Optimization-Based Path Planning for Separation Assurance on Small Unmanned Aircraft

Matthew Duffield[*], Andrew Ning[†], Timothy McLain[‡]

Brigham Young University, Provo, UT, 84606, USA

This paper presents a time-based path planning optimizer for separation assurance for unmanned aircraft systems (UAS). Given Automatic Dependent Surveillance-Broadcast (ADS-B) as a sensor, intruder information such as position, velocity, and identification information is available at ranges on the order of 50 nautical miles. Such long-range intruder detection facilitates path planning for separation assurance, but also poses computational challenges. The time-based path optimizer presented in this paper provides a path-planning method that takes advantage of long-range ADS-B information and addresses the associated challenges. It is capable of long-range path planning and, due to the convex formulation, is computationally efficient enough to run successively for increased robustness. The ultimate result of this research is a convex, time-based path planner that is suitable for a detect-and-avoid solution on small UAS in the National Airspace System.

I. Introduction

A. Motivation

The number of both public and private applications of unmanned aircraft systems (UAS) is increasing at an amazing rate. Governmental institutions are taking an interest in UAS for their ability to efficiently perform tasks such as weather research, search and rescue, wildlife surveillance, law enforcement, wildfire monitoring, and military training. The US Department of Transportation has projected that by the year 2035 there will be approximately 70,000 UAS operated by US governmental agencies [1]. Private industry is also very interested in UAS applications. Anticipated non-governmental UAS operations include infrastructure inspection, cinematography, precision agriculture, oil exploration, and news and traffic reporting. The demand for UAS operations in the National Airspace System (NAS) is growing rapidly.

The Federal Aviation Administration (FAA) has mandated that for UAS to be permitted in the NAS, UAS must be capable of an equivalent level of safety (ELOS) to the see-and-avoid mandate for manned aircraft [2, 3]. For manned aircraft each pilot has a responsibility to visually scan the surrounding airspace for possible intruding aircraft and take action to avoid a collision. Likewise UAS must be capable of an equivalent degree of monitoring and avoidance of other aircraft. This mandate is known as detect-and-avoid (DAA).

To satisfactorily accomplish the DAA requirement, UAS must be able to both detect other aircraft and plan a collision free path to avoid them. This results in essentially two separate, albeit very related, tasks: detection and avoidance. Many different sensors have been applied to intruder detection for DAA efforts. While radar and visual methods have drawn a particularly large amount of attention [4–7], another promising sensor is Automatic Dependent Surveillance-Broadcast (ADS-B). ADS-B is a cooperative sensor that supports an exchange of position, velocity, and identification information between aircraft at demonstrated ranges of up to 80 nautical miles [8]. In DAA efforts to avoid intruders, such long-range, detailed intruder information is particularly valuable.

Many efforts in collision avoidance focus on small time horizon, reactionary avoidance where the goal is to avoid an imminent collision as quickly as possible [9]. The maximum detection ranges for radar and visual methods on small UAS typically lend themselves to this type of approach. However, with the long-range

[*]Graduate Student, Mechanical Engineering, Student Member AIAA.
[†]Assistant Professor, Mechanical Engineering, Senior Member AIAA.
[‡]Professor, Mechanical Engineering, Associate Fellow AIAA.

intruder information available through ADS-B, the avoidance paradigm can shift to focus on long-range path planning to avoid the possibility of a collision scenario. This is referred to as separation assurance or conflict resolution [10].

Planning a path to assure separation focuses on aircraft maintaining a safe distance between themselves. In focusing on maintaining a safe distance, the probability of a collision decreases significantly due to the fact that the path planner ensures that aircraft never get close to each other. Thus, a failure of the separation path planner results in a loss of separation rather than an immediate collision. In this way planning for separation assurance provides a much higher level of safety than planning for collision avoidance. Some of the challenges that accompany long-range separation path planning are the computational expense of long-range path planning, uncertainty in intruder aircraft positions, and unpredictability of intruder aircraft future maneuvers. To develop a path planning method that offers the benefits of ADS-B-based separation assurance and to mitigate the associated challenges, the goal of this research is to develop a convex optimization-based path planner for separation assurance on UAS in a dynamic environment.

B. Relevant Literature

Other research has addressed the problem of optimal path planning between waypoints for UAS in the presence of obstacles. UAS path planning methods such as rapidly-exploring random trees (RRT) and tree branching algorithms have been demonstrated in relatively short-range applications with great success [11, 12]. In long-range applications, however, the computational resources necessary to plan a long-range path with these methods grows exponentially. Jointly optimal collision avoidance (JOCA) is another proposed path planning method for DAA on UAS [13]. While it is capable of planning a path that maintains self-separation, it only plans 30 s into the future. In the case of a small UAS moving at approximately 20 kt, this may only be approximately 1000 ft or less than a sixth of a mile. Thus, JOCA does not take advantage of the long-range information available through ADS-B. The method does have a way to increase the look ahead distance, but it comes with exponentially increasing cost. Airborne Collision Avoidance System X (ACAS-X) is an FAA-funded research effort to provide conflict/collision avoidance logic for manned aircraft [14]. The research will likely result in a system that can be modified for small UAS, but this system is not ready for implementation. Some estimates indicate that it will be approximately ten years before the system is fully implemented and operational.

Optimization-based paths can be formulated to achieve long-range path planning with limited computational resources. Foo, Knutzon, Oliver, and Winer presented a three-dimensional path planning optimizer for UAS using a particle swarm algorithm [15]. This method uses a hybrid objective function that had user-defined weights for fuel minimization and threat avoidance. While the work demonstrated avoidance of ground threats, it did not address dynamic aerial threats. Additionally, it relied on an operator to select the final weighting distribution between fuel minimization and threat avoidance.

A linear programming, three-dimensional path planning method for UAS is presented by Chen, Han, and Zhao [16]. This research is particularly applicable to the separation assurance path planning challenge. It presented a linear programming method to plan a path in the presence of dynamic obstacles. The reported execution time is suitable for real-time applications. The overall goal of the algorithm was to find the optimal path along which a UAS could pursue a moving target and avoid obstacles. This work is relevant to separation assurance path planning, but the goal and scenario are different. The scenarios demonstrated in the article have distances on the order of 7.5 nmi. This is significantly less than the 15-50 nmi range expected in a separation scenario. Ultimately, it is likely that Chen, Han, and Zhao's work could be transformed into a separation assurance path planning method, but further work is necessary to accomplish and demonstrate this.

Optimization-based path planning has been successfully demonstrated, but an opportunity remains to apply it to a self-separation DAA context. To do so computational expense must be a paramount consideration. Convex optimization is a method that offers guaranteed convergence and excellent computational efficiency. As such it is a promising candidate for optimization-based path planning for DAA. Ultimately an optimization-based path planner that takes advantage of long-range ADS-B information, mitigates imperfect knowledge of intruder positions, and accounts for the time-varying nature of intruder positions is necessary. The path planner presented in this paper provides such solution.

II. Problem Formulation

The approach for this research is to use convex, constrained optimization techniques to optimize the position of waypoints, or nodes, along a path so as to find the minimum length path. The problem formulation, robustness measures, and underlying assumptions are detailed in this section. The overall problem formulation uses a squared Euclidean distance as the objective function. The Cartesian coordinates of each node are the design variables, and the self-separation criterion serves as a linear constraint. This formulation is given formally as

$$\begin{aligned}
\text{Minimize} : & \sum_{i=1}^{n-1} (x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2 && (1)\\
\text{With respect to} : & \; x_1, x_2, ..., x_n, z_1, z_2, ..., z_n && (2)\\
\text{Subject to} : & \; ||[x_i, y_i, z_i] - [x_{i-1}, y_{i-1}, z_{i-1}]||_2 \leq \mathit{Threshold} && (3)\\
& \; x_{min} \leq x_i \leq x_{max} \text{ or} && (4)\\
& \; z_{min} \leq z_i \leq z_{max}.
\end{aligned}$$

A. Objective Function

The squared Euclidean distance formula shown in Equation (1) provides an intuitive choice for a path-length-minimization objective function. Excluding the square-root operation from the typical Euclidean distance formula simplifies the objective function and reduces it to a second-order function. While the output of the objective function no longer represents the length of the path, and consequently is less intuitive, the overall optimization is more efficient. More importantly the second-order formulation of the distance formula is convex, where as the standard Euclidean distance is not.

B. Design Variables

The design variables for the optimization are the x and z coordinates of each of the path nodes in the local frame where the x axis lies perpendicular to the nominal path in the horizontal plane and z axis points upward. The y coordinate for each node lies along the nominal path and is excluded to reduce the dimensionality of the problem. This exclusion does not significantly reduce the flexibility of the solution due to the fact that prior to optimization the positions of the start and end node can be linearly transformed to both lie on the y axis. In such an orientation, the most impactful coordinate variation will occur orthogonal to the y axis. Thus, the y coordinate is an unnecessary dimension. Figure 1 shows the orientation of the x-y-z local coordinate frame.

Another key consideration in selecting the design variables is the number of nodes between the start and end node. A greater number of nodes adds more degrees of freedom to the solution, and increases the computational requirements of the optimizer. Fewer nodes requires less computational expense, but also reduces the conformability of the path. A brief analysis of the effect of the number of nodes is shown in Section IV of this paper.

C. Constraints

The constraints are designed to ensure that at every time step the ownship maintains self-separation from each intruder aircraft. Additionally the constraints are designed to be linear so as to ensure the convexity of the problem [17]. The constraint in Equation (3) is straightforward. It requires that two nodes not be separated by more than a given distance. This prevents the nodes from being so far apart that an intruder can be between two nodes without violating the bound constraint in Equation (4). Although the constraint in Equation (3) is not linear, it is a convex formulation of a sphere. The constraints in Equation (4) are formulated such that the nodes must be between an upper and lower bound constraint, *min* and *max* respectively. The value of these constraints is determined by the position of the intruders. Using the speed of the ownship, it is possible to calculate the time when the ownship will arrive at each node. With the position and velocity estimates of the intruder, it is also possible to linearly extrapolate the states of the intruders into the future to determine their positions when the ownship arrives at each node. The intruder positions at the appropriate time become the bound constraints for each path node. The constraint is offset from the nominal position of each intruder so as to provide the recommended buffer for self-separation. A set of heuristics is applied to determine whether each intruder should be considered an upper or lower bound. The heuristics are necessary to allow for a broader feasible region than would be available with only one set of linear constraints by systematically determining whether each constraint should be an upper or lower bound. A detailed description of the heuristics is included Section III of this paper.

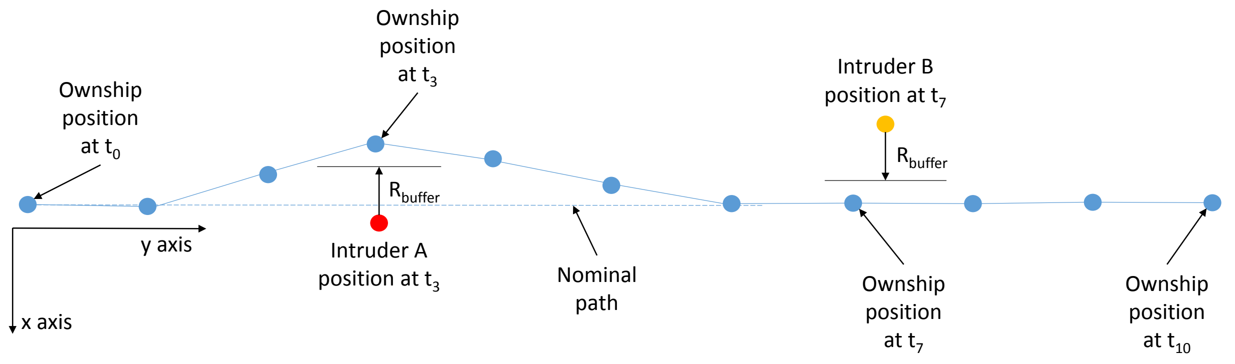

Ownship position at t_0
Ownship position at t_3
Intruder B position at t_7
R_{buffer}
R_{buffer}
y axis
x axis
Intruder A position at t_3
Nominal path
Ownship position at t_7
Ownship position at t_{10}

Figure 1. An example of how the upper and lower constraints are formed.

Figure 1 provides a graphical example of how the constraints are determined. The waypoint at time t_3 is constrained to be above the line created by the position of intruder A. This constraint is offset from the nominal position of the intruder by R_{buffer} and results in a lower-bound constraint. At the node for t_7, the position of intruder B creates an upper bound constraint which constrains the position of the corresponding path node. By combining the upper and lower bound constraints for each path node, the feasible optimization space becomes a channel or corridor between the two sets of constraints. While this method does require preprocessing of the intruder positions and velocities, it results in a set of convex, linear constraints.

D. Robustness

The nature of long-range, time-based path planning requires extrapolation of intruder positions over long time horizons. This necessity introduces two forms of error: state error and model error. State error results from the growing uncertainty as imperfect state information is propagated into the future. Model error also grows as it is predicted into the future, but it results from uncertainty in the model by which the information is propagated. To mitigate these two types of error, we use a successive replanning method. Successive replanning robustness is a method by which the path is re-planned at regular intervals. While this is not a new method, it does significantly contribute to the overall applicability of the path planner. Velocity uncertainty, unpredictable maneuvers, and environmental factors such as changes in wind can be addressed by regularly replanning the optimized path. This is a major reason that the computational expense of the optimization routine is of interest. With a more rapid path planning method, new plans can be generated more quickly and more often.

E. Assumptions

For this problem formulation several assumptions are necessary. The assumed sensor with which intruder information is gathered is ADS-B. This sensor provides latitude, longitude, altitude, ground speed, heading, and climb rate. The intruder positions and velocities used to propagate intruder positions into the future are derived from this information. Furthermore the propagation method for intruder positions is a constant-velocity method. Thus, it is assumed that the intruders are not maneuvering. While this assumption may not be entirely correct, successive path replanning can alleviate much of the error in the assumption. The distances used in the intruder buffer, R_{buffer}, are proposed well clear definitions from the Sense and Avoid Science and Research Panel (SARP) [18, 19]. In the horizontal plane $R_{buffer} = 4000$ ft, and in the vertical plane $R_{buffer} = 500$ ft.

III. Optimization Implementation

While the formulation of the optimization is rather straightforward, the implementation leverages several key aspects that lead to the success of the optimization. One of the primary goals of the implementation is to achieve and maintain a convex problem formulation. In Equation (4), only one set of upper and lower bound constraints must be satisfied, either the vertical or the horizontal. While each individual set of constraints is convex, the "either-or" formulation is not convex. An "and" formulation of the horizontal and vertical constraints would be convex, but such a formulation would result in avoidance paths that are not representative of true aircraft behavior which rarely combines vertical and horizontal maneuvers. To mitigate this we address each direction, vertical and horizontal, separately. Essentially we run separate optimizations in the same sequential planning iteration. In the first run, we allow the z coordinates to vary, and we impose the vertical set of constraints. For the second run, we allow the x coordinates to vary, and we impose the horizontal set of constraints.

As mentioned previously a set of heuristics is necessary to determine whether an intruder should be represented as an upper or lower bound constraint. In our implementation we use nine heuristics, four in the vertical direction and five in the horizontal direction. This results in nine separate optimization runs. While this does require multiple optimization runs, the efficiency of convex optimization allows for this sequential optimization without imposing too great a computational cost. It is important to note that while none of the heuristics is individually valid for all intruder scenarios, the collective set is designed to find at least one feasible path for a wide range of intruder configurations.

A. Constraint Heuristics

The first heuristic is horizontal and categorizes the intruders based off of the straight-line path between the start and end node. Essentially if an intruder has a position on the positive x axis then the constraint will be an upper bound constraint. Alternatively, if the intruder's position is on the negative x axis then the constraint will be a lower bound constraint. This heuristic is intended to identify the straight-line path between the start and end node. The second heuristic is also horizontal. It seeks to find a path to the right of all intruders. Simply put this method identifies all intruders as lower bound constraints. If all intruders are moving on the left side of the ownship, then this method will find a feasible path. The third heuristic is an analog to the second in that it seeks to find a path to the left of all intruders. All intruders are viewed as an upper bound. Thus, if all of the intruders are to the right of the ownship, this heuristic will lead to a feasible path. The fourth heuristic also seeks a path to the right of the ownship, but is looks for a gap in the traffic. In other words, instead of trying to fly either along the straight-line path or to the right of all of the intruders, it looks for a gap between the intruders on the right hand side. This allows the ownship to find a path that is much shorter than going around all intruders. If there is not gap available, this heuristic results in a path that is identical to the path around the right of all of the intruders. The fifth heuristic is identical to this method except that it looks for a gap to the left. Figure 2 shows the hypothetical resultant paths for each of the first five heuristics.

The sixth, seventh, eighth, and ninth heuristics are vertical heuristics. They are direct analogs to the second, third, fourth, and fifth heuristics shown in Figure 2 in that they look to fly above all the intruders, below all of the intruders, through a gap above the straight line path, and through a gap below the straight-line path respectively. After all of the optimization runs have been completed, the final path is selected based on the length of the path. Thus, the shortest overall path from all nine of the optimization runs is selected as the path. Overall, this set of nine heuristics significantly increases the feasible optimization space. It also allows for a convex "either-or" formulation for the horizontal and vertical formulation.

B. Convex Solver

To solve the convex optimization we used CVX, a package for representing and solving convex programs [20, 21]. Due to the fact that this tool was designed to run with MATLAB, it was very useful for our development. For implementation on a small UAS platform, however, we would use a solver that is compatible with C++ or some other compiled language. Implementation in a compiled language would result in much greater computational efficiency.

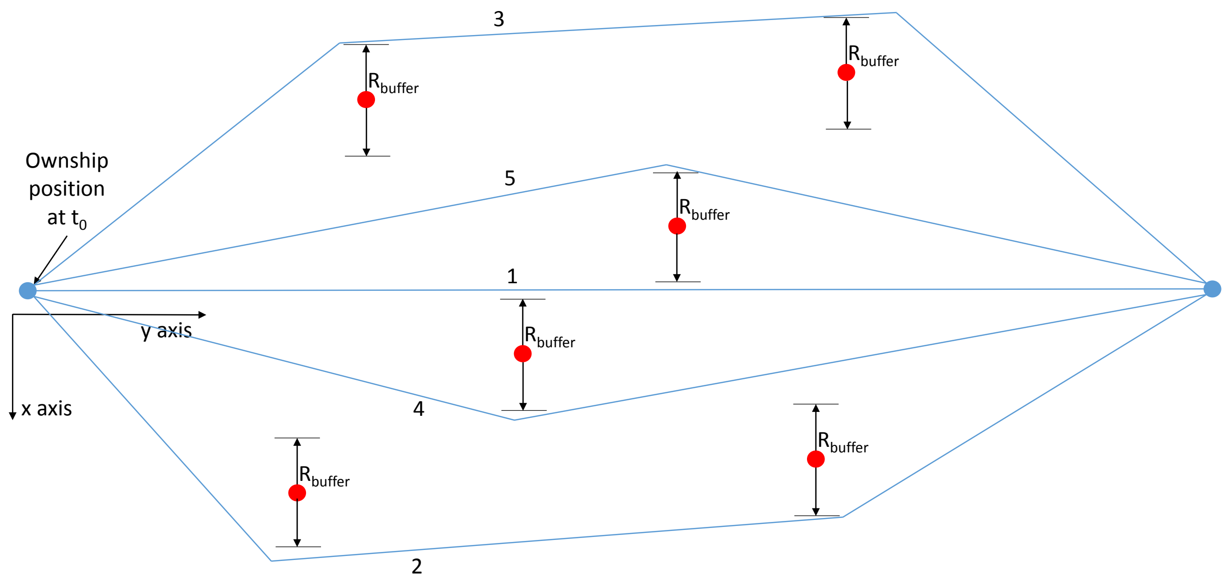


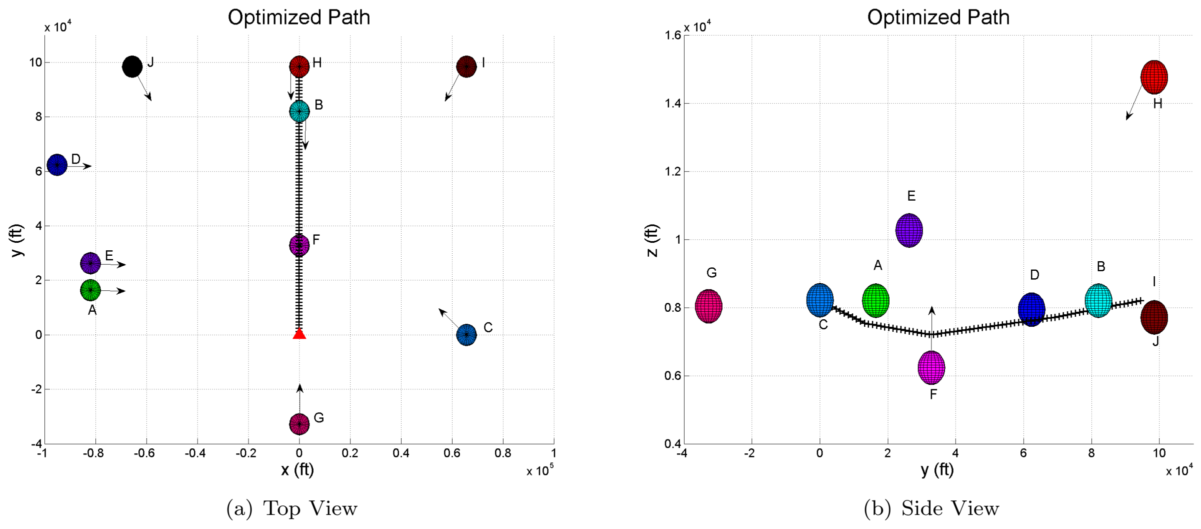
Figure 2. A diagram showing hypothetical path results from each of the first five heuristics.

IV. Results

Testing of the convex optimization-based path planner for self-separation included single iteration feasibility testing, an analysis of the number of nodes and run time, and a simulation-based test of the sequential path planning robustness.

A. Single Iteration Testing

The goal of the single iteration feasibility testing was to determine how the optimizer responds in scenarios with a high number of intruders. To test this, the optimizer had to find a feasible path between a start and end node in the presence of ten intruders. The start and end point of the path were placed at (0, 0, 8200) and (0, 91141, 8200) feet respectively. The path between the start and end nodes had 78 evenly spaced intermediate nodes.



(a) Top View

(b) Side View

Figure 3. Initial positions and velocities of the intruders.

Figure 3 shows the initial intruder states for the scenario. The circles in the figure represent the self-separation volume of the intruders. The red triangle is the ownship, and the nodes of the path are represented by the crosses along the black ownship path. The arrows in the figure show the direction of travel for each of the intruders. From the figure it is clear that there are crossing intruders from both sides, a head-on intruder, an overtaking intruder, a descending intruder, and a climbing intruder. Each of the intruders are positioned so as to interact with the ownship either by crossing the ownship's path or by flying near to the path so as to increase the complexity of the solution space. To provide a quantitative perspective, Table 1 lists all of the initial intruder positions and velocities. The alphabetical labels in the table correspond to the labeled intruders in Figure 3. Ultimately this set of intruders represents a very complex self-separation scenario.

Table 1. Initial intruder positions and velocities.

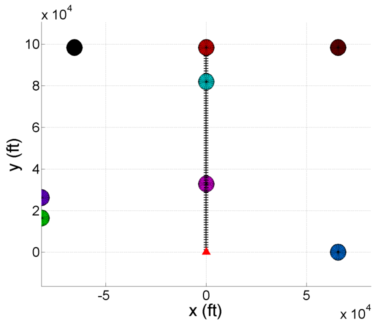
Intruder	Px (ft)	Py (ft)	Pz (ft)	Vx (kt)	Vy (kt)	Vz (ft/min)
A	-82021	16404	8200	97.1	0	0
B	16	82021	8200	0	-19.4	0
C	65617	0	8215	-29.2	29.2	0
D	-95144	62336	7940	77.8	0	0
E	-82021	26247	10266	97.2	0	0
F	0	32808	6234	0	0	984.3
G	0	-32808	8038	0	97.2	0
H	0	98425	14764	0	-55.4	-787.4
I	65617	98425	7710	-58.3	-58.3	0
J	-65617	98425	7710	149.7	-233.3	0

With the intruder configuration shown in Figure 3, we ran the convex optimization-based path planner for self-separation. The results of the optimization are shown in Figure 4. To provide a three dimensional, time-variant perspective of the results, Figure 4 includes subfigures that show the ownship progress along the optimized path. Figures 4(a)-4(c) show a top, end, and side view, respectively, of the optimized path while the intruder is at the initial node. The subsequent rows show a similar perspective when the ownship is at node #20, #50, and #80 respectively. A video of the path optimization result is available at `https://youtu.be/StR4tSKYhhY`.

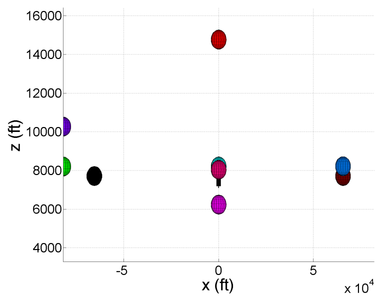
From Figure 4 it is clear that the optimizer maintains self-separation from all of the intruders throughout the entire path. This is the primary goal of the optimization. It is interesting to note that the optimizer selected a vertical avoidance path. This results from the fact that the self-separation definition requires 4000 ft of separation in the horizontal plane, but only 500 ft of separation in the vertical plane. Thus, it is reasonable that the optimizer nearly always finds a vertical avoidance path [22]. Another reason that the optimizer typically finds a vertical avoidance path is that the ownship is moving slowly compared to the intruders. The ownship is designed to resemble a small UAS, and it is only moving at 29 kts. This is slower than all but one of the intruders. Thus, the ownship is unable to outrun the intruders. In finding a horizontal avoidance path with linear constraints, this means that it is very easy for intruders crossing from different directions to pinch off the channel through which the ownship might fly. Thus, there is a greater likelihood that a vertical path will be feasible than there is that a horizontal path will be feasible.

To determine the limitations of the planner, we identified and tested several conditions where the planner fails to find a feasible solution. If the self-separation volumes are excessively large, the optimizer is unable to find a viable solution. We tested the optimizer with collision volumes equivalent to the FAA ATC-separation volumes for manned aircraft. These volumes require 5 nmi of separation horizontally and 500 ft of separation vertically. While the optimizer is able to find a feasible path for some of these scenarios, it fails to find a feasible path for scenarios with more than approximately 5 intruders. This is a result of the solution space being overwhelmed by the large separation volumes. Another condition where the path optimizer fails is when there is an intruder on top of, or very near to, either the start or end node. This prevents the ownship from approaching the end node, to which it is constrained. Thus, a feasible path cannot be found.

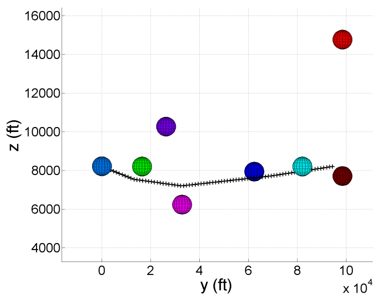
In extended testing the optimizer successfully found a feasible self-separation path in the presence of



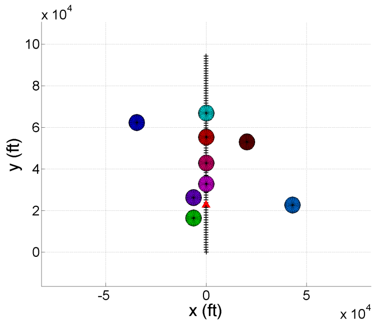
(a) Top View at Node #0.



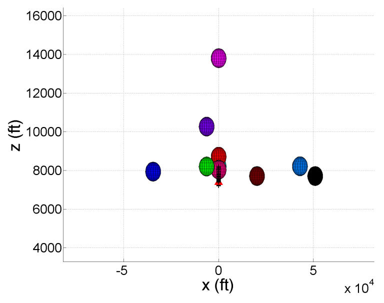
(b) End View at Node #0.



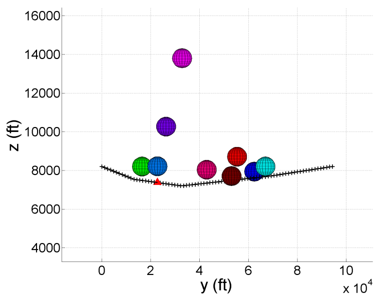
(c) Side View at Node #0.



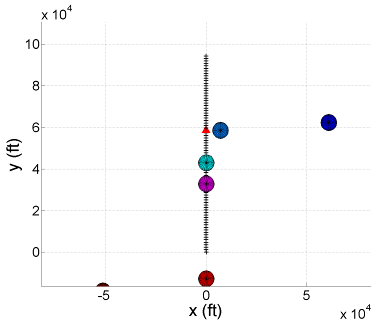
(d) Top View at Node #20.



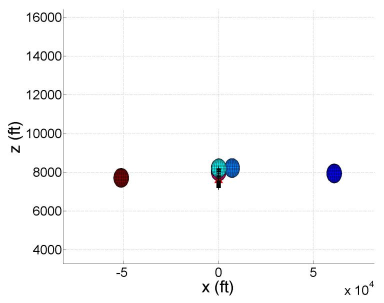
(e) End View at Node #20.



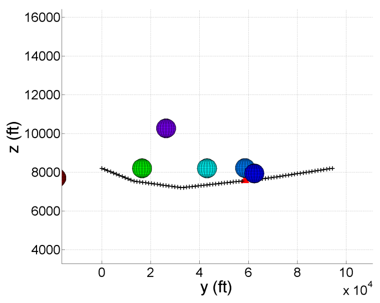
(f) Side View at Node #20.



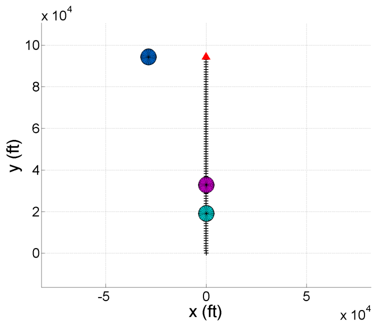
(g) Top View at Node #50.



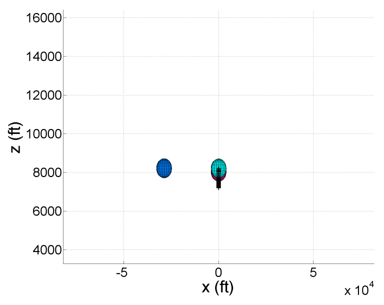
(h) End View at Node #50.



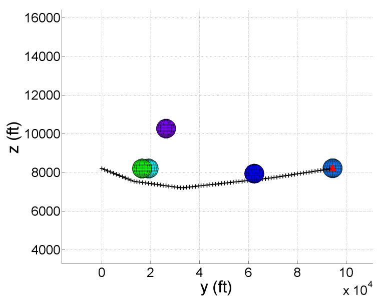
(i) Side View at Node #50.



(j) Top View at Node #80.



(k) End View at Node #80.



(l) Side View at Node #80.

Figure 4. Self-separation path as time progresses.

up to 19 intruders on a collision path with the nominal path of ownship. This scenario included multiple crossing, climbing, and descending intruders from all directions. The ability of the optimizer to successfully plan a path in the presence of 19 intruders converging on the ownship is a testament to the viability of the planner and the robustness of the nine heuristics.

Table 2. Change in run time and path length as a function of the number of nodes.

# of Nodes	CPU Time (s)	Path Length (ft)
60	3.45	94411.90
80	3.82	94411.54
100	4.50	94412.61
120	4.74	94412.08

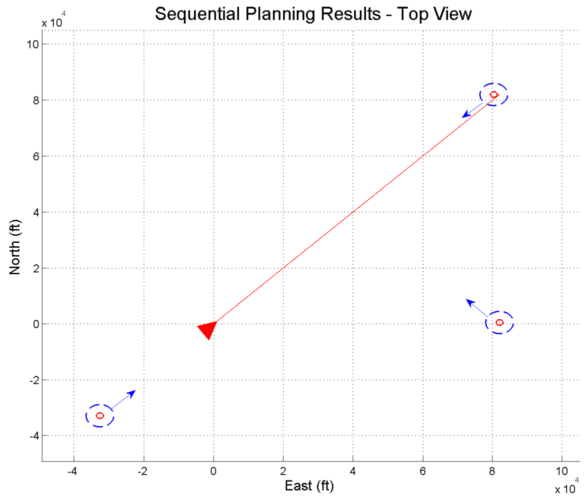
Table 2 shows an analysis of the CPU time verses the number of nodes along the path. The optimization runs for this table were executed in MATLAB on a 3.1 GHz CORE i5 processor. One very interesting correlation in the results is that as the number of nodes increases, the run time increases slightly and the path length remains essentially unchanged. This correlation indicates that when choosing the number of nodes it is reasonable to err on the side of fewer nodes.

Another important observation from Table 2 is that all of the CPU times shown are slower than the 1 Hz measurement rate of ADS-B. While this initially seems to be a serious downfall of the path planning method, it is not a significant drawback. Any implementation of this path planner on a UAS would require a conversion from MATLAB to C++ or a another compiled language. In converting from MATLAB to a compiled language, such as C++, the computational expense of the planner will significantly decrease. An additional factor that reduces the impact of slower-than-real-time computation is that since the resulting path of the planner is time-based, it does not lose validity over time in the same way that a non-time-based method does. A time-based path is computed using future positions of both the ownship and intruders. Thus, it is theoretically always valid. There is uncertainty in the propagated intruder positions and the timing associated with the path nodes, but this uncertainty is less impactful than the uncertainty associated with considering the intruders to be static. Thus, as a result of using an interpreted language for testing, and the increased validity period associated with a time-based path, the seemingly slow run times are not a significant concern.

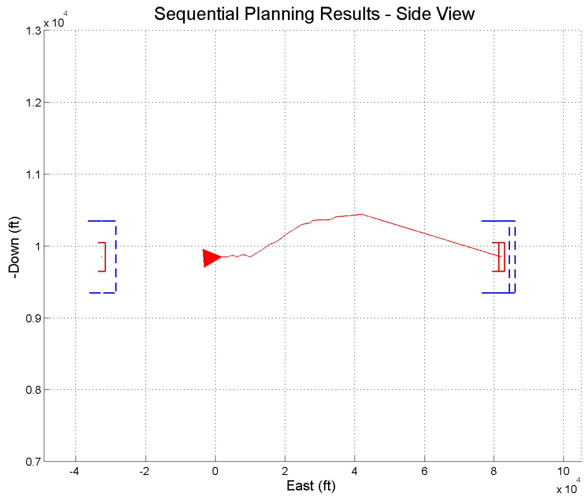
B. Successive Path Planning Testing

The robustness testing was done using a full simulation to capture effects of the successive path planing robustness. The simulation executes the full dynamics of a 12-state ownship and three 12-state intruders. In the simulation, the ownship is flying between two waypoints that are separated by 18.6 nautical miles. One intruder is overtaking the ownship from behind at a closing speed of 29.2 kts. Another intruder is approaching head-on at a closing speed of 77.8 kts, and a third intruder is crossing the path of the ownship from the right. All intruders are at the same altitude as the ownship. To generate ADS-B messages, the truth data from each of the intruders is corrupted using the error characterization based on FAA regulations for ADS-B error. Every 10 seconds the path optimizer plans a new path for separation assurance. The planner first checks the old path to determine whether it is still valid. If it is valid the optimizer outputs the previous path and no new optimization is performed. If the path is no longer valid, then the planner reoptimizes the waypoints. In light of the fact that the optimization occurs in the local frame of ownship as described in Section II, it is necessary to linearly transform the global frame coordinates of the start waypoint, end waypoint, and intruder states into the ownship local coordinate frame. After the optimization the optimized path is transformed back into the global reference frame for the path following algorithm. Figure 5 shows the initial simulation scenario and the path flown by the ownship.

In Figure 5 a top view is shown in the left plot and a side view is shown in the right plot. The red triangle is the ownship. The red line shows the actual path flown by the ownship over the course of the approximately 40 minute simulation with successive path replanning every 10 seconds. Each of the intruders is surrounded by both a blue circle indicating the separation volume and a red circle indicating the collision volume. Their positions in the figure show their initial positions and trajectories at the beginning of the



(a) Top View



(b) Side View

Figure 5. Initial intruder configuration and the avoidance path flown by the ownship during the sequential planning simulation.

simulation. Table 3 shows the maximum ownship elevation change, the maximum ownship climb angle, and the number of seconds in a loss of separation (LOS) state.

Table 3. Key metrics for the fully simulated path optimizer.

Metric	Value
Max Ownship Elevation Change	592.70 ft
Max Ownship Climb Angle	0.75 deg
Number of Seconds of LOS	7.06 s

The value of the optimization path planner is illustrated by Figure 5 and Table 3. The maximum elevation change and maximum climb rate listed in Table 3 demonstrate that the path planner can avoid a loss of separation with gradual maneuvers over a long time horizon. These gradual maneuvers are beneficial in that they support minimal deviations from peak efficiency operating conditions such as airspeed. They also present a relatively small variation from the original path and mission goals. Table 3 does indicate that there are short periods of time where the separation thresholds are not preserved. These violations are a result of the error in the intruder estimates. Due to the fact that each of the violations occurs at the top outer bound of the separation volume, the violation of the thresholds is of minimal concern. Additionally improved estimation of the intruder states would result in less error and thus the path planner could entirely prevent LOS. The small climb rate, reasonable maximum altitude change, loss of separation results, and robustness effectiveness combine to demonstrate the value of the optimization path planner presented.

V. Path Planning Conclusions

In conclusion, the convex, time-based path optimizer presented in this paper is capable of long-distance path planning for separation assurance in an environment with dynamic obstacles. The convex optimization formulation allows for a large number of nodes which leads to high resolution long-range planning without excessive computational cost. The incorporated robustness measures result in a path planner that is viable in the presence of uncertainty in intruder positions, velocities, and future maneuvers.

Additionally the ability to maintain separation assurance in the presence of head-on, overtaking, and crossing intruders illustrates the value of ADS-B information. In scenarios with intruders traveling much faster than the ownship, the combination of ADS-B information and the time-based path optimizer allows the ownship to maintain separation with gradual maneuvers. Ultimately the time-based path optimizer presented in this research is a capable, long-range path planner. As such it is a valuable path planner for separation assurance and a key step toward a DAA solution for UAS in the NAS.

Acknowledgments

This work has been funded in large part by the Center for Unmanned Aircraft Systems, a National Science Foundation-sponsored industry/university cooperative research center (I/UCRC). Additional funding was provided by the Utah NASA Space Grant Consortium.

References

[1]U.S. Department of Transportation, "Unmanned Aircraft System (UAS) Service Demand 2015-2035," Tech. rep., U.S. Air Force, 2013.

[2]Hottman, S., Hansen, K., and Berry, M., "Literature Review on Detect, Sense, and Avoid Technology for Unmanned Aircraft Systems," Tech. Rep. September, 2009.

[3]Federal Aviation Administration, "Subchapter F - Air Traffic and General Operating Rules," 2015.

[4]Mackie, J., Spencer, J., and Warnick, K., "Compact FMCW Radar for GPS-Denied Navigation and Sense and Avoid," *IEEE Antennas and Propagation Society, AP-S International Symposium*, 2014, pp. 989–990.

[5]Dey, D., Geyer, C., Singh, S., and Digioia, M., "Passive , long-range detection of Aircraft : Towards a field deployable Sense and Avoid System," *Field & Service Robotics*, 2009, pp. 1–10.

[6]Lai, J., Ford, J. J., Mejias, L., Shea, P. O., and Walker, R., "See and Avoid Using Onboard Computer Vision," *Sense and Avoid in UAS: Research and Applications*, chap. 10. See an, 2012, pp. 265–294.

[7]Mejias, L., McNamara, S., Lai, J., and Ford, J., "Vision-based detection and tracking of aerial targets for UAV collision avoidance," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2010, pp. 87–92.

[8]Moody, C. and Strain, R., "Implementation Consideration for Automatic Dependent Surveillance - Broadcast on Unmanned Aircraft Systems," *AIAA Infotech@Aerospace Conference*, American Institute of Aeronautics and Astronautics, Reston, Virigina, April 2009, pp. 1–8.

[9]Sahawneh, L. R., Beard, R. W., Avadhanam, S., and Bai, H., "Chain-based Collision Avoidance for UAS Sense-and-Avoid Systems," *Aerospace Robotics and Unmanned/Autonomous Systems IV*, 2013, pp. 1–15.

[10]George, S., "Concepts of use for UAS Sense and Avoid Equipment," Tech. rep., 2009.

[11]Klaus, R. A., Mclain, T. W., Beard, R. W., and Colton, M. B., *Development of a Sense and Avoid System for Small Unmanned Aircraft Systems*, Ph.D. thesis, Brigham Young University, 2013.

[12]Beard, R. and Mclain, T. W., *Small Unmanned Aircraft: Theory and Practice*, Princeton University Press, 2012.

[13]Chen, W.-Z., Wong, L., Kay, J., and Raska, V. M., "Autonomous Sense and Avoid (SAA) for Unmanned Air Systems (UAS)," Tech. rep., 2009.

[14]Kochenderfer, M. J., Holland, J. E., and Chryssanthacopoulos, J. P., "Next-generation airborne collision avoidance system," *Lincoln Laboratory Journal*, Vol. 19, No. 1, 2013, pp. 17–33.

[15]Foo, J. L., Knutzon, J. S., Oliver, J. H., and Winer, E. H., "Three-dimensional path planning of unmanned aerial vehicles using particle swarm optimization," *Mechanical Engineering Conference Presentations, Papers, and Proceedings*, No. September, 2006, pp. 1–10.

[16]Chen, Y., Han, J., and Zhao, X., "Three-dimensional path planning for unmanned aerial vehicle based on linear programming," *Robotica*, Vol. 30, No. 05, 2012, pp. 773–781.

[17]Boyd, S. and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[18]Johnson, M., Mueller, E. R., and Santiago, C., "Characteristics of a Well Clear Definition and Alerting Criteria for Encounters between UAS and Manned Aircraft in Class E Airspace," *11th USA/Europe Air Traffic Management Research and Development Seminar*, Lisbon, Portugal, 2015.

[19]Cook, S. P., Brooks, D., Cole, R., Hackenburg, D., and Raska, V., "Defining Well Clear for Unmanned Aircraft Systems," *AIAA Infotech@Aerospace*, No. January, Kissimmee, FL, 2015, pp. 1–20.

[20]CVX Research, I., "CVX: Matlab Software for Disciplined Convex Programming, version 2.0," `http://cvxr.com/cvx`, Aug. 2012.

[21]Grant, M. and Boyd, S., "Graph implementations for nonsmooth convex programs," *Recent Advances in Learning and Control*, edited by V. Blondel, S. Boyd, and H. Kimura, Lecture Notes in Control and Information Sciences, Springer-Verlag Limited, 2008, pp. 95–110, `http://stanford.edu/~boyd/graph_dcp.html`.

[22]Krozel, J. and Peters, M., "Strategic conflict detection and resolution for free flight," *Proceedings of the 36th IEEE Conference on Decision and Control*, Vol. 2, No. December, 1997, pp. 1822–1828.