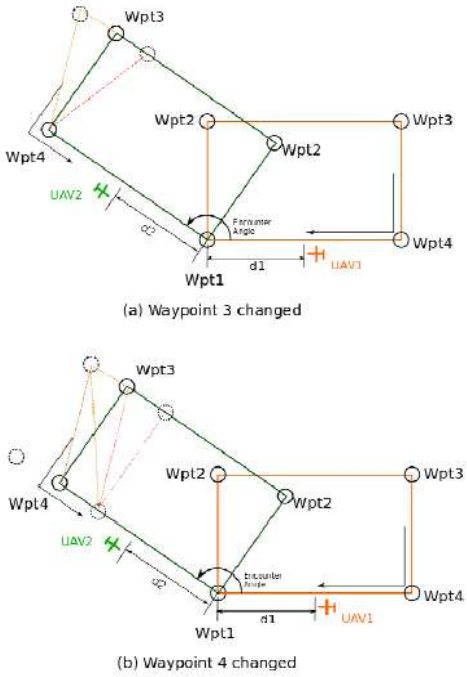

Fig. 5: Waypoint adjustments to synchronize time of arrival [6]

As shown in Fig. 5, each UAV is programmed to fly a rectangular flight plan with four waypoints. In this example, the circuits of the UAVs are in opposite sense to each other (i.e. UAV1, orange plan, clockwise, UAV2, green plan, counter-clockwise) and share a common waypoint, with a small offset in altitude to prevent collision, (Wpt1) as shown.

The goal is to minimize the distance error (d1-d2) adjusting the location of either Wpt3 or Wpt4 (UAV2 path plan) as the two aircraft are both approaching Wpt1. When this distance error is tuned to zero, 4D synchronization is achieved. The success of the algorithm, and the speed at which is converges to

a 4D synchronization solution depends strongly on the flight conditions.

VI. INTRODUCTION TO PHI MANEUVRE

During simulated testing of opposing circuits within the 4D Simulation environment, a new method was discovered. While setting up opposing circuits, one of the aircraft was forced to intercept a waypoint that was inside its minimum turn radius. Most autopilots like the Piccolo II feature an escape algorithm to prevent this situation, by forcing the UAV to fly to the next waypoint if it is impossible for it to reach the current one. However, the representation used in the 4D Simulation does not have this algorithm. Instead, the aircraft continued to turn in an attempt to intercept the waypoint. The aircraft, in this case a GiantStik, began to orbit the waypoint at its minimum turn radius, which at the commanded airspeed (20 m/s) was approximately 150m. Meanwhile, the second aircraft continued to fly its normal circuit. As it passed through the orbit of the first aircraft, there were one and sometimes two interception opportunities as the aircraft passed by. Note that this happened without any special timing or synchronization, during every pass of the second aircraft through the first aircraft's orbit.

The flight-plan of the second aircraft was changed from a circuit into a "dog-bone" pattern. The second aircraft now would make repeated passes down the runway in both directions, with tight 180 degree turns at each end of the runway. This significantly increased the number of encounters (from once every 90-120 sec, to once every 45 seconds). The 4D maneuver geometry created is as shown in Fig. 6.

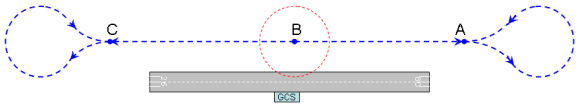

Fig. 6: The phi maneuver

The only critical waypoint is point B, which is simply the orbit center of the first aircraft shown in RED. The second aircraft flies the BLUE dog-bone flight-plan. Waypoints A and C are used to line up the second aircraft such that it will fly straight down the runway and through point B. Thus, the central theme of this maneuver is simply a circle with a line through it – resembling the Greek letter Φ (PHI), and whence the name "PHI Maneuver" has been created. The simplicity of this maneuver is that only one waypoint needs to be defined accurately. As shown this should be shifted away from the GCS so that the encounter area is always in front. Most commercial autopilots have a waypoint-orbit capability built-in so the RED flight-plan is easy to define. The radius should be chosen such that it is a wider than the minimum turn radius of the aircraft. Otherwise, the orbital radius will effectively become this minimum radius. The AP will struggle to maintain the aircraft in a stable orbit that is on the edge of what the vehicle can do aerodynamically. By leaving some margin for an inward turn, the maneuver options for the orbiting aircraft are also kept open.

The other aircraft only has to fly through waypoint B to guarantee at least one 4D encounter for every pass. The random nature of the 4D encounters created by this maneuver geometry will be such that the exact locations of each aircraft will not be known ahead of time. The range of each encounter will also be random. This assumes no enhancements are used to improve the chances of an interception.

VII. INITIAL RESULTS

The 4D Simulation has been used to experiment with the concept of giving one of the aircraft active interception guidance control, similar to that of an air-to-air missile, to force a more deliberate interception. The orbiting aircraft was given a simple interception guidance program, which remains dormant while the second aircraft is far away. Once the second aircraft comes within range of the orbiting aircraft, it goes from being "passive" to "active", and attempts to intercept the approaching aircraft. Simple Pursuit (SP) was used initially, where the heading command has set equal to the instantaneous bearing to target. However, as expected this almost always resulted in the pursuing aircraft flying behind and chasing the second aircraft, especially if the aircraft was being forced to turn inwards and thus exceed its maximum turn rate. The difference in airspeeds was usually not enough to give the interceptor any advantage once it was chasing the target, so the encounter never happened. The target aircraft simply flew away, beyond the active pursuit range, and the interception attempt was stopped. The orbiting aircraft then returned to its previous (orbit) program to wait for the next opportunity.

A. Switch to Proportional Navigation

A very basic Proportional Navigation (PN) guidance scheme was added to the simulation, replacing the Simple Pursuit (SP) method described above. PN is the standard Guidance Navigation Control (GNC) method used by practically all modern guided missiles, replacing the earlier Simple Pursuit (SP) method since the early 1960s due to its limitations [7]. PN uses the rate of change of the target bearing angle as the primary driver for the missile course correction [8].

As illustrated in Fig. 7, at the moment represented the missile and target are both moving at different velocities and headings. Relative to the missile, the target will be located at a Range (R), and at a Line-of-Sight (LOS) angle (σ). Due to the relative motions of both objects, this angle will likely be changing ($\dot{\sigma}$) as shown.

The PN guidance law seeks to control null-out the LOS rate until it becomes zero (i.e. $\dot{\sigma} = 0$). It may be shown that the missile should adjust its heading (ψ) to counter this LOS angle change according to the law:

$$\psi = K * \dot{\sigma} \quad (1)$$

Where,

K = Proportional Navigation Gain

$\dot{\sigma}$ = rate of change of the LOS angle

$\dot{\psi}$ = rate of change of missile heading

If the LOS angle rate is zero, and range is decreasing, the missile will eventually intercept the target. The choice of K depends on how aggressive a maneuver is required or desired, and the turn-rate capabilities of the missile. A value of 3-5 is typical, but a value as high 10 may be needed to accommodate head-on engagements [9].

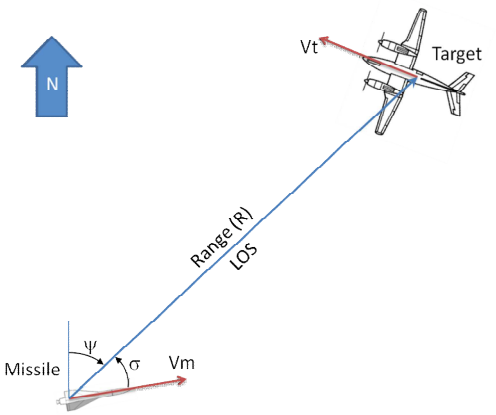

Fig. 7: Proportional navigation geometry

When PN was applied to the PHI Maneuver, the orbiting aircraft was able to intercept the approaching target aircraft with fair regularity, once the PN gain and maximum bank angle of the interceptor were properly adjusted. In several cases, a side-ways encounter (angle ~90 deg) was quickly followed by a second pursuit-style (180 deg) tail-on encounter. The chance of a successful encounter appeared to be associated with where the orbiting aircraft was in its orbit when the approaching aircraft was detected. The best situations were when the interceptor was just turning past the 12 or 6 o'clock positions, with the aircraft approaching in front of the interceptor.

An alternative strategy was to provide the aircraft flying the "dog-bone" pattern with the interception GNC program, while the target assumes the orbit pattern. This resulted in even more dramatic interceptions by the approaching aggressor aircraft.

B. *Anti-Proportional Navigation as an Avoidance Method*

The initial purpose was to give the Intruder aircraft intelligent interception behaviour during 4D Simulations of the PHI maneuver. While setting up the PN scheme, a sign error was made in the LOS angle correction which had the effect of creating a course correction AWAY from the situation of $\dot{\sigma} = 0$. This had the effect of causing the Intruder to engage in an evasive maneuver. Since the desired behaviour was to intercept, the sign error was fixed, but this hinted at a very promising avoidance method.

The erroneous PN algorithm initially programmed was in fact one form of the Anti-Proportional Navigation (APN) scheme as investigated by others [10] [11]. It is noted by several authors that even a modest APN gain of 3 appears to be sufficient to create an effective avoidance maneuvers for typical small UAVs, provided the other aircraft is detected at sufficient range [12]. Our own initial results indicate that for the aircraft used in our simulations, with cruise speeds of 20 to 30 m/s, a detection range of 300 to 500m appears to be the minimum needed to guarantee a 500' (165m) miss distance. The simplest way to implement the APN is to use exactly the same guidance law as equation 11 but with a negative value for the gain (K).

VIII. Future Research Plans

Some of the more interesting discoveries during this research will be the subject of continued research, including:

1. Determining the effectiveness of the PHI Maneuver as a means to generate a statistically significant variety of 4D encounters.
2. Live field testing of the PHI Maneuver using two UAVs.
3. Investigating the effectiveness of APN as an avoidance algorithm, especially against the worst case "kamikaze" type pursuing aircraft.
4. Implementing the APN avoidance algorithm into an ADS-B based prototype DSA system and field testing this using live dual aircraft flights.

Acknowledgments

The authors wish to thank the support of Dr. Siu O'Young and Dr. Luc Rolland, plus the entire RAVEN Project team at Memorial University, for their support in the research outlined in this paper.

References

[1] Unmanned Dynamics Inc., *"AeroSIM Aeronautical Simulation Blockset"*, Version 1.2, Hood River, OR, 2006. Version 2.0, 2007.

[2] Cook, M. V., *"Flight Dynamics Principles"*, Elsevier Publishing Ltd., Amsterdam, 2007.

[3] CH Robotics, *"Understanding Euler Angles"*, CH Robotics LLC, 2013. Available: http://www.chrobotics.com/library/understanding-euler-angles.

[4] Mathworks, *"Aerospace Blockset 2 User's Guide"*, The MathWorks, Inc., Natick, MA, 2007.

[5] Michael Basler and Martin Spott, "FlightGear Flight Simulator – *Installation and Getting Started"*, Version 0.8 for FlightGear version 0.9.10, April 5, 2006.

[6] Fang, S. X., *"UAV 4D Synchronization"*, St. John's: Project RAVEN, 2014.

[7] Abramovitz M., *"Theoretical Investigation of the Performance of Proportional Navigation Guidance Systems"*, 1953, Ames Aeronautical laboratory.

[8] Zarchan P, *"Tactical and Strategic Missile Guidance"*, 2nd Edition, American Institute of Aeronautics and Astronautics, 1994.

[9] Stallard D. V., *"Classical and Modern Guidance of Homing Interceptor Missiles"*, Raytheon Company, April 1968.

[10] Ham, S. and Bang, H., *"Proportional Navigation-Based Optimal Collision Avoidance for UAVs"*, 2nd International Conference on Autonomous Robots and Agents, December 13-15, 2004.

[11] Kumar, B. A. Ghose, D., *"A Proportional Navigation Based Collision Avoidance Guidance Strategy for Low-Altitude Flight"*, Proceedings of the 3rd Asian Control Conference, Shanghai, China, Jul. 2000.

[12] George, J. and Ghose, D., *"A Reactive Inverse PN Algorithm for Collision Avoidance among Multiple Unmanned Aerial Vehicles"*, 2009 American Control Conference, June 10-12, 2009.