AIAA SciTech Forum
4-8 January 2016, San Diego, California, USA
AIAA Infotech @ Aerospace

UAV Collision Avoidance using a Predictive Rapidly-Exploring Random Tree (RRT)

Jeremiah Farinella[1], Clayton Lay[2], and Subodh Bhandari[3]
Cal Poly Pomona, Pomona, CA, 91768

This paper presents a method for the collision avoidance and path planning algorithm for small autonomous Unmanned Aerial Systems (UAS) to operate autonomously and safely in obstacle rich environments. The proposed method uses Predictive Rapidly-Exploring Random Tree (RRT) algorithm to safely navigate around multiple obstacles or other aircraft. The RRT algorithm guarantees a collision-free path, and maneuvers UAS's around randomly generated dynamic obstacles in a simulated environment to the specified goal waypoint. The algorithm assumes the availability of the Automatic Dependent Surveillance-Broadcast (ADS-B) sensors and secondary sensors such as a scanning Lidar for collision detection. These sensors coupled with the Predictive RRT algorithm guarantees collision free autonomous navigation in a dynamic environment without the need for a human pilot.

Nomenclature

δ_e	=	change of elevator angle, deg
δ_T	=	change of throttle control, deg
F_x	=	X force, lb
F_y	=	Y force, lb
F_z	=	Z force, lb
I_x	=	moment of inertia about x-axi, slug-ft^2
I_{xz}	=	product of inertia about xz-plane, slug-ft^2
I_y	=	moment of inertia about y-axis, slug-ft^2
I_z	=	moment of inertia about z-axis, slug-ft^2
L	=	rolling moment, ft-lb
M	=	pitching moment, ft-lb
m	=	mass, slugs
N	=	yawing moment, ft-lb
p	=	roll rate, deg/sec
$\dot{p}$	=	rate of roll rate, deg/sec^2
q	=	pitch rate, deg/sec
$\dot{q}$	=	rate of pitch rate, deg/sec^2
r	=	yaw rate, deg/sec
$\dot{r}$	=	rate of yaw rate, deg/sec^2
u	=	forward velocity, ft/sec
$\dot{u}$	=	forward acceleration, ft/sec^2
v	=	lateral velocity, ft/sec
$\dot{v}$	=	lateral acceleration, ft/sec^2
w	=	vertical velocity, ft/sec
$\dot{w}$	=	vertical acceleration, ft/sec^2
X	=	X force, lbs

I. Introduction

UNMANNED Aerial Systems (UAS's) offer many advantages over manned aircraft for several applications. The main advantages are: a) they are cheaper and simpler than manned aircraft, b) they can fly to places that are impossible for manned aircraft to fly to, and c) they pose no risk to human operators. Military application

[1]Graduate Student, Aerospace Engineering Department.
[2]Graduate Student, Aerospace Engineering Department, and AIAA Student Member.
[3]Professor, Aerospace Engineering Department, 17-2116, and AIAA Senior Member.

range from target recognition and tracking, intelligence, surveillance, and reconnaissance (ISR), and battlefield damage assessment. The civilian applications include remote sensing, transport, scientific research, rescue missions, surveillance of fire-, flood-, and earthquake-affected areas, synthetic aperture radar (SAR) interferometry, vegetation growth analysis, assessment of topographical changes, etc. The role of UAS's is expected to increase in future years. UAS's are the fastest growing sector of the aerospace industry, and it is expected that the UAS spending will be doubled over the next decade from current worldwide expenditures.[1,2]

Despite their huge potential for replacing manned aircraft for a number of applications, the use of UAS's is limited to restricted airspaces or for research purposes by academia or Government agencies only, with prior approval from the Federal Aviation Administration (FAA) .Widespread use of these vehicles for commercial applications requires that these systems be integrated into the National Airspace System (NAS). This, in turn, requires that the UAS's have collision and obstacle avoidance capabilities. These capabilities are precondition for the integration of UAS's into the NAS, and have potential to enable the UAS's to fly alongside manned aircraft without any safety concerns.[3,4] Opening the NAS to UAS's will only become possible by satisfying the FAA's "sense and avoid" requirements. UAS's must have human equivalent level of safety (ELOS) to autonomously avoid collisions and navigate around them.[5]

UAS's operate without a pilot onboard and in some cases need to operate completely autonomously, based on the mission profile or loss of interaction with the remote operator. In such cases, UAS's solely rely on onboard sensors for collision avoidance. A number of problems can occur when the pilot is not directly in the loop or has limited airspace awareness.

- Situation awareness only from telemetry data acquired by onboard sensors sent to the ground control station (GCS).
- Latency issues (up/down-link) in the communication datalink exist due to the data transfer rates between UAS and the GCS.
- Potential failures such as loss of data link, jamming, et cetera can occur. In such cases UAS's must be able to autonomously recover the mission.
- Existing air traffic sensors such as Mode-C or IFF (Identification Friend or Foe) transponder that can reliably help the ATC detect and redirect manned aircraft are not feasible for many UAS's because their weight, size, and power requirements exceed payload constraints by a large margin.

Currently, the FAA is in a transition period upgrading to the Next Generation Air Transportation System (NextGen) modernization of Air Traffic Control (ATC) radar surveillance.[6] The FAA is deploying a relatively new technology called Automatic Dependent Surveillance-Broadcast (ADS-B), where aircraft equipped with Global Positioning System (GPS), both manned and unmanned, can transmit and receive the locations and altitudes of other nearby aircraft, as well as can receive the updates from air traffic control.[7]

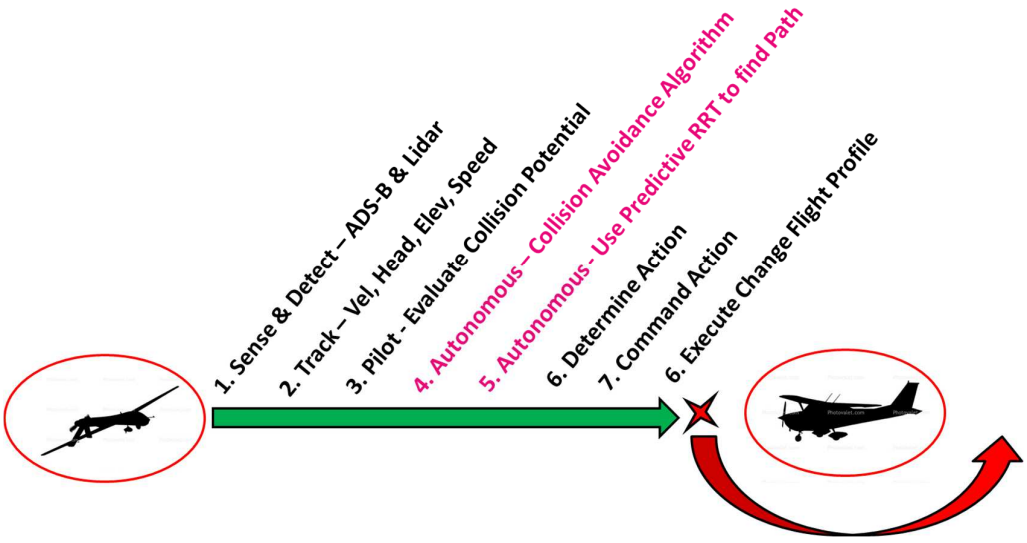


Figure 1. Sense and avoid timeline

As shown in Fig. 1, UAS's utilize the sense and avoid timeline to evaluate an obstacle threat and safely navigate around one or more obstacles. During the sense and avoid timeline, a collision avoidance and path planning algorithm is implemented as shown in the figure (event 4 and 5).

Widespread and cost-effective use of UAS will require the missions carried out autonomously using a series of waypoints to a set destination or end goal. In dynamic and complex environments, mission profiles cannot always be preplanned. When unforeseen changes occur, re-planning can be a computationally burdensome task and cannot always be completed in a timely manner. UAS air traffic separation has a number of issues that can be attributed to the growing number of UAS's in operation, FAA regulation, current air traffic systems' ability to detect, and the limits of remote pilot operations. Many techniques have been used or proposed for path planning features in real-time for navigating UAS's around obstacles such as Genetic Algorithms, Potential Functions, Graph Constructions, and Rapidly-Exploring Random Trees (RRT's).[8-11] Despite existing works on developing obstacle avoidance and real-time path planning capabilities for UAS's, technologies to safely maneuver the UAS's in obstacle rich environments have not matured yet. Also, computationally capable and cost-effective system requires a custom avionics system.[12]

Autonomous system architectures have explored using RRT algorithms for stationary and dynamic path planning for both air and ground vehicles.[13] Many of these studies have been utilizing a rotary wing aircraft and don't address how to deal with other airborne obstacles in the environment.[8] RRT is an algorithm designed to efficiently search nonconvex high-dimensional spaces by randomly building a space-filling tree, as shown in Fig. 2.[14,15] The tree is constructed incrementally from randomly drawn sample nodes from the search space and is inherently biased to grow towards large unsearched areas of the problem. RRT algorithms have the ability to rapidly explore dozens of path planning options in an environment full of obstacles and navigate the lowest cost collision-free path to the specified waypoint.

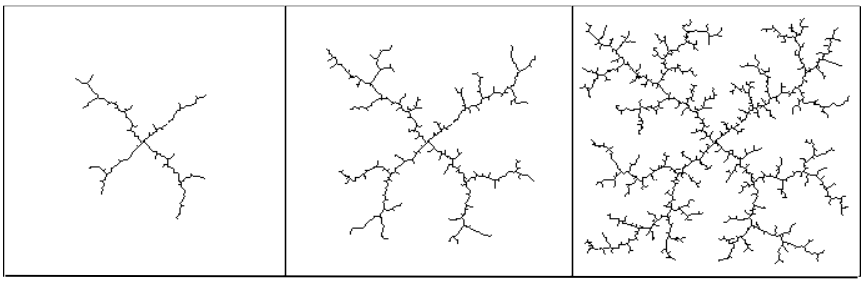


Figure 2. Holonomic RRT growth convex region

This paper proposes a Predictive RRT algorithm[16] to address several shortcomings of the various path-planning methods that are described in the following paragraphs.

The paper is organized as follows: Second section talks about the various path planning methods used for collision avoidance and path planning including Predictive RRT method and the pseudo code for 2-D Predictive RRT algorithm. The 2-D simulation results are presented in Section III. In Section IV, UAS being used for this research is presented along withthe flight dynamics model of the UAV for the 3-D RRT implementation. The 3-D simulation results are presented in Section V followed by the conclusion and future work in Section VI.

II. Collision Avoidance Algorithm

A. Path Planning and Waypoint Navigation

The planned path for a UAS is usually described in terms of a set of waypoints or specified coordinates that the aircraft has to visit sequentially. The UAS flight control systems coupled with the guidance and navigation algorithms provide location updates for the flight profile that meet the minimum requirement for navigation to the desired waypoint. In addition to navigating to the desired waypoints, it's necessary to have a collision avoidance algorithm that ensures collision-free navigation in obstacle rich environment that contains static, pop-up, and dynamical obstacles. When obstacles appear in the flight path, UAS's must be able to provide safe separation, maintain nominal flight profile towards the desired path, and drive the final flight computation to the desired waypoint safely. There are many path planning algorithms such as randomized potential field, probabilistic roadmaps, forward search method,and RRT.[17-20]

Randomized potential field (RPF) provides gradient descent naturally for incremental simulation. However, it is not suitable to achieve optimal solution and is difficult to deal with local minima. They are also slow to achieve the solution.[18]

Probabilistic roadmaps (PRM) provide very uniform exploration of unsearched spaces and can be easily analyzed. However, connecting the nodes is complex and computationally expensive. Also, it requires solving two point value problem, which makes computational times long. PRM can build the graph of state-space, but lack generating a path to the desired goal destination.[17,19]

Forward search method is a general nonholonomic planner. However, it is limited to defined grid space, and requires high computational timeline due to search state.[20]

RRT requires little to no heuristics, and the method is suited for high-dimensional space. However, the completeness of all possible solutions is sacrificed. They also perform poorly in scenarios where there are bottlenecks and/or long narrow corridors.[21] Despite some of these shortcomings, RRT was chosen for this research for the reasons described below.

B. RRT as the Desired Solution

As discussed above, the RRT is an algorithm used to explore large state-space in a relatively short time by randomly expanding and building a space-filling tree in unsearched regions. The RRT method used provides the quickest approach for a solution to navigate through high-dimensional space, and provides a guaranteed collision-free path, which can quickly be repeated as the environment's dynamics change over time. The main drawback of the RRT is that the explored available paths are not always optimized for the most direct path to the goal destination. However, alterations to the algorithm such as goal biasing, increasing node population, and repeating scans of the environment more often can make the path more optimal.[22] When the shortest and most optimal path is found or a maximum number of scan iterations are reached, the RRT terminates.

The RRT algorithm solution used in this research relies on a deterministic representation of the environment from the onboard sensor suite, where the algorithm knows a priori if a node is collision-free or lies within an obstacle. With the current available UAS sensor technology and ADS-B systems being implemented,[3] it's assumed that obstacles within the flight environment can be identified and their locations known at any given time. Since the location of obstacles in a dynamic environment is a function of time, it is essential to collect new information of obstacle locations and update the path planning of the RRT.

C. Predictive RRT

The proposed solution is an extension of the basic RRT algorithm and takes into account the initial obstacle position during the exploration of the scanned space and the predicted future positions that will be traveled by the obstacles over a short period of time.[19] All the explored configurations are maintained in the explored tree in any given iteration, and new scans will update position data and will drive new explored tree construction. The UAS will follow a solution from initial point to an end goal using waypoint navigation from node to node. The solution described can be conceived as a Predictive RRT method using waypoint navigation. Figure 3 shows 12 iterations of the Predicative RRT, where each iteration is displayed in a different color until the solution converges to the goal state.

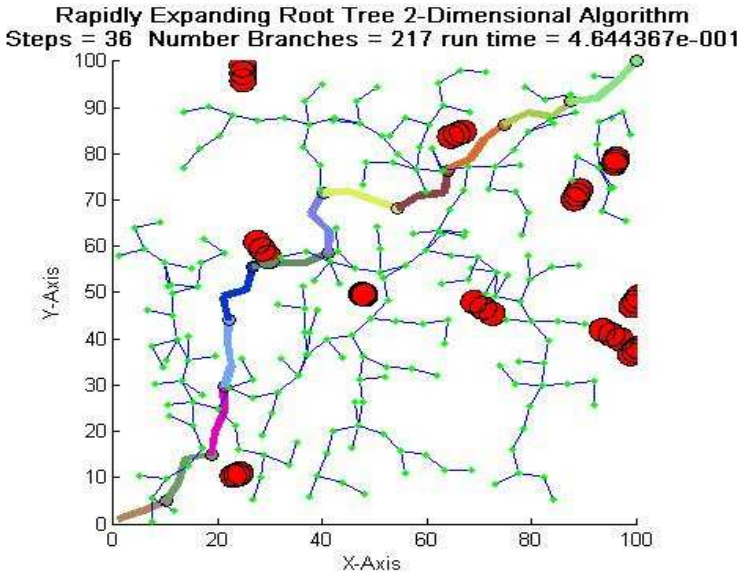


Figure 3. Predictive RRT goal convergence

The Predictive RRT used in this analysis is designed to efficiently and quickly explore open spaces and apply simple waypoint navigation in a dynamic environment. The advantage of the RRT for UAS waypoint navigation is the algorithm's ability to find a feasible solution between complex dynamic obstacle configurations. Each execution of the algorithm provides a scan of the airspace or "snap shot in time" for all available collision-free paths.

In order to make the algorithm predictive, the obstacles' trajectories are taken into account and are used to project and predict obstacle positions and overlay them on the path planning grid. The obstacle positions over time are walled off as unavailable space for path planning. The RRT is implemented and a path tree is constructed for explored collision-free paths. In each iteration, the algorithm solution takes into account the respective time based obstacle locations to the times associated to the generated branches then maps a path for a small number of specified waypoints for the UAS to navigate. Once the UAS has traveled this small specified number of waypoints, the sensor data from the first scan is refreshed and used to determine any obstacle velocity and/or heading changes. The process of updating data can be repeated and a new scan of the environment can be implemented as needed until the desired goal waypoint is reached. Each iteration is biased toward the goal waypoint which reduces computation time and provides a more direct path toward the goal destination.

Figure 4 shows the predictive multiple path iterations to reach the end goal. The figure shows the number of steps and branches, and the run time.

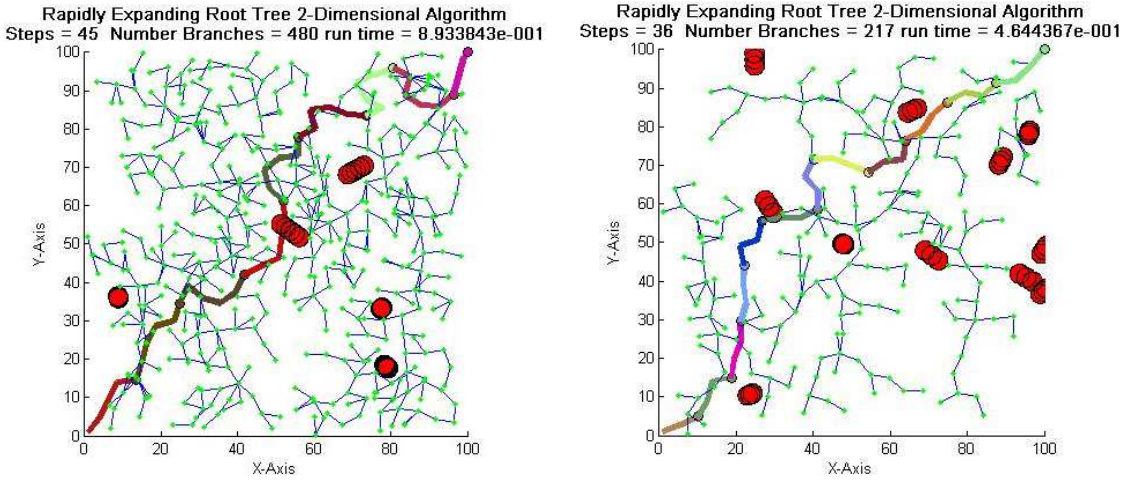


Figure 4. Predictive RRT multiple path iterations

D. Pseudo Code for Predictive RRT Algorithm

As shown in Fig. 5, an initial node (Xinit) and a goal node (Xgoal) are chosen in unexplored space where the UAS plans to travel. In order to grow the RRT, a random node (Xrand) is chosen in two-dimensional free space. The random point (Xrand) that is nearest to the parent (Xnear) is selected. The new random point generated is a check to see if it resides within the bounds of an obstacle, obstacles' predicted positions in the future, or lies intersecting the existing tree.

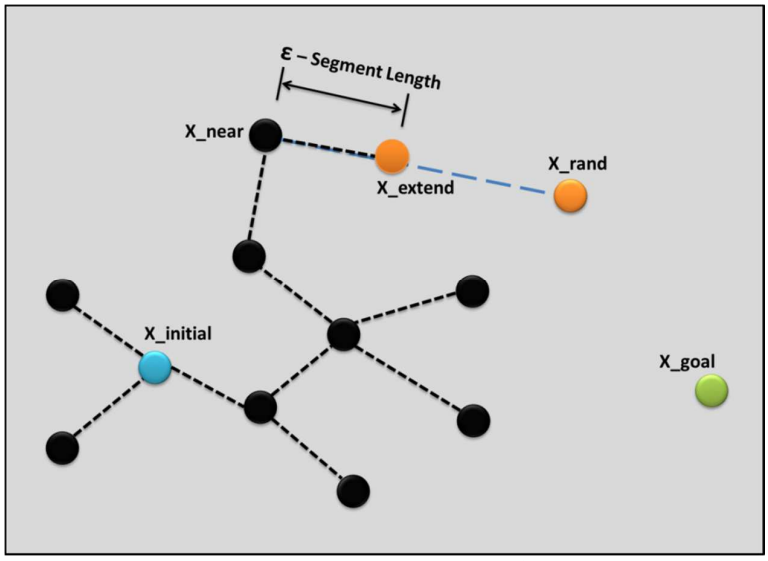


Figure 5. Basic RRT expansion

The Euclidean distance is used to find the distance between the parent node (Xnear) and the random node (Xrand), and then is normalized based on the desired segment length between the branches to become the extended node (Xextend) as given by Eq. (1) below. The normalization is an important metric in order to keep the tree growth rate and expansion under control during the iteration process.

$$d(q,p) = \frac{\sqrt{(q_1-p_1)^2+(q_2-p_2)^2}}{(\text{SegLength})} \tag{1}$$

where $\mathbf{p} = (p_1, p_2, \ldots\ldots, p_n)$ and $\mathbf{q} = (q_1, q_2, \ldots\ldots, q_n)$ are two points in Euclidian n-space.

The algorithm then moves from Xnear toward Xrand to produce a new point Xextend. The new extended point produces a single branch of the tree. This process is repeated until the required number of iterations is reached to path plan a collision-free path to the end goal node. The algorithm then analyzes all possible paths and determines the lowest cost or shortest collision-free connecting path to the goal waypoint (Xgoal). The collision-free "path" of the UAS is biased toward the goal waypoint and is restricted to navigating between a small set of specified waypoints before the sensor data needs to be refreshed. Pseudo code variable inputs have been created and random dynamic obstacles generated in the simulation environment. The flow of the pseudo code is given below.

1. Choose an initial node ***Xinit***(starting location UAV) add it to ***Tree_array*** do;
2. **While** node ***Xgoal***has not been reached || path has not been found to end node;
3. Pick a random node ***Xrand***in free space;
4. Determine the node ***Xnear***in ***Tree_array***nearest ***Xrand***, finding Euclidean distance;
5. Eliminate any extraneous nodes from this path **return** ***Tree_array;***
6. Move incremental segment length from ***Xnear***to ***Xrand***, resulting in state ***Xextend;***
7. Search for obstacle collisions along the true trajectory from ***Xnear***to ***Xextend;***
8. **if** collisions not found between ***Xnear***and ***Xextend***then;
9. Add the node ***Xextend***, and an edge from ***Xnear***to ***Xextend***, to ***Tree_array****;*
10. Try to connect ***Xextend***directly to the goal by performing collision testing;
11. **if** collisions not found between ***Xextend***and ***Xgoal***then;
12. Add the node ***Xgoal***, and an edge from ***Xextend***to ***Xgoal***, to ***Tree_array;***
13. **end if**
14. **end if**
15. **end while**
16. Complete **path** by finding the lowest costoptimal connecting path from nodes X***init***to ***Xgoal;***
17. Repeat iterations for new scan when location data time has elapsed or needs to be refreshed.

III. 2-D Simulation and Results

The Predictive RRT algorithm was tested using multiple simulation runs to generate complete solutions that convergence on desired goal waypoints picked. The Predictive RRT algorithm performs this task by connecting a series of random waypoints to form a tree containing several collision-free paths. The dynamics of the environment are accounted for with each updated scan by performing sequential RRT path iterations that are biased towards the goal waypoint. Sequential RRT path iterations were optimized to find the shortest available connecting path towards the biased goal state. A complete solution is reached when the series of RRT iterations has converged on the goal destination. The results below in the Fig. 6 show a single execution of the Predictive RRT algorithm in which 10 iterations are required to reach the goal state. Each iteration can be viewed as a smaller path planning subset that makes up the total collision-free path when added in series. UAS navigates each node in the collision-free path to guarantee the safety of the aircraft. Each path color represents an iteration in the path planning cycle. For any particular iteration at that given time, no collisions occur.

All test runs were successful at providing a collision-free path to the specified goal point using iterative waypoint navigation. Approximately 100 test runs were executed using a variety of number of obstacles sets with the simulation generating random locations, headings, and velocities for each obstacle. There are five parameters that can be altered: obstacles, segment length, steps, spacesize, and radius. The tunable parameters used for optimizing the algorithm were: obstacles, segment length, and steps. The number of obstacles selected was tested for both low and high values. Test results shown display 5, 15, and 30 obstacles at the start of first iteration. "Segment length" was changed and tested for values between 1 and 10. Values outside this range yielded undesirable results for a "spacesize" of 100. "Segment length" is a function of "spacesize" and would yield different desirable segment length range if "spacesize" is altered. The segment length for the results was optimized at a value of 5.

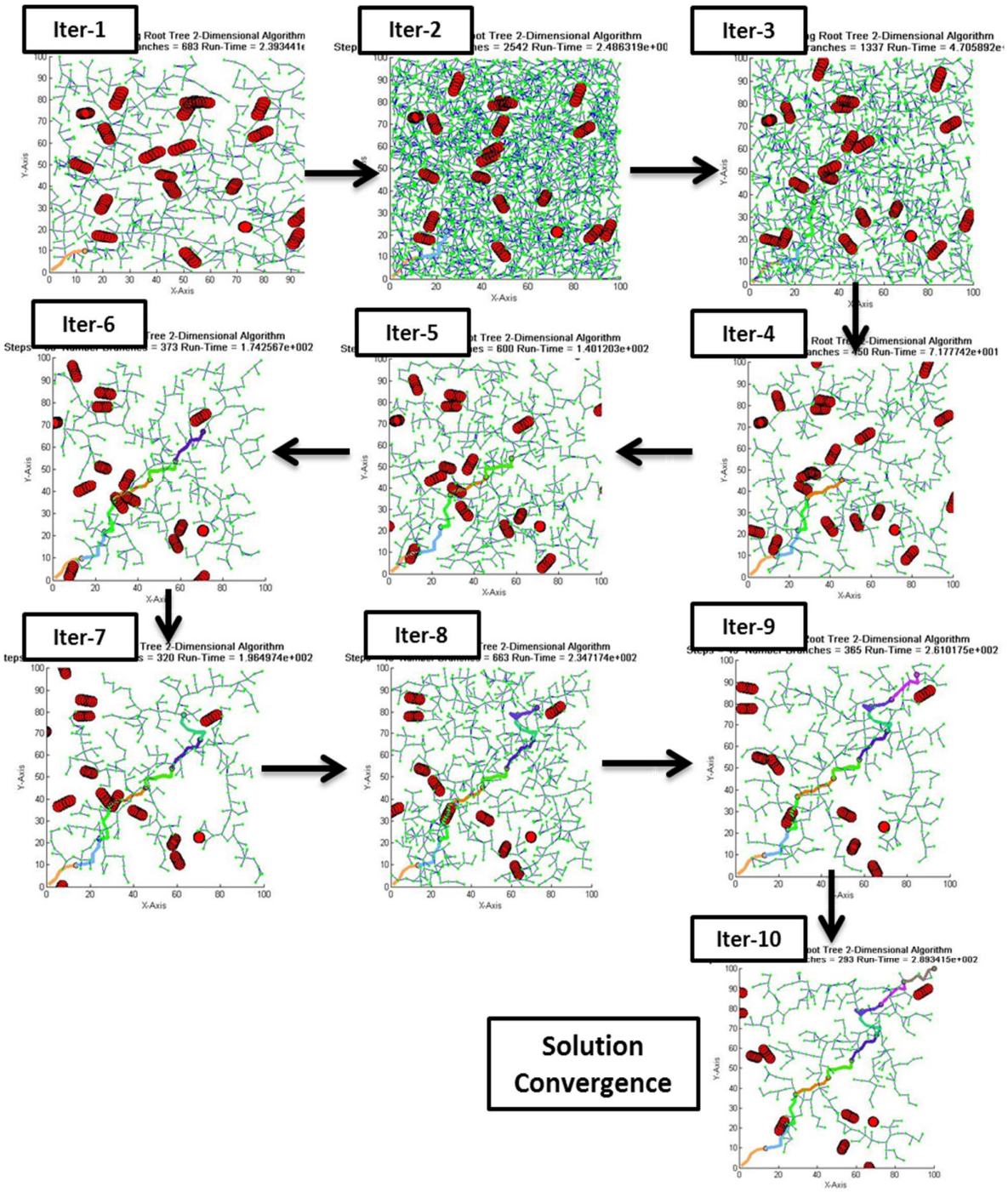


Figure 6. 2-D predictive RRT execution and solution convergence

The parameter "steps" corresponds to the number of waypoints traveled by the UAS in a single iteration or scan. "Steps" shares an inverse relation with the number of obstacles present in the simulated environment. Optimization "steps" value was decreased when obstacles in the simulated environment was high, and "steps" was increased when the obstacles in the simulated environment was low. The test was conducted for 5, 15, and 30 obstacles, where starting node was at [0,0] and goal node set at [100,100]. For runs 1 to 3, the starting obstacles set was 15, for runs 4 to 6 the starting obstacle set was 30, and for runs 7 to 9 the starting obstacle set was 30. Upon the final iteration, there may appear to be fewer than the starting number of obstacles and this is contributed to obstacles travel outside the field of regard (FOR) of the UAS.[23] Figure 7 shows the example for test runs 1 and 2.

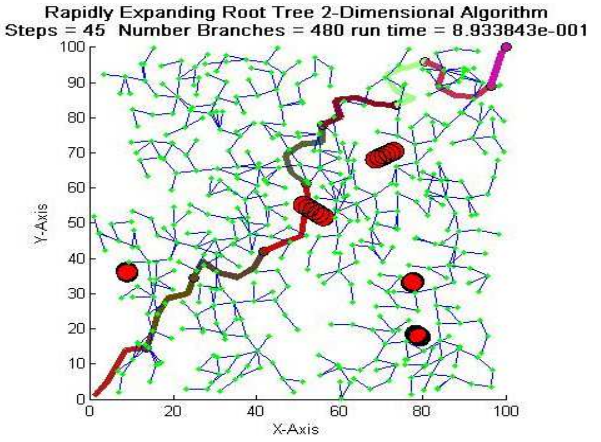
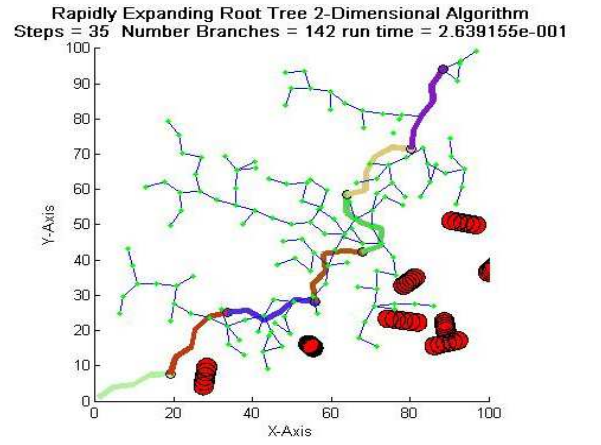
Test Runs	Scans Req'd	Time/Scan (sec)	Branches/Scan	RRT Time Cost	Convergence of Final RRT Solution to Goal Waypoint
1	1	1.551	619	17.308	
	2	2.0436	971		
	3	1.7503	819		
	4	2.3154	1098		
	5	1.0697	467		
	6	1.4976	609		
	7	2.4866	783		
	8	3.7002	1225		
	9	0.89338	480		
2	1	7.92	784	27.666	
	2	0.82137	410		
	3	1.5894	799		
	4	1.6587	821		
	5	1.241	664		
	6	6.7248	469		
	7	6.7248	3331		
	8	0.98625	542		

Figure 7. Test runs 1 and 2 (obstacles=15, segment length=5, & steps=5)

Each test run ranges from 8 to 14 iterations, and partial path trajectories before the goal solution convergence is reached. The exploring tree is updated with a new scan of the simulated environment for each RRT iteration, and the final state of the previous trajectory becomes the root of the new exploring tree. The algorithm works in series and a new exploring tree is generated for each new scan of the environment. This conservative approach allows the UAS to navigate through the dynamic environment and guarantees that the nodes of each scan exist in a collision-free state in space for the obstacles' current position and projected position data. The time duration of path planning iterations depends on the complexity of the obstacle geometry lying between the last position of the UAS and the goal waypoint. Computational times required for a new exploring tree ranged between 0.2 to 8 seconds. In the results, generated exploring trees that yielded low computational times of 0.2 seconds showed low density branches of 300 to 400, whereas exploring trees with computational times of up to 8 seconds yielded high density more complex trees of around 3000 to 4000 branches. During experimental tuning of the algorithm, parameter sets in some cases generated paths that resulted in stalemate situations, where obstacle geometry and exploring tree did not yield an available path and the algorithm terminated. The stalemate issue can occur when too many obstacles are present in the environment and their predicted paths form a wall. This coupled with the algorithm provides the UAS too many steps to navigate in a single iteration. The issue was corrected by reducing the number of "steps" or waypoints traveled by the UAS during a single iteration of the partial path. Another issue encountered was a high density of obstacles and a large amount of steps an iteration provided a solution to the UAS contained a path with bendy and unrealistic turns, as shown in Fig. 8, which shows 50 obstacles and their predicted positions for 5 steps in a single iteration. This example shows that high density obstacles and obstacle geometry can create a wall where the collision-free path is unrealistic for the UAS flight path.

In order to reduce the clutter or the "wall off" effect in an obstacle high density environment, the steps can be reduced. Reduced steps would drive an increase in iterations and computation time, but would provide a more realistic path for the UAS in a high density environment as shown Fig. 9.

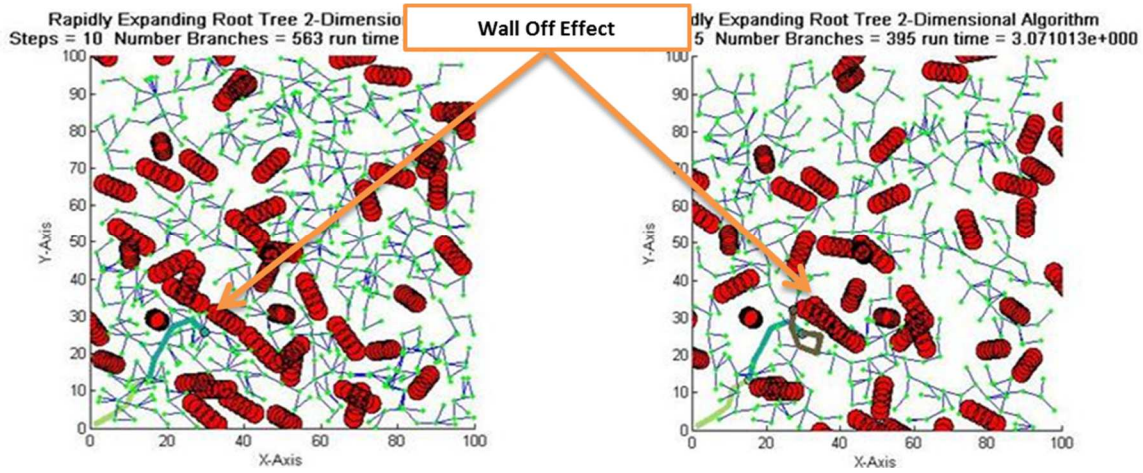


Figure 8. High density environment & unrealistic planning flight path(obstacles=50, steps=5)

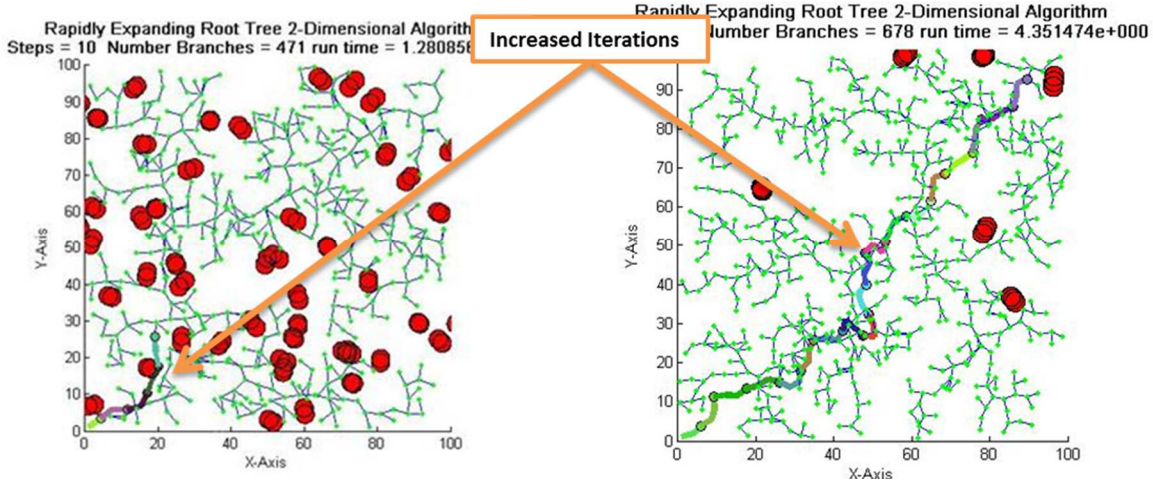


Figure 9. High density environment with reduced steps (obstacles=50, steps=2)

IV. Application to an Unmanned Aerial System

As stated previously, one of the issues encountered was bendy and unrealistic turns for a fixed-wing aircraft; therefore, an aircraft model was incorporated which would depict more realistic flight paths. The aircraft used for this research was the Sig Kadet Senior airplane, which is shown in Figure 10.

A flight dynamics model for the airplane was incorporated into the simulation as well as expanding into 3-D space. The model consists of the longitudinal and lateral-directional dynamics. These are linear state-space representations of the aircraft's dynamics that predict the behavior of the aircraft to control inputs. The state-space model is developed from equations of motion that describe the rigid body motion of an aircraft. There are three nonlinear force and three moment equations that describe this motion, shown below[24]:

$$F_x = m(\dot{u} + q \cdot w - r \cdot v) \tag{4}$$

$$F_y = m(\dot{v} + r \cdot u - p \cdot w) \tag{5}$$

$$F_z = m(\dot{w} + p \cdot v - q \cdot u) \tag{6}$$

$$L = I_x \cdot \dot{p} - I_{xz} \cdot \dot{r} + q \cdot r(I_z - I_y) - I_{xz} \cdot p \cdot q \tag{7}$$

$$M = I_y \cdot \dot{q} + r \cdot p(I_x - I_y) + I_{xz} \cdot (p^2 - r^2) \tag{8}$$

$$N = -I_{xz} \cdot \dot{p} + I_z \cdot \dot{r} + p \cdot q(I_y - I_x) + I_{xz} \cdot q \cdot r \tag{9}$$

Figure 10. Sig Kadet Senior airplane in flight

The total force and moments result from contributions of aerodynamic, propulsive, and gravitational forces. The forces and moments are functions of instantaneous values of perturbation variables. Perturbation variables are the instantaneous changes from the reference conditions. Applying small perturbation theory and Taylor series expansion about a trim point, these equations can be linearized and simplified, shown below for the X forces:

$$\Delta X = \frac{\partial X}{\partial u}\Delta u + \frac{\partial X}{\partial w}\Delta w + \frac{\partial X}{\partial \delta_e}\Delta \delta_e + \frac{\partial X}{\partial \delta_T}\Delta \delta_T \tag{10}$$

The terms $\frac{\partial X}{\partial u}, \frac{\partial X}{\partial w}$ are stability derivatives and $\frac{\partial X}{\partial \delta_e}, \frac{\partial X}{\partial \delta_T}$ are control derivatives. Equations 5-9 can be similarly rearranged and put into the state-space form:

$$\dot{x} = A \cdot x + B \cdot u \tag{11}$$

$$y = C \cdot x + D \cdot u \tag{12}$$

where x is the state vector, A is the system matrix consisting of stability derivatives, B is the control matrix consisting of control derivatives, C is the output matrix (usually an identity matrix), D is the matrix representing the coupling between input and output, u is the input vector, and y is the output vector.

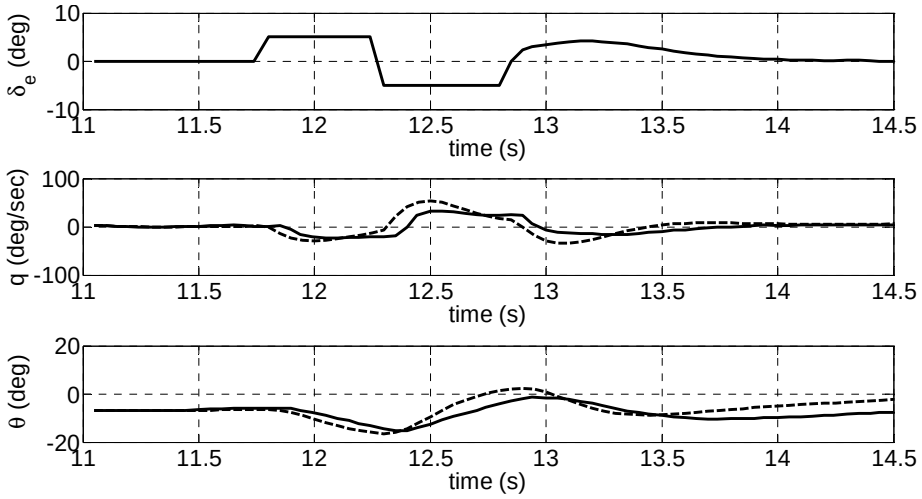


Figure 11. Comparison of the flight data (solid) with the model response for pitch rate and pitch angle

Athena Vortex Lattice (AVL) software was used for the development of the dynamics model of the airplane.[25] The model was validated using the flight data as shown in the Fig. 11 above.

Incorporating an aircraft model also includes consideration of the flight constraints of the particular aircraft such as minimum turn radius, maximum turn rates, and the normalized metric given by Eq. (1), because it is a nonholonomic system and has trajectory limitations. The minimum turn radius and maximum turn rates were determined from averaging the minimum turn radius and maximum turn rates of various flight conditions (level turn, pull-up, and pull-down turns). These parameters were based on the physical characteristics of the SigKadet Senior airplane. The metric is a difficult problem with a model. An initial metric was determined by running the model for a specific time step from an initial condition and running through the allowable command inputs, then taking the average of the simulated distance traveled. This process can be seen in Fig.12 below where the magenta dot is the initial point, the yellow dots are the end points, and the blue lines are the paths. Afterwards, the metric was continuously averaged with the next acceptable iterated step of the RRT simulation.

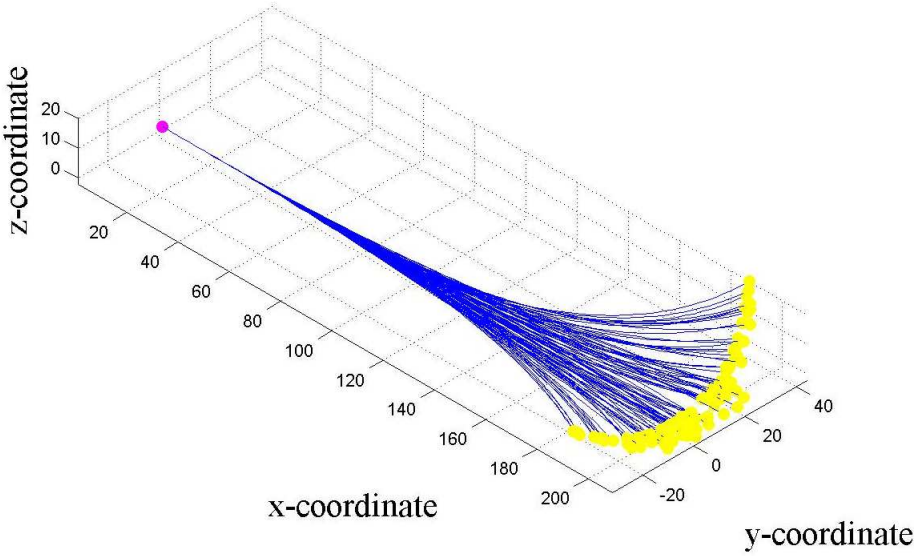


Figure 12. Initial metric determination (coordinates in feet)

The inclusion of a model would add several steps between steps 6 and 7 in the pseudo algorithm presented in Section II. These new steps would be random control inputs into the model in a closed-loop configuration, to attempt to reach ***Xextend***. If the simulated path is considered close enough to ***Xextend***, the algorithm would continue, if not, the algorithm would return to step 3 to find a new random node and continue the process.

Including an aircraft model and moving into 3-D spaces does address some of the issues mentioned previously, however it also presents new issues that is discussed in the following section, such as what is evident in Fig. 12.

V. 3-D Simulation and Results

The 3-D simulation takes place within a 3900x3900x2000 feet space defined by the diagonal vertices (0,0,0) and (3900,3900,2000). The space includes static and dynamic obstacles. The desired path has a start point of (100,100,1000) and an end point of (3800,3800,1000). All the initial conditions of the aircraft model are set at zero (turn rates, Euler angles, etc) aside from an initial velocity of 40 ft/sec and the start location. The model can handle three command inputs, which were directed at the rate of roll, pitch, and yaw rates (angular accelerations) which were all randomly chosen between the constrained ranges from -5 to 5 deg/sec/sec. The simulation uses time steps of 0.05 seconds for model stability, time increment of 5 seconds for each branch simulation, and 3 time increment steps per RRT iteration. The branch limit per RRT iteration was set at 500.

The Predictive RRT algorithm was tested multiple times to find a path from the initial coordinate to the end goal coordinate. Figure 13 below shows an iterative process that approaches close to the goal. These simulations either terminate when any branch reaches within an acceptable distance from the end goal so a path can be found through the tree or if the branch limit is reached. These simulations also include nine simple static obstacles that can be seen as black 'blocks' and ten dynamic obstacles that can be seen as red/black "snakes" because they overlap themselves along their respective trajectories.

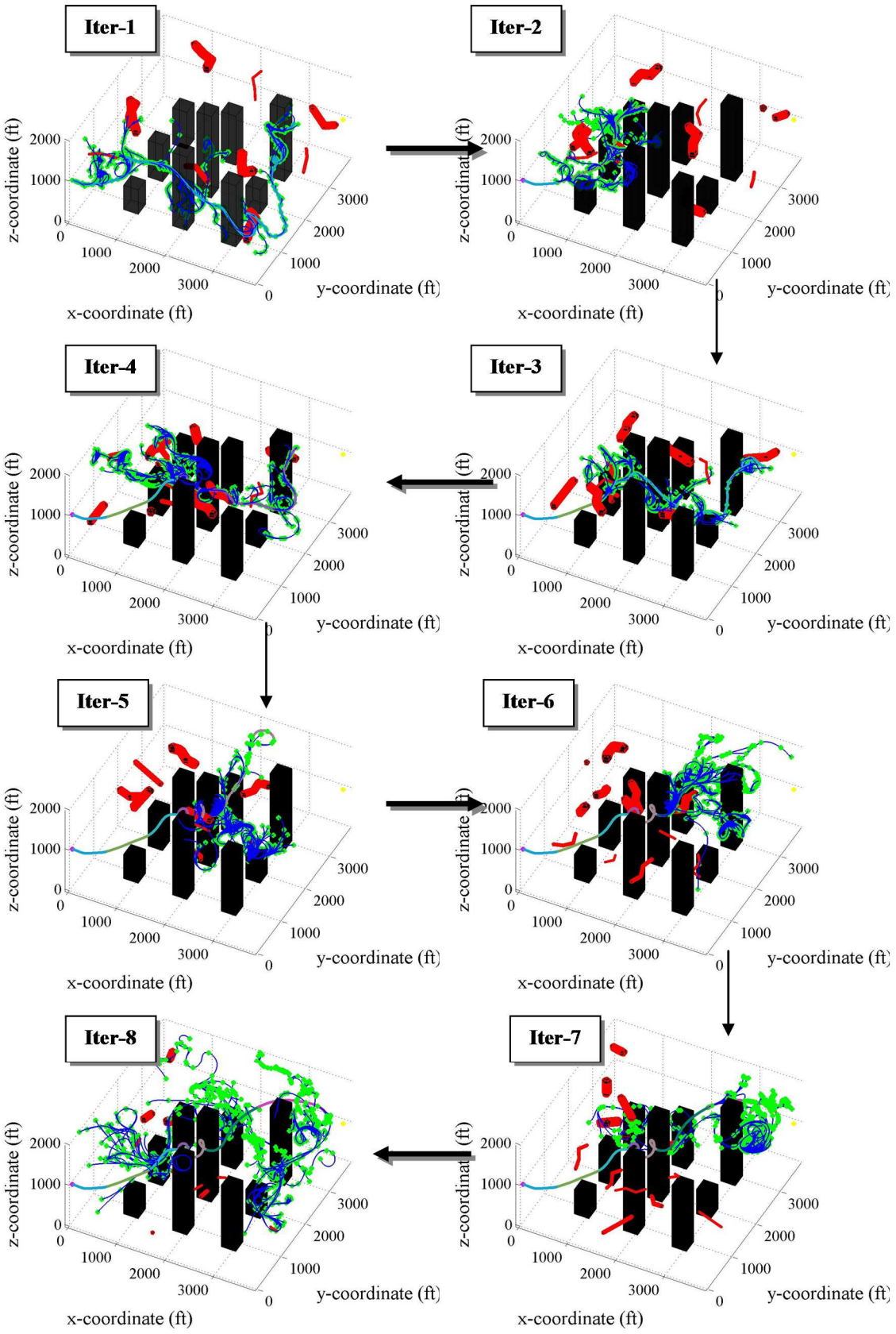


Figure 13.1 3-D Predictive RRT execution

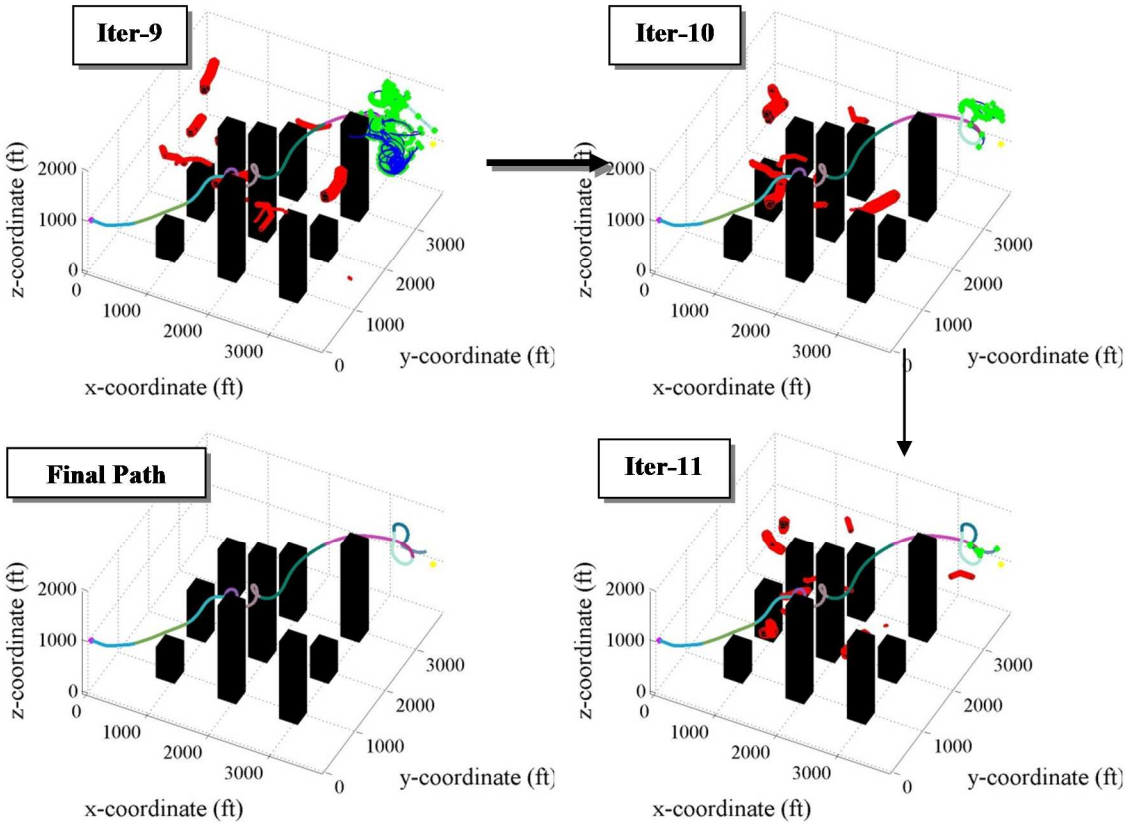


Figure 13.2 3-D Predictive RRT execution and final path

Unlike the branches shown in Section II, the branches in these RRTs do not thoroughly explore the open space. Even if the branch limit was increased, the space exploration would not improve significantly. This is due to the limitation of the aircraft itself. Figure 11 from Section IV shows the limited possible paths with varying inputs, which clearly cannot explore the space as versatile as the previous simulation. Several attempts have been made to improve the space exploration by varying the time increments, widening the control input constraints, and augmenting the state-space models with a throttle control but without significant improvements.

The static obstacles were placed throughout the space to loosely simulate an urban environment. The dynamic obstacles are generated sequentially using the Predictive RRT except without a model for simplicity and to decrease the computing time. The dynamic obstacles start with a randomly generated start and end coordinates, which then go through the Predictive RRT algorithm. These obstacles tend to reach their goal and disappear from the space before the aircraft can reach its own goal coordinate. This was compensated by generating consecutive obstacles to an estimated total run time to ensure that at every time step, the number of obstacles desired were present, which is why more than ten "snakes" can be seen at a time per iteration.

With a model included, the dynamic states of the aircraft can be observed and changes in the flight plan can be adjusted to allow for more favorable conditions or constraints, such as the turn radius and rates. Figures 14 and 15 below show the states and Figure 16 shows the control inputs of the path of the SigKadet Senior UAV while traversing the trajectory shown in Figure 13. The simulated path spans 148.7 seconds.

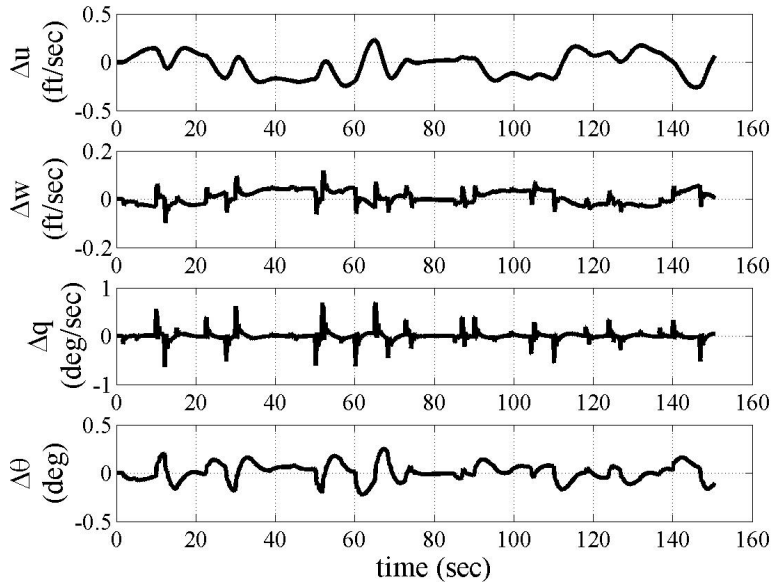


Figure 14. Longitudinal motion of the UAV traversing the 3-D Predictive RRT path

Since each path is simulated for five seconds the system does not have adequate time to completely converge on the command inputs and thus varies throughout the journey from start to finish.

The main disadvantage of these simulations, is again, the limited movement of the aircraft itself. No solutions were found with the simulation parameters and criteria implemented. The result shown is one of the closest the UAV comes to successfully fulfilling the goal reach criteria.

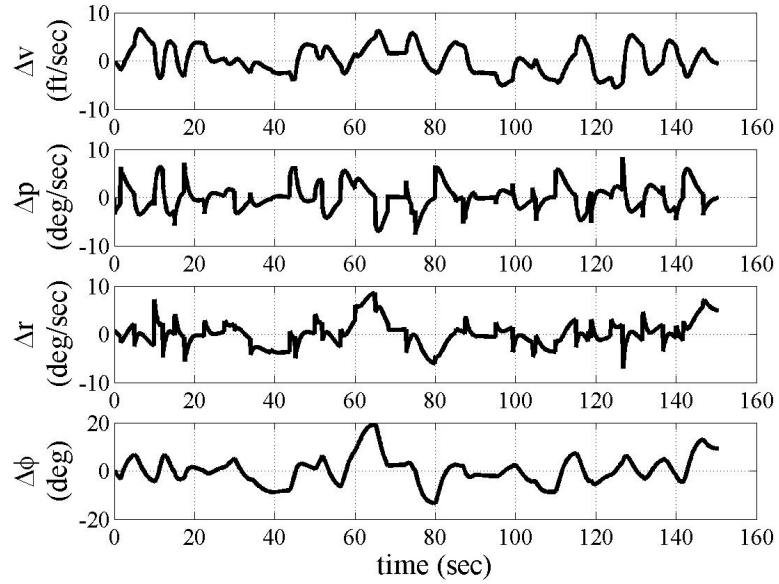


Figure 15. Lateral-directional motion of the UAV traversing 3-D Predictive RRT path

Another issue with these simulation results are also the validity of their application for real-time obstacle avoidance and path planning, but may still be suitable for initial path planning if adjustments can be made to obtain complete solutions. 'Close solution' simulation times range from around 3 to 8 hours. This relatively long computing time is due to the methods used for collision avoidance and the nature of random control inputs to reach a goal that the model could never reach and having to designate new random nodes. Future work will focus on addressing these issues and improving the algorithms.

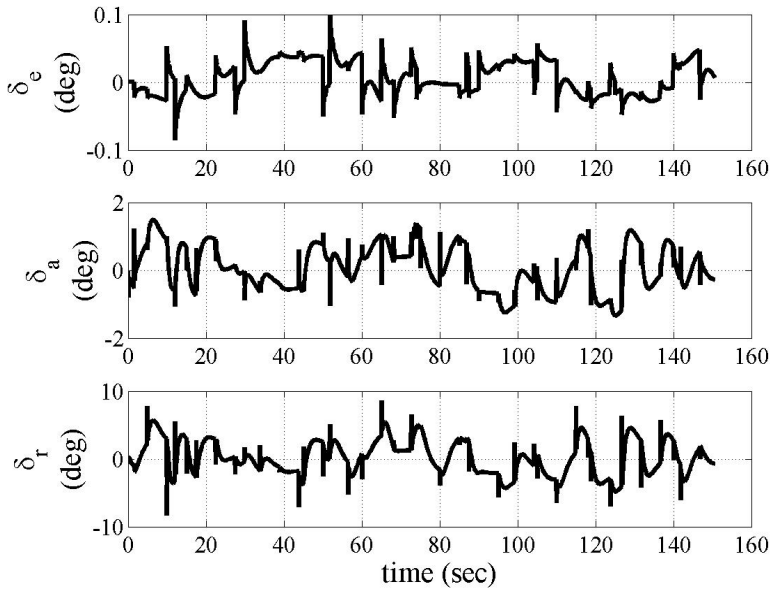


Figure 16. Control inputs of the UAV traversing 3-D Predictive RRT path

To guarantee safety, full scale collision avoidance architecture would rely on accurate and reliable sensor integration, a primary path planning control system, and secondary collision systems such as a reactive and or predictive type algorithm. The partial paths generated by the iterations of the RRT algorithm in some cases resulted in a path that was sometimes bendy and unrealistic for a UAS to use for waypoint navigation to traverse between nodes. This issue could be addressed by implementing smoothing techniques based on the flight control inputs and physical characteristics of the aircraft model.

An effective and efficient UAV collision avoidance system relies on several software and hardware components to be optimized and work cooperatively together. The detection components used vary between different airframes based on size and useable payload available. Depending on the UAV, the system performance will drive optimizations alterations to the collision avoidance algorithm based on desired airspeed, turn radius, and other characteristic of flight dynamics and control of a particular UAV. Figure 17 shows the architecture for the implementation of UAV collision avoidance.

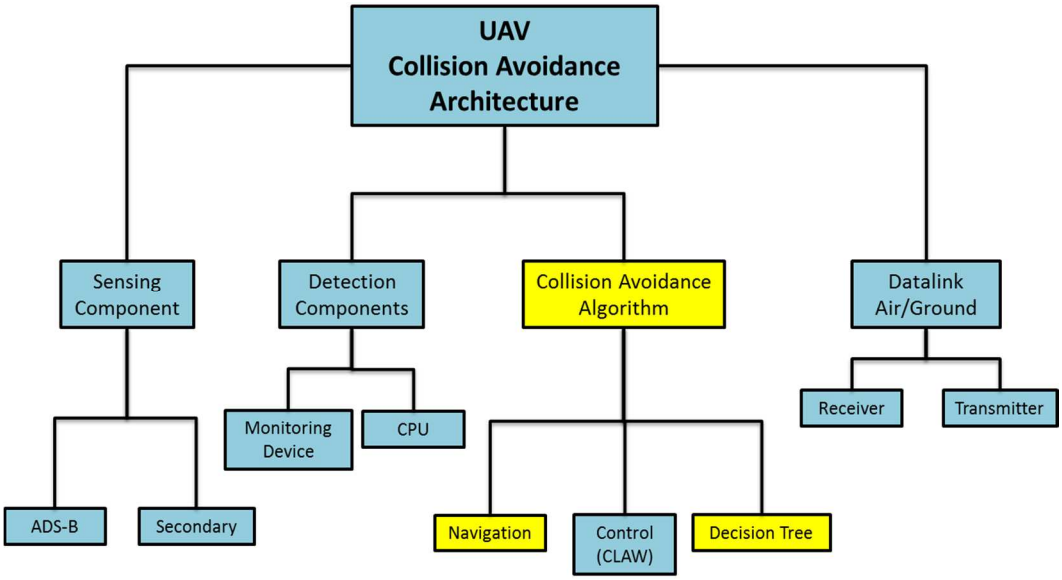


Figure 16. Collision Avoidance Architecture

The error for potential collision states cannot be clearly identified unless real world tests are conducted. Prior to flight testing, the collision avoidance and path planning algorithm will be tested in the software-in-the-loop (SIL) and hardware-in-the-loop (HIL) simulations that will use the FlightGear Flight Simulator as shown in Fig. 18, which shows the aircraft model in a simulated flight.[26] Also, errors of the Predictive RRT execution system could pose a failure of the primary collision system, and it would be ideal to have a secondary reactive collision avoidance system available as a backup.[8]

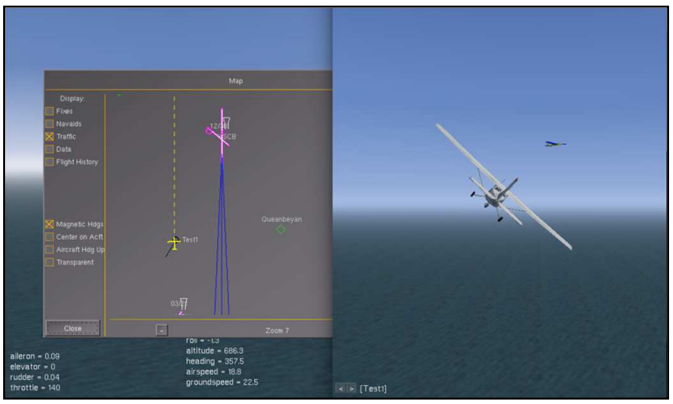


Figure 18. FlightGear model of the Sig Kadet Senior airplane

VI. Conclusion and Future Work

This Predictive RRT algorithm provided an excellent proof of concept for dealing with dynamic obstacles in a 2-D environment but more work is needed for its validity in a 3-D environment. The predictive behavior of dynamic obstacles was implemented with a modeled motion providing initial starting conditions of random position, direction, and velocity in the 2-D environment and Predictive RRT paths of their own in the 3-D environment. In static or controlled environments, path planning techniques are much simpler, whereas high dynamic environments and nonholonomic models present many difficulties such as detection and tracking of dynamic obstacles, prediction of future obstacle position states, collision free path planning to the desired goal waypoint, and model limitations. In the case of UAS's where flight path decisions require quickness and efficiency over the effectiveness of a near optimal solution, utilizing the computational strengths of the Predictive RRT can be effectively utilized in guaranteeing a collision free path for a holonomic system but for a nonholonomic system with path trajectory restrictions, it may be more suitable for initial path planning rather than real-time collision avoidance path planning.

Future work will involve adjustments for the 3-D simulations so that complete solutions can be found with varying criteria and testing the algorithms in SIL and HIL simulations using the custom avionics system that is being developed at Cal Poly Pomona.[12] If found satisfactory in these simulations, the system will be tested in flights. Also, this method will be applied and tested for a slower vehicles such as helicopter and multicopter UAS's.

References

[1]http://www.aviationtoday.com/regions/usa/Worldwide-UAV-Market-to-Top-$80-Billion_66225.html.

[2]http://www.suasnews.com/2010/12/3216/world-renowned-uav-control-expert-presents-gseas-distinguished-guest-lecture/.

[3]Schaeffer, R., "A Standards-Based Approach to Sense-and-Avoid Technology," *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, September, 2004.

[4]Geyer, C., Singh, S., and, Chamerlain, L., "Avoiding Collisions between Aircraft: State of the Art and Requirements for UAVs operating in Civilian Airspace", Tech. Report, CMU-RI-TR-08-03, Robotics Institute, Carnegie Mellon University, March, 2008.

[5]"Sense-and-Avoid Equivalent Level of Safety Definition for Unmanned Aircraft Systems", *NASA Dryden Flight Research Center*, Rev9, 2005.

[6]"NextGen", https://www.faa.gov/nextgen/.

[7]"Air Traffic Services Brief -- Automatic Dependent Surveillance-Broadcast (ADS-B)", 17 January 2013, http://www.aopa.org/Advocacy/Air-Traffic-Services-,-a-,-Technology/Air-Traffic-Services-Brief-Automatic-Dependent-Surveillance-Broadcast-ADS-B.

[8]Redding, J., Amin, J.N., et al., "A Real-Time Obstacle Detection and Path Planning System for Autonomous Small-Scale Helicopters," *AIAA Guidance, Navigation, and Control Conference*, Hilton Head, SC, August 20-23, 2007.

[9]Kang, K., Prasad, J.V.R. and Johnson, E.N., "An Integrated Framework for Adaptive Optimal Obstacle Avoidance for Rotary-Wing UAVs," *American Helicopter Society 68th Annual Forum*, Fort Worth, TX, May 3-5, 2012.

[10]Johnson, E.N., Mooney, J.G., et al., "Flight Testing of Nap-of-the-Earth Unmanned Helicopter Systems," *American Helicopter Society 67th Annual Forum*, Virginia Beach, VA, May 3-5, 2011.

[11]Griffiths, S., Saunders, J., et al., "Obstacle and Terrain Avoidance for MAVs," *IEEE Robotics and Automation Magazine*, September, 2006.

[12]Bhandari, S., Pernalete, N., Bettadapura, A.., Dadian, O., et al., "Avionics System for UAV Flight Controls Research," *Infotech@Aerospace*, Boston, MA, August 19-22, 2013.

[13]Lee, R., and Ippolito, C., "A Perception and Mapping Approach for Plume Detection in Payload Directed Flight," *AIAA Infotech@Aerospace Conference and AIAA Unmanned...Unlimited Conference*, Seattle, Washington, April 6-9, 2009.

[14]LaValle, S.M., "Rapidly-Exploring Random Trees: A New Tool for Path Planning," TR 98-11, Iowa State University, October, 1998.

[15]Kothari, M., Postlethwaite, I., and Gu, D., "Multi-UAV Path Planning in Obstacle Rich Environments Using Rapidly-exploring Random Trees," *Proceedings of 48th IEEE Conference on Decision and Control*, Shanghai, China, December, 2009.

[16]Fulgenzi, C., Tay, C., et al., "Probabilistic Navigation in Dynamic Environment using Rapidly-Exploring Random Trees and Gaussian Process," *IEEE International Conference on Robotics and Intelligence*, Nice, France, September 22-26, 2008.

[17]LaValle, S.M. and Branicky, M.S., "On the Relationship between Classical Grid Search and Probabilistic Roadmaps," *The International Journal of Robotics Research*, Vol. 23, No. 7-8, pp. 673-692, August, 2004.

[18]Caselli, S., Reggiani, M., and Rocchi, R., "Heuristic Methods for Randomized Path Planning in Potential Fields," *Proceedings of IEEE International Symposium on Computational Intelligence and Automation*, 2001.

[19]Kavraki, L., Svestka, P., et al., "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, pp. 566-580, 1996.

[20]Plaku, Plaku, Kavraki, L., and, Vardi, M., "Discrete Search Leading Continuous Exploration for Kinodynamic Motion Planning," *Robotics: Science and Systems*, 2007.

[21]LaValle, S.M., "Planning Algorithms," Cambridge University Press, 2006.

[22]Urmson, C. and Simmons, R., "Approaches for Heuristically Biasing RRT Growth," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[23]A. Muraru, "A Critical Analysis of Sense and Avoid Technologies for Modern UAVs," *Advances in Mechanical Engineering*, ISSN 2160-0619, Vol. 2, No. 1, March 2012.

[24]Roskam, Jan, "Airplane Flight Dynamics and Automatic Flight Controls," Lawrence, Kansas, DAR Corporation, 1998.

[25]Anderson, N., Hagenauer, B., Erickson, R., and Bhandari, S., "Flight Testing of a UAV Airplane for Autonomous Operation using Piccolo II Autopilot," *AIAA Atmospheric Flight Mechanics Conference*, Honolulu, HI, August 18-21, 2008.

[26]Gray, J., Wood, J., Piana, M., Bhandari, S., Tang, D., et al., "Collision Avoidance System for Unmanned Aerial Vehicles," *Proceedings of the AUVSI Unmanned Systems 2015*, Atlanta, GA, 4-7 May 2015.