2016 International Conference on
Unmanned Aircraft Systems (ICUAS)
June 7-10, 2016. Arlington, VA USA

Studying the Interaction of UAS and Human Pilots Using the X-Plane Flight Simulator

Craig Lombardo, Ian Miller, and Jon Wallace
Lafayette College, Easton, PA, USA
e-mail: lombardc@lafayette.edu, millerid@lafayette.edu, wall@ieee.org

***Abstract*—A system of networked computer components is described allowing realistic simulation and performance assessment of see-and-avoid algorithms for unmanned aircraft systems (UAS). Due to its realistic physics engine and relatively open architecture, the X-Plane flight simulator is adopted to simulate computer-generated traffic or an aircraft controlled by a human pilot. Using the UDP interface in X-Plane, real-time collision avoidance algorithms are then implemented, where the simulated unmanned aerial vehicle (UAV) controller reads aircraft positions from X-Plane and updates control inputs to the UAV using sequential quadratic programming (SQP) to avoid collisions in real time. The basic concept is illustrated through an example flight involving a single aircraft in an airport pattern and a UAV flying through the same airspace. A second experiment is performed where a human pilot intentionally flies across the path of the UAV.**

I. Introduction

There is increasing interest in unmanned aircraft systems (UAS), not only for military uses but also for scientific and commercial applications. A clear concern of the FAA is the safety impact of UAS traffic on conventional air traffic as well the safety of people on the ground. The FAA has reported many near collisions involving passenger aircraft and drones in recent years [1]. UAS platforms that can see other traffic and automatically avoid collisions would not only be a great benefit to aviation safety, but also may lead the way to less restrictive rules for UAS operation.

One method for assessing the effectiveness of collision avoidance algorithms is through detailed computer simulation, where in addition to the UAS control algorithm, factors such as aircraft physics, weather, and pilot reaction should be captured. Developing such an environment from scratch can be difficult and time consuming. Another approach is to develop experimental UAS test beds to test collision avoidance algorithms in real environments. However, the cost and safety of this approach may be prohibitive.

In this paper, we adopt the popular approach of leveraging the X-Plane flight simulator to simulate the environment and aircraft dynamics. This approach has found recent success in visualizing and optimizing algorithms for basic UAV flight control as well as formation flight (e.g. see [2, 3]). Whereas existing work focuses on control and simulation of one or multiple UAVs using this methodology, our work uses X-Plane to study the interaction of UAVs and an aircraft controlled by a *human* pilot. Although of timely concern, this latter application has received little attention. This paper describes a very simple prototype setup involving only a single pilot-controlled aircraft and a drone, but several enhancements are described at the conclusion of the paper that will be reported on in a future publication.

The remainder of this paper is organized as follows: Section II describes the general simulation system that was developed based on X-Plane and its UDP interface. Section III presents a simple collision avoidance algorithm that was implemented, which uses path deformation with sequential quadratic programming (SQP). Section IV studies the performance of the algorithm in the presence of a single intercepting aircraft. Section V describes several future enhancements that are planned for the work and gives some concluding remarks.

II. X-Plane Based Simulation System

Figure 1 depicts the main components of the initial X-Plane-based simulation environment. The X-Plane 10 simulator is the heart of the system and implements both the physics engine for the piloted aircraft, referred to as the piloted aerial vehicle (PAV), and rendering of the pilot's view. UAV control is implemented with two processes, accomplished with two instances of MATLAB that communicate via UDP. The navigation process computes the desired path of the UAV and sends this information to X-Plane. The see-and-avoid process receives PAV coordinates from X-Plane and the desired UAV path from the navigation process. When the projected PAV and UAV paths violate minimum safe separation, the SQP-based algorithm deforms the UAV path and sends the modified path to the navigation process.

The UAV in this study is a quad-copter drone. The drone model was created using Google Sketchup,

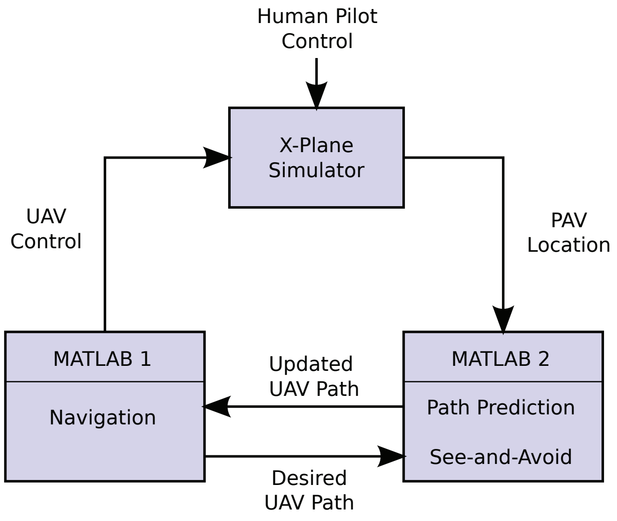

Fig. 1. Components of the X-Plane-based UAS see-and-avoid simulation environment. Aside from the human input at the top, data between the components is exchanged via UDP.

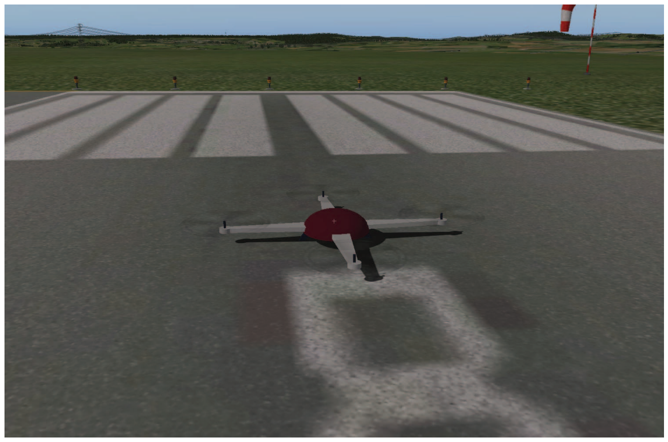

Fig. 2. X-Plane's rendering of the designed drone model.

exported as 3D objects using the SketchUp2XPlane plugin, and then imported into X-Plane's PlaneMaker application. An X-Plane rendering of the drone model is shown in Figure 2. Although the current purpose of the drone model is only for visual rendering of a UAV, the model can be extended in the future to simulate detailed physics and low-level control.

III. Collision Avoidance Algorithm

Figure 3 depicts the basic path variables that were used to develop the see-and-avoid algorithm, consisting of a single piloted aircraft and UAV. In this scenario, the vector $\underline{\mathbf{x}}_U^{(n)}$ indicates the desired position of the UAV at time step n, whereas $\mathbf{x}_U^{(n)}$ and $\mathbf{x}_P^{(n)}$ are the actual positions for the UAV and PAV, respectively, at the same time step.

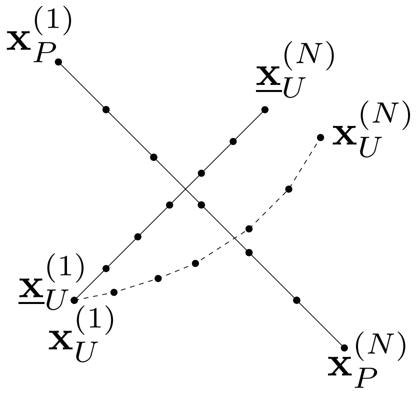

Fig. 3. Path variables defined for the see-and-avoid algorithm, where $\underline{\mathbf{x}}_U^{(n)}$ and $\mathbf{x}_U^{(n)}$ are the desired and actual (deformed) path positions of the drone at time step n, while $\mathbf{x}_P^{(n)}$ is the position of the piloted aircraft.

In this initial work, we consider a simple collision avoidance algorithm that solves

$$\mathbf{x}_U = \arg\min_{\mathbf{x}_U,\epsilon} \sum_n \left[c_1 \|\mathbf{x}_U^{(n)} - \underline{\mathbf{x}}_U^{(n)}\|^2 + c_2\epsilon_n \right], \quad (1)$$

such that

$$\|\mathbf{x}_U^{(n)} - \mathbf{x}_P^{(n)}\|^2 + \epsilon_n > \Delta_{\text{sep}}^2, \quad \forall n, \quad (2)$$

$$\epsilon_n > 0, \quad \forall n, \quad (3)$$

$$\|\mathbf{x}_U^{(n+1)} - \mathbf{x}_U^{(n)}\|^2 < \Delta_{\text{uav}}^2, \quad \forall n \in [1, N-1], \quad (4)$$

$$\mathbf{x}_U^{(1)} = \underline{\mathbf{x}}_U^{(1)}, \quad (5)$$

where Δ_{sep} is the minimum desired UAV-PAV separation, Δ_{uav} is the maximum distance that can be traveled by the UAV in a single time step, and ϵ_n is a slack variable at time n that allows violation of the minimum separation distances with a high cost.

The cost function in (1) jointly minimizes deviation from the desired UAV path with cost c_1 and violation of the safe separation distance with cost c_2, where typically $c_2 \gg c_1$. Constraints (2) and (3) are intended to keep the UAV-aircraft separation larger than Δ_{sep}. Sometimes, this separation may not be possible, and instead of paralyzing the UAV with an infeasible problem, the slack variables ϵ_n allow violation of safe separation with a high associated cost c_2. Note that when safe separation is possible at step n, $\epsilon_n = 0$, resulting in no cost for the slack variable. The constraint (4) limits the distance that the UAV may travel in a single time step, which represents a very simple model for UAV movement. A more advanced formulation of the problem could model limited acceleration and velocity of the UAV, which then would be used to compute UAV position. Finally, constraint (5) requires the UAV to start at a prescribed location.

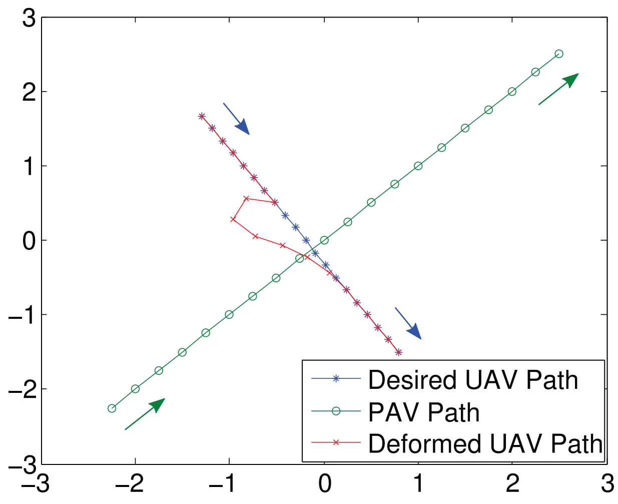

Fig. 4. Example run of the SQP-based see-and-avoid algorithm. Note that each sample is shown with a symbol, where the PAV (green) moves from the bottom left to top right, while the UAV moves from the top-left to the bottom-right.

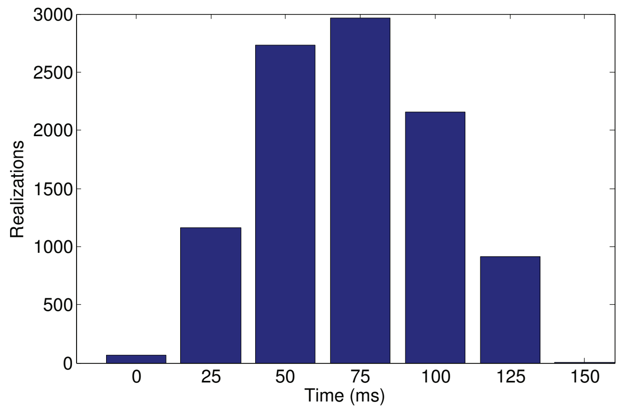

Fig. 5. Histogram of run time for the see-and-avoid algorithm running on 10^5 random realizations of the drone path. The mean run time was 72 ms.

Unfortunately, the problem above is non-convex due to the safe separation constraint (2). Existing research has studied convex semi-definite programming (SDP) relaxations of this constraint [4–6], appropriate for finding approximate solutions or solution bounds. The SDP relaxation approach was also tried in this research, but unfortunately we were unable to obtain useful path deformations this way. Instead, we chose to use sequential quadratic programming (SQP) to find a local minimum to the problem, which seems to work well for the scenario that was considered. For this initial work, the SQP solver that is built into MATLAB's fmincon function was used to simplify the development.

Figure 4 shows a single run of the SQP-based algorithm in MATLAB, where the PAV flies from the bottom-left to the top-right, while the desired path for the UAV is from the top-left to bottom-right, and paths are defined with $N = 20$ points. As can be seen, the algorithm deforms the desired UAV path to provide safe separation from the piloted aircraft, while attempting to change the path as little as possible. To give an idea of the run time of the algorithm, the algorithm was run 10^5 times for $N = 20$ path points, where the UAV had a random starting and ending position in the upper-left and bottom-right quadrants, respectively. Figure 5 shows a histogram of the resulting run time in MATLAB on a 3.6 GHz i7 desktop computer, where a mean time of 72 ms indicates that the algorithm is a good candidate for real-time see-and-avoid control.

Note that in this work, the drone and aircraft positions were limited to a two-dimensional (2D) plane, which not only makes it more challenging for the UAV to avoid the PAV, but also simplifies plotting and analysis of the results. In the future, this can be easily extended to three dimensions.

IV. Research Case Study

This section describes two scenarios that were explored in this initial study. In both cases, a Cessna 172 general aviation (GA) airplane was used for the piloted aircraft.

A. *Computer Controlled Aircraft*

Figure 6 shows the first scenario, where the GA airplane is controlled by the computer to fly in a rectangular pattern around runway 26 at Alexandria Airfield (N85) at a constant speed of 90 knots and altitude 1500 feet above mean sea level (MSL). The quadcopter UAV is programmed to fly in a similar pattern, but in the opposite direction around the middle of the field, thus presenting a serious collision hazard. The desired path has the quadcopter flying at 10 knots, while the maximum quadcopter speed is set to be 25 knots.

The effectiveness of the see-and-avoid algorithm was tested by setting the desired safe separation to be 300 ft and programming the PAV to fly in 100 loops of the pattern. Figure 7 plots histograms of the minimum separation distance for all near collisions for the original UAV path (blue) and that with the see-and-avoid algorithm (red). As can been seen, the separation distance of the PAV and UAV is significantly improved by the algorithm, nearly reaching the desired 300 ft separation distance.

B. *Human Controlled Aircraft*

Figure 8 shows the X-Plane simulator setup that is maintained and used by the Aviation Club at Lafayette

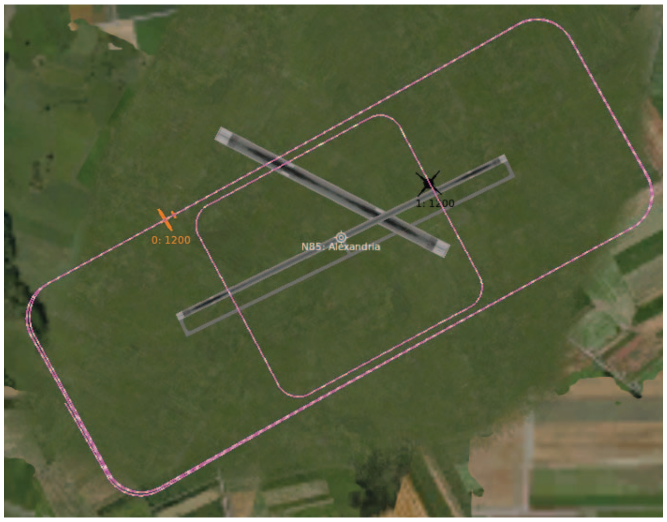

Fig. 6. Visualization of the flight paths for the first see-and-avoid study, generated by the X-Plane instructor console. The PAV (orange) is controlled by the computer to fly in a rectangular course. The drone UAV is shown in black. The path of the PAV and the desired path of the UAV are shown as magenta/white lines.

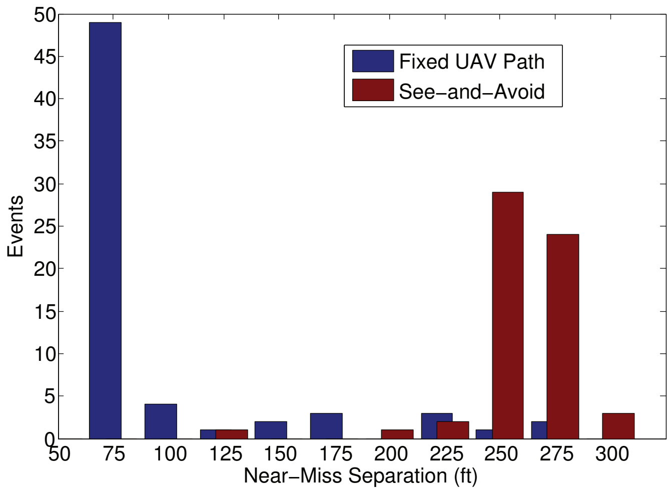

Fig. 7. Histogram of near-collision distances for the scenario of a PAV flying an airport pattern shown in Figure 6.

Fig. 8. X-Plane simulator setup at Lafayette College, consisting of pilot cockpit (left) and instructor control console (right).

College. A similar X-Plane simulator setup was integrated with the collision avoidance components described previously, allowing a human pilot to interact with the UAV. To test the limits of the collision avoidance algorithms, experiments were performed where the UAV was programmed to fly back and forth at 10 knots on a line aligned with runway 23. A human pilot repeatedly flew a Cessna 172 over the airfield attempting to collide with the UAV. We admit that the scenario is contrived, since a real human pilot would not attempt to collide with a UAV, but the intent is to explore a very challenging scenario for the UAV. In the future, we plan to turn the experiment around, where the UAV is placed randomly on paths in front of the PAV. In this case, the pilot should take normal action to avoid the UAV if possible. The experiment was repeated twice, one time with the drone flying a fixed path and a second time with the drone using the see-and-avoid algorithm. For each experiment, the pilot attempted to collide with the drone 20 times, which required about 45 minutes of flight time.

During the experiment it was observed that when the human pilot flew a fairly stable crossing path that the UAV was best able to avoid the PAV. In cases where the human pilot had to make last second changes to the path in an attempt to collide, the UAV had difficulty adapting quickly enough due to the delay of the algorithm. It is expected that a binary code (non-MATLAB) implementation of the algorithm would be much faster, and might not have this limitation. Also, it is expected that a hybrid collision avoidance technique would be most beneficial, where longer term planning is performed using the SQP optimization, whereas last-second collision avoidance for unexpected situations is achieved with a fast, simple algorithm, such as the repulsion-type methods [7].

Minimum separation for each of the 20 crossings was stored for the case of the UAV without and with the see-and-avoid algorithm, and the results are shown in Figure 9. As can be seen, without the algorithm, the PAV was usually able to collide (or nearly collide) with the UAV. On the other hand, with the see-and-avoid algorithm, it was much more difficult to cause a collision. Even though there were cases where near collisions occurred with the see-and-avoid algorithm running, the results are still promising given the difficult nature of escaping from a PAV intent on colliding.

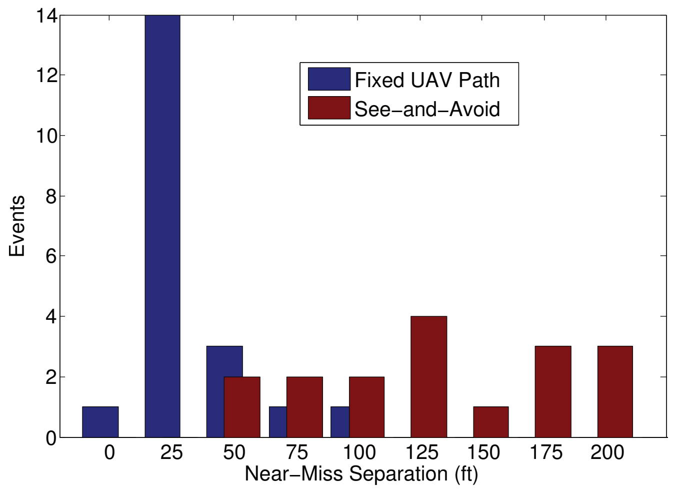

Fig. 9. Histogram of near-collision distances for a human pilot trying to collide with the UAV for 20 crossings of the UAV path. In the first case, the drone is flown on a fixed path, and in the second case, the see-and-avoid algorithm is used by the UAV to avoid collisions.

V. Conclusion

This paper has reported on initial research at Lafayette College whose aim is to develop see-and-avoid algorithms for UAVs that are effective at avoiding collisions with human piloted aircraft, that allow near-optimal path planning, and that can be used for real-time control. The work described herein was to develop a simulation environment that allows the interaction of UAVs and piloted aircraft to be studied in detail. To accomplish this, the popular X-Plane simulator was adopted to allow easy simulation of the flight environment and aircraft physics and to allow aircraft to be controlled by human pilots. We found the open and networked nature of the X-Plane simulator to be nearly ideal for the goals of this research.

Although the initial MATLAB-based simulation components that were developed are somewhat simplistic, several future enhancements are planned for this work. For example, we plan to replace the UDP-based interface with an X-Plane plugin, which should afford more versatility and stability between X-Plane releases. Improvements in the speed of the collision avoidance algorithm are possible by porting the code to a binary implementation that does not depend on MATLAB. The existing SQP-based see-and-avoid path planning algorithm assumed a very simplistic model for the drone, where only the maximum step between time points (velocity) is constrained. In the future, this should be extended so that the drone position is computed using accelerations and velocities that are physically constrained. Additionally, detailed physics could be designed into the UAV model, allowing a realistic autopilot that generates control inputs to be simulated.

The scenarios that were considered in this paper can be seen as just a starting point in the study of the interaction of UAVs and human pilots, and several interesting follow on studies could be performed. For example, it is of interest to study the effectiveness of the see-and-avoid algorithms when a human pilot flies a planned route and a UAV appears out of nowhere crossing the pilot's path. The danger of such scenarios as well as the effectiveness of see-and-avoid UAV algorithms could be studied for different phases of flight, such as during takeoff, airport patterns, en-route, and landing. It is also of interest to understand the effect of evasive action by a human pilot on the effectiveness of the UAV's see-and-avoid algorithm, where the most optimal algorithm would likely take into account expected reactions of human pilots, right-of-way rules, and phases of flight. There would also be a great benefit in generating databases of pilot reactions for many pilots over a wide range of flight experience and aircraft types.

It is expected that as the interaction of human pilots and UAS is studied in detail, not only will better see-and-avoid algorithms for UAVs be developed, but a much clearer picture of the true risks will be realized. This knowledge may pave the way to greater aviation safety and less restrictive rules for UAS operation.

References

[1] C. Whitlock, "Near-collisions between drones, airliners surge, new FAA reports show," *Washington Post*, Nov. 26, 2014.

[2] A. Bittar, H. V. Figuereido, P. Avelar Guimaraes, and A. Correa Mendes, "Guidance software-in-the-loop simulation using X-Plane and simulink for UAVs," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, Orlando, FL, USA, May 27-30, 2014, pp. 993–1002.

[3] R. Garcia and L. Barnes, "Multi-UAV simulator using X-Plane," *J. Intell. Robot. Syst.*, vol. 57, pp. 393–406, Feb. 2010.

[4] B. Alrifaee, M. G. Mamaghani, and D. Abel, "Centralized non-convex model predictive control for cooperative collision avoidance of networked vehicles," in *2014 IEEE International Symposium on Intelligent Control (ISIC)*, Antibes, France, Oct. 8-10, 2014, pp. 1583–1588.

[5] R. Dai, "Three-dimensional aircraft path planning based on nonconvex quadratic optimization," in *2014 American Control Conference (ACC)*, Portland, Oregon, USA, June 4-6, 2014, pp. 4561–4566.

[6] J.-H. Oh, J. M. Shewchun, and E. Feron, "Design and analysis of conflict resolution algorithms via positive semidefinite programming," in *Proc. 36th Conference on Decision and Control*, San Deigo, CA, USA, Dec. 1997, pp. 4179–4185.

[7] A. Alexopoulos, A. Kandil, P. Orzechowski, and E. Badreddin, "A comparative study of collision avoidance techniques for unmanned aerial vehicles," in *2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 13-16, 2013, pp. 1969–1974.