Applying a Probabilistic, Real-Time Reactive Planner for avoiding hostile fire to both the Apache and a Fixed Wing UAV

Richard Stottler[1] and Cory Barton[2]
Stottler Henke Associates, Inc., San Mateo, CA,94402 and US Army Aviation Applied Technology Directorate (AATD), Fort Eustis, VA, 23604-5577

A previous paper[1] described the design of a highly reactive, real-time planner for aggressive 3D aircraft maneuvering to avoid unguided threats, its design, and the results of applying a prototype to the OH-58D Kiowa Warrior in FlightLab simulation studies. This paper describes the development of the full-scale Probabilistic Road Map (PRM) Path Planner (PP), including its application to two different aircraft, the Apache and an Unmanned Aerial Vehicle (UAV). This included populating the maneuver libraries required by the PRM PP and testing with the two different aircraft models and associated maneuver libraries in FlightLab. Experimental simulation results are presented, including the surprisingly low level of effort required to adapt the PRM PP from a rotary wing manned platform to a fixed wing unmanned one. The PRM PP avoids likely paths from several simultaneous or sequentially fired munitions, avoids terrain and popup obstacles, considers a wide variety of other criteria including breaking or maintaining line of site with several stationary points or moving platforms, and reliably returns within a tenth of a second while finding a solution in all cases tested when one exists. In addition to showing that the techniques are applicable to widely different air platforms, they are also applicable to non-aircraft.

Nomenclature

PRM = Probabilistic Road Map
PP = Path Planner
UAV = Unmanned Aerial Vehicle

I. Motivation

In the course of unmanned and manned aviation operations, emergency situations arise that call for extreme aircraft maneuvers. Hostile enemy fire or impending threat of collision with another aircraft or object may force the choice between executing a maneuver at the maximum dynamic limits of the airframe or potential loss of the system. Emergencies happen suddenly, and a pilot who is fully engaged in completing mission tasks may take a few extra seconds to decide how to react. In this case, there is a clear need for a real-time, short-term Path Planner (PP) that can generate flight paths that provide maximum deviation from current position and reach an end state that is stable and safe.

For a manned platform with fly by wire flight controls and a sophisticated autopilot, a real-time short-term PP could be used to automatically avoid one or multiple threats. A pilot may not be comfortable having the aircraft maneuver automatically, so the PP could be used to provide cues to the pilot to suggest the flight path that would provide maximal threat avoidance in such a way that the pilot could manually follow the path.

It is also likely that future combat scenarios will include an increased incidence of hostile threats to unmanned aircraft, and that these aircraft will be fitted with threat detection systems. An automated real-time PP that can generate aggressive maneuvers is a natural fit for such an unmanned aircraft fitted with a threat detection system,

[1] President, 1670 South Amphlett Blvd., Suite 310, San Mateo, CA, 94402, AIAA Member.
[2] RDMR-AAI, Aviation Applied Technology Directorate, US Army Research, Development & Engineering Command (RDECOM), Bldg 401, Lee Blvd., Fort Eustis, VA 23604-5577, AIAA Contributor.

which will give the aircraft a greater chance of survival. In addition, a real-time aggressive PP would be useful for an unmanned aircraft that is flying in coordination and close proximity with other manned or unmanned aircraft.

As autonomy improves and the attack capabilities of unmanned aircraft are improved, aggressive maneuvering will be an asset to an attacking aircraft, including enabling nap of the earth flight to avoid detection on approach to a target. A real-time PP coupled with a powerful terrain detection system would be capable of sustaining nap of the earth flight near the dynamic limits of the airframe and physically closer to the terrain than a human pilot would be able to.

In an emergency situation, valuable time could be saved by presenting the pilot with a flight path that provides maximum threat avoidance while accounting for vehicle states, aircraft rate limits, external safe airspace constraints, restricted operating zones, terrain types, datalink and line-of-sight limits, actions by threats, potential collisions and obstacles in the flight path, and other mission constraints imposed by pilots or air vehicle controllers. For a pilot who has been under a heavy mission workload, the presentation of a good evasive path suggestion could potentially save lives.

Given the ever increasing variety of unmanned aircraft being fielded, the ability to quickly adapt an aggressive maneuver path planner to different aircraft platforms presents significant benefits. The development costs are allocated across more platforms and the software becomes more tested and mature with each application.

II. Brief Path Planner Description

A. System Overview – High-Level Architecture

The context for the PP system is shown in Fig. 1. The same PP software code is relatively straightforward to apply to different aircraft by swapping in each aircraft's configuration information (size/shape, definition of controls, maneuver library, etc.). One of the specific advantages of our approach is the lack of dependency on assumptions and the resulting ability to be easily applied to many different aircraft types and environments. It is also very straightforward to consider a wide variety of constraints. Any constraint that can be expressed as a cost that can be calculated in a volume or while traversing through a volume can be applied.

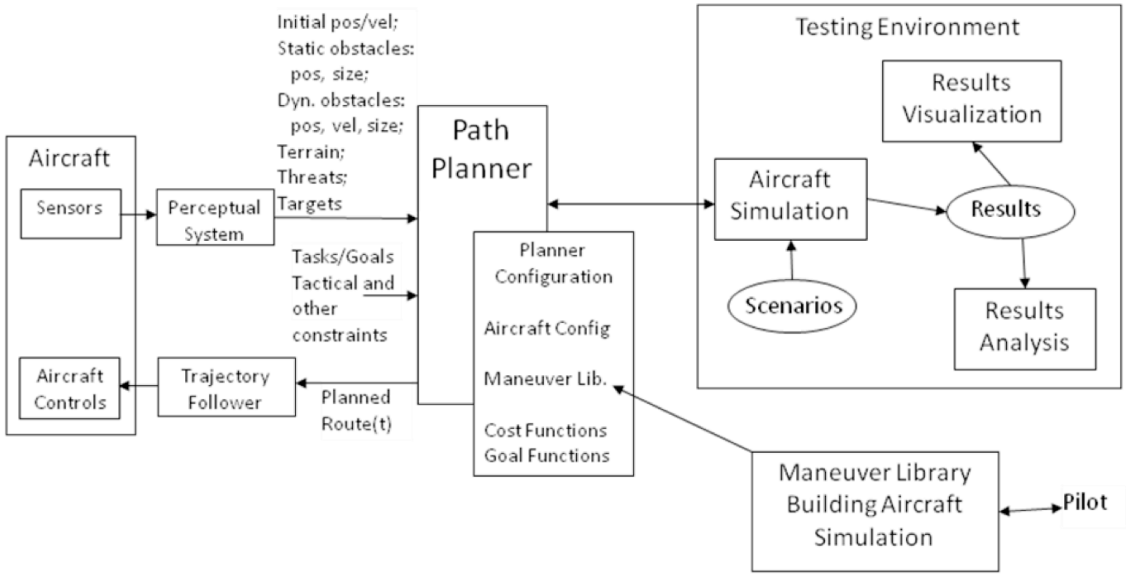

Figure 1. Aircraft Path Planner Context

The PP is interfaced to the Perceptual System (which is not a focus of this effort), which processes sensor data to determine the static and dynamic obstacles, their sizes and positions, and the velocities of dynamic objects. It passes this information to the PP along with the terrain, threat locations and/or bearings, and target locations. The PP takes this information from the Perceptual System and, along with the current aircraft's tasking, mission objectives, tactical constraints such as maintaining line of sight or relative positions to static locations or moving vehicles or aircraft and considering cost and goal functions, plans a route over time that meets the physical constraints of the aircraft, avoids the static and dynamic obstacles, and reaches a goal point most quickly and in an efficient manner. It then passes back a feasible, collision-free planned route to the aircraft flight control system. Note that although the algorithm computes control settings to determine the feasible path, it does not pass these back, nor expect them to be

used by the trajectory follower. The control settings are only used by the PP to ensure that the path is feasible. This decouples the PP from the details of the autopilot.

The PP builds feasible routes as a tree of reachable milestones. Each milestone has 13 dimensions corresponding to the aircraft's translational and angular positions and velocities, and the time at which the aircraft is planned to arrive at that state. Each milestone represents a reachable state. For each path planning episode, the planner builds a new roadmap (tree of reachable milestones) in the collision-free subset of the aircraft's state-time space, where a state typically encodes both the configuration and the velocity of the aircraft. To sample a new milestone, it first randomly selects a milestone and then selects an applicable maneuver from a library of maneuvers that is appropriate for that milestone's state and then uses the stored delta state values for the maneuver to determine the candidate milestone from the previously generated, selected milestone. By construction, the local trajectory thus obtained automatically satisfies the kinodynamic constraints and structural limits of the aircraft since the maneuver was previously simulated and verified in a high-fidelity simulation called FlightLab. If this trajectory does not collide with the obstacles, its endpoint is added to the roadmap as a new milestone. This iterative incremental procedure produces a tree-shaped roadmap rooted at the initial state-time point and oriented along the time axis. It terminates when a milestone can easily reach a goal point (a point in a goal volume) without collision. This goal point may be the actual desired final goal state or a milestone on a previously planned route to the goal point. Or the goal state may be implicitly defined to simply meet a number of constraints such as a safe point at the end of a safe route, after all munitions have passed. Furthermore, the goal point, like all milestones will be scored as to a number of positive and negative issues with various priority weights to arrive at a final desirability sum for the goal point.

B. Probabilistic Road Map (PRM) Planner Details

The architecture for the Probabilistic Road Map (PRM) PP is shown in Fig. 2. The dark arrows represent method calls from one component to another and the normal arrows represent data flow. The heart of the algorithm is the generation of new milestones and the associated checking that a route has been found. The first step in this process is to select an existing milestone with probability inversely proportional to the density of the other milestones near it. This will tend to favor milestones on the edge of unsampled regions, which, in turn, tends to favor sampling milestones in unsampled and under-sampled regions. This typically leads to a uniform sampling throughout the reachable space. Experience has shown that, although one may improve the performance of a PRM planner on some examples by biasing the distribution of milestones, a sampling strategy that yields a uniform distribution of milestones over the reachable free space avoids pathological cases and gives the best results on the average. Also, uniform sampling is required for the proof of the theorem that the algorithm finds a solution if one exists with a very high probability (effectively 100%) for a reasonable space and obstacles.

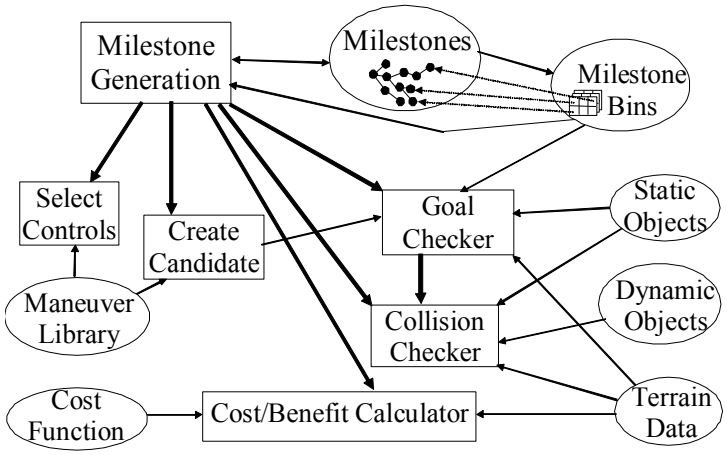

Figure 2. PRM Path Planner Architecture

Prototyping revealed the need to have the maneuver library be created by having real pilots fly the aircraft model and recording their control inputs. To make this process as efficient as possible for the pilots, a maneuver recording facility was added to the FlightLab simulation that stepped the pilots through different initial conditions and requested they fly aggressive maneuvers with different objectives in mind (e.g., maximum downward displacement, maximum right displacement, maximally up and to the left, etc.). They were also requested to fly the maneuvers such that they ended in constant attitude turns. So the typical maneuver would snap the aircraft rapidly to a new attitude and then, using this new attitude, maximally turn the aircraft with constant attitude in a coordinated way for at least a few seconds. The software could determine how constant the attitude was and to what degree any aircraft limits were violated (e.g., exceeded torque limit by 3% for two-tenths of a second). The pilot could then decide whether his maneuver was adequate and save it or redo it. By having constant attitude turns, the maneuvers are effectively parameterized in time so that the PP, by choosing different time periods for the maneuver, effectively chooses how much of a turn will be executed (how many degrees the aircraft's velocity vector will change). The

software also records how the position, velocity and attitude change compared to the initial conditions. This is used to save several different snapshots at different time instants, which give how the position and velocity are changed by the maneuver, given the length of time that the maneuver will be held. Of course since attitude is constant (after the initial change), the change in attitude is fairly independent of time and is simply the difference between the initial attitude and the final attitude. There is also the facility to request the pilot transition from the maneuver to trim and from one maneuver directly to another (e.g., transition from a turn right and up to a turn left and up). By storing these captured maneuvers from pilots for a variety of conditions, the PP has a large library to draw from.

C. Milestone/Path Desirability Calculation

After the candidate milestone is verified to be reachable, the benefit of reaching it must be calculated. Desirability is the weighted sum of how well the milestone meets every mission objective, including survival. If there are munitions in the air, the weight associated with achieving a safe minimum distance from all munitions is very high. But there will be other weights reflecting the relative importance of other objectives depending on the aircraft's mission and role within that mission, such as maintaining communications, maintaining contact with reconnaissance targets, destroying enemy units, maintaining aircraft energy, attacking threats engaging this aircraft, and targeting threats engaging this flight's other aircraft. Any objective or other desirability criteria can be included. The Desirability Function for each milestone can consider the type of maneuver and its parameters and the time consumed as well as its energy use or loss and the desirability of the location (from a tactical perspective, or other factors). This desirability is stored with the milestone. Additionally the total score of each milestone is calculated from the total score from the previous milestone plus the desirability of the new one. Line of sight (LOS) calculations are normally one of the more expensive calculations and we were able to show that even performing many of these at each milestone in was possible within the tenth of a second limitation.

The path planner generates a multiple of parallel path options in the form of a tree of milestones. Milestones that collide with terrain are immediately pruned away so that only feasible non-colliding paths will remain. Then paths are evaluated (so the best one can be chosen) by evaluating each milestone in the path being evaluated using the following formula:

$$Score(p,v) = W_S \cdot min(min_i(d_i(p), T) + W_E \cdot \Delta E(p,v,p_{-1},v_{-1}) + \Sigma_j n_j(p) \cdot w_j \quad (1)$$

p: The position of the aircraft at the milestone
v: The velocity of the aircraft at the milestone
p_{-1}: The position of the aircraft at the previous milestone
v_{-1}: The velocity of the aircraft at the previous milestone
$d_i(p)$: The distance from p to threat I
T: Threshold distance at which we are considered safe and consideration shifts to other priorities
$\Delta E(p,v,p_{-1},v_{-1})$: The change in energy from the state at t-1 to the state at t, as a ratio to kinetic energy
$E(p,v)=E_p(p)+E_k(v)$
$E_p(p)=32.174*altitude(p)$
$E_k(v)= (1/2)v^2$
W_S: Weight associated with survivability
W_E: Weight associated with maintaining energy
$n_j(p)$: Whether objective, j, (normally to maintain or break line of sight) is met at p
w_i: Weight associated with objective, j

III. Recording Maneuvers at Multiple Speeds and Automatically Creating Transition Maneuvers

One issue that arose was that some of the maneuvers, when executed at different speeds from what they were recorded at, would end up in states that did not match any existing maneuver, which was a problem for two reasons. The most obvious one was that the path planner would get stuck if the helicopter arrived in a state that matched none of its existing maneuvers. The second was that this implied that our predictions for the expected state of the helicopter after the maneuver finished were off when using maneuvers at different speeds from when they were recorded. To fix this latter problem we replayed all the maneuvers offline at all the speeds (1-knot increments) so that the resulting change of state could be recorded so that when they were utilized in real time, the software would have better state change estimates. An analysis of the results showed that while the basic maneuvers all appeared to perform well across their required speed ranges, the transition maneuvers (e.g., transitions from a Dive to a Right Turn) all mostly performed poorly except at or very near the speeds for which they were recorded. However, manually creating transition maneuvers for finer knot increments would be too costly so it was decided to attempt to

automatically create the maneuvers using the slice idea (described further below)—but off-line instead of in real-time. Basically, we executed each transition maneuver (at different speeds from the one for which they were created but close enough that they were supposed to be applicable) and when the state deviated significantly from expected, looked for a matching slice from either the same transition maneuver or from a maneuver with the same desired end-point (e.g., for a Dive-Right Turn, anything ending in the Right Turn state).

IV. Slicing/Fixing

Our work was focused on path planning, on generating highly aggressive trajectories, not on following them and specifically not on developing an aggressive auto pilot. However in order to test the plans generated by the system, we needed some mechanism to execute the trajectories in the simulation. Our previous work with the Kiowa Warrior showed that the control inputs stored with the maneuvers could be used in open loop fashion to accurately follow the planned routes for several seconds, and so we therefore attempted to utilize this same concept with the Apache. However, because the Apache is a more maneuverable aircraft, it is also less stable and open loop control led to instabilities and, over time, the helicopter diverged from the planned trajectory. This is undesirable because it means we cannot guarantee that the plans the PP comes up with are collision-free or that they do the beneficial things in terms of avoiding threats or maintaining/breaking LOS with assets, etc.

A solution to this problem was to develop a kind of trajectory follower using the concept of slicing/fixing. These are the act of querying the maneuver database for a "slice" of a maneuver whose beginning is similar to the current state and whose end is similar to the desired state (slicing) and substituting for the currently executing maneuver the sub-maneuver following that slice ("fixing"). To keep the aircraft on the course of the current plan, the system performs the following loop during execution:

1. Get the current state and time from FlightLab.
2. Get the planned state for this time.
3. If the actual and planned states are similar enough, return to 1.
4. Get the planned end state of the currently executing maneuver.
5. Given these two states and the amount of time between the current time and when the maneuver is supposed to end, find a sub-maneuver in the maneuver database that starts with a state similar to the current state and ends with a state similar to the planned end. This is the "fixing maneuver."
6. Interrupt the current maneuver.
7. Queue the fixing maneuver.
8. Return to 1.

V. Adapting to a UAV Model

The decision was made to choose a fixed wing unmanned aerial vehicle (FW UAV) as our second aircraft model and then to roughly base that model on the Predator. The first decision was primarily to show the most generality of our approach, to use an aircraft most different from the first one, the Apache, a manned rotorcraft. The second decision was to use the Predator as a basis for the FW UAV model because the Predator is fairly maneuverable and so, we expected, should show good benefits from our approach, tends to fly lower than some of the alternatives (e.g., Global Hawk), would benefit from the ability to evade enemy unguided hostile fire, and is similar in size to small manned aircraft, which provides additional transition opportunities.

The UAV model was based on researching open sources for relevant Predator data, was built in FlightLab, and was tested to confirm that it exhibited reasonably accurate flight dynamics. After testing was complete, the model was integrated into the current software and graphics user interface that we had previously been using to run the Apache model. The Predator-like UAV was modeled as a rigid body in FLIGHTLAB. The fuselage, wing, horizontal tail, and rudder were modeled as 3-D lifting line bodies for the aerodynamics. A turbo-prop engine was used for propulsion. A four channel control system was built for the UAV with each channel controlling the motion of the plane in longitudinal, lateral, axial, and yaw directions. The roll control was cross-coupled with rudder deflection to counter yaw-roll coupling. This model was trimmed at several test points with varying altitudes, velocities, and gross weights. The general inputs to the model included the pilot stick positions, yaw pedals, and engine throttle. The outputs from the model included the 12 vehicle states. A description of the model is given below.

- Length: 27 ft (8.22 m)
- Wingspan: 48.7 ft (14.8 m)
- Height: 6.9 ft (2.1 m)
- Wing area: 123.3 sq ft (11.5 m²)
- Empty Weight: 1,130 lb (512 kg)
- Loaded weight: 2,250 lb (1,020 kg)
- Max Takeoff Weight: 2,250 lb (1,020 kg)
- Powerplant: 1 × Rotax 914F four-cylinder engine, 115 hp (86 kW)
- Maximum Speed: 135 mph (117 knots, 217 km/h)

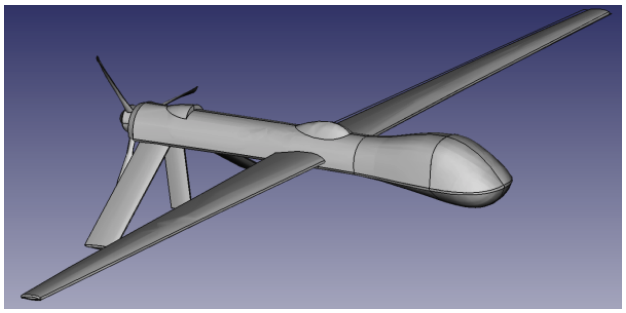

Figure 3. Predator-like UAV

- Cruise Speed: 81–103 mph (70–90 knots, 130–165 km/h)
- Stall Speed: 62 mph (54 knots, 100 km/h) (dependent on aircraft weight)
- Structural Dynamics
  - Plane modeled as rigid body
- Aerodynamics
  - Wing: Modeled using Lifting Line with 3-D Airfoil Table
  - Fuselage: Modeled using Lifting Line with Airloads Table
  - Two Horizontal Stabilators using Lifting Line with 3-D Airfoil Table
  - Rudder using Lifting Line with 3-D Airfoil Table

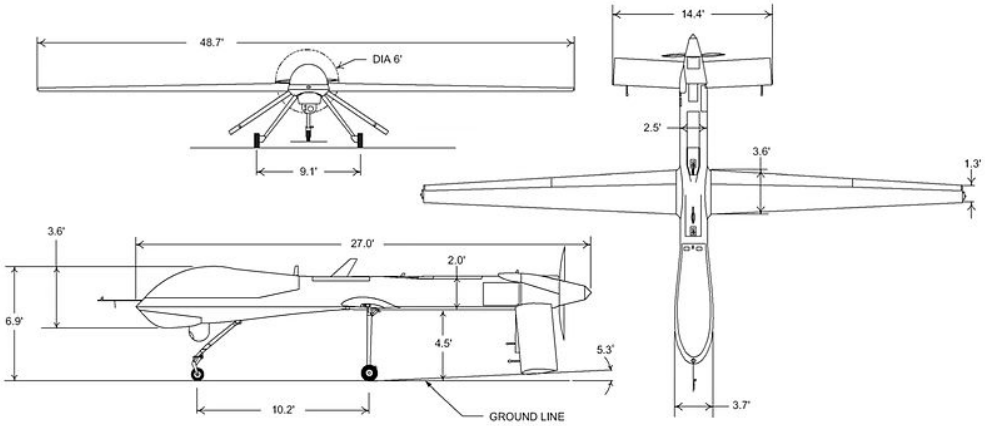

Figure 4. Predator Dimensions

When we received the final, complete "Predator-like" FW UAV model in FlightLab, we integrated it with our simulator, PRM PP and with our maneuver editor. After resolving some minor incompatibilities between the two models (Apache and UAV), we were able to get all parts of all software functioning correctly. We then developed the UAV maneuver library and immediately began gathering test data. The ease of this transition from Apache to FW UAV was truly remarkable. It was really proven that the techniques and the planning software itself were both widely applicable and easily applied to new aircraft.

VI. Scenario Testing Results

Several scenarios were defined to test the ability of the system to plan acceptable routes in different circumstances. The planner and FlightLab simulation were executed for each scenario for each aircraft type (Apache and UAV) at a variety of different speeds. The data captured for each one included many variables: 3-D position vector, 3-D velocity vector, 3-D angular attitude (roll, pitch, and yaw), 3-D angular rates (roll, pitch, and yaw rates), the four control inputs (lateral stick, longitudinal stick, yaw pedals, and collective (for the Apache) or thrust (for the UAV)), Torque, turbine speed, airspeed, sink rate, and Gs, every one-hundredth of a second. Software was written to automatically check that torque and Gs were not exceeding the specifications. During development and debugging, these were useful for detecting problems. However, because planning is based on prerecorded maneuvers, which have been engineered (and tested) to not exceed torque or G limits, unsurprisingly the planned routes do not exceed

them when executed. Automatic software also checked for terrain collision during planning and executing; this was also not found to be a factor. (And interestingly, it was not a factor even during development and debugging, probably because pruning milestones that collide with terrain is the very first step the planner has always performed.)

The main optimizing criteria therefore became maximizing separation, up to a threshold, from hostile munitions or popup terrain (e.g., lately discerned cables). The closest distance each shot in each scenario came to each aircraft was plotted for the various initial speeds that each scenario was run against. One of the important factors for avoiding a munition is how quickly the aircraft can deviate from the trajectory that it was on when the shot was fired. The distances that the aircraft have deviated from their paths after 1 and 2 seconds were plotted for each scenario for each initial speed. This is not as important a number as the closest point of approach of the munition, but it does sometimes supply some supporting insight. Not surprisingly, the Apache, as a higher-performance aircraft, tends to perform better than the UAV against all of these metrics. Shown below is an example plot of closest distance versus starting speed for one scenario. As was typical, the Apache achieved greater separation from the hostile munition.

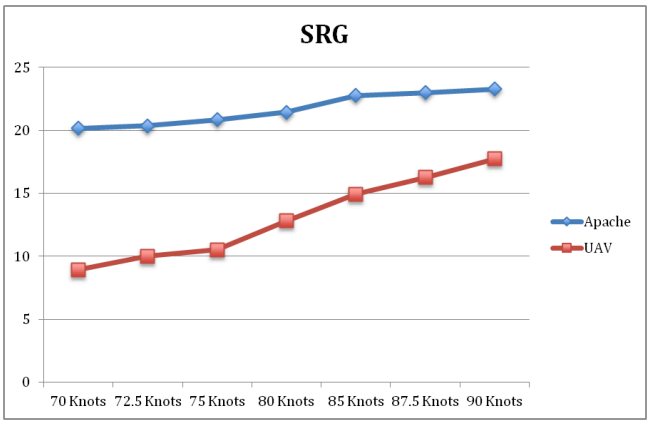

Figure 5. Shot from Right, in Gorge (SRG)

The full set of data (listed above) collected each hundredth of a second is very difficult to comprehend. Even just the three-dimensional aircraft routes can be difficult to understand, especially on two-dimensional paper, but an example of one is shown below. (They are much easier to perceive on a computer screen, where they can be dynamically rotated.) Shown below is one example 3-D route plotted along with the hostile shot.

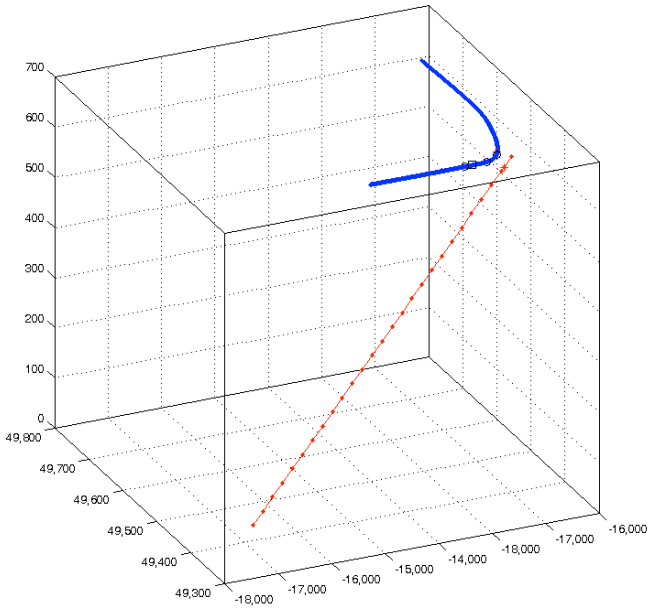

Figure 6. Example SBBG Route, Red Shot Fired, Blue Apache Path, Starting at Left

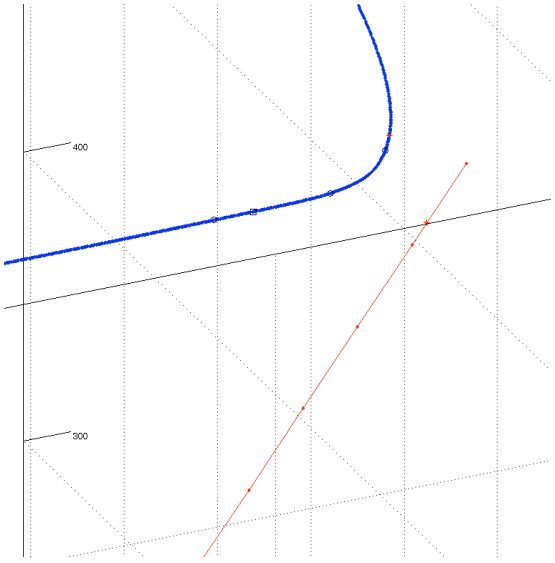

Figure 7. Same Route Zoomed In, Red "*"s Indicate Time of the Closest Point of Approach while Apache

Shown below are some typical plots that illustrate the various testing results and compare the Apache and Predator-like UAV. The first one below, Fig. 8, is the same scenario and therefore corresponds to Fig. 5, above. Figures 5 and 8 are noteworthy because both the Apache and UAV performed well. This was expected because in the SRG scenario, a shot from the right while the helicopter was flying in a gorge, the best maneuver is a quick climb and, since climbs do not require a roll first, this leads to a rapid response. In particular the ability of the Apache to consistently deviate almost ten feet from its initial trajectory in one second is remarkable. The threshold for clearance for the Apache is 20 feet, meaning that once it achieves a 20 foot separation between itself and the munition, it does not try to further increase the distance, so any separation distances above 20 feet are effectively irrelevant. So given this threshold, although in Fig. 8, the UAV appears to perform better at two seconds, this is not really the case. Additionally, at this time the munition has already passed each aircraft.

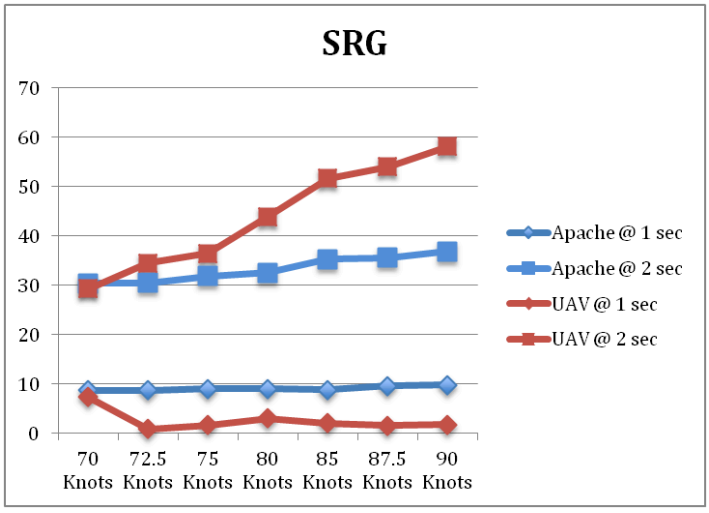

Figure 8. Shot from Right, in Gorge (SRG), 1 and 2 second separations from initial trajectories

The SBBO scenario, Shot from Behind and Below in the Open, provides some contrast. The optimum maneuver is an aggressive turn in this situation. As shown in Fig. 9, the Apache, for the most part achieves acceptable separation at all speeds; the UAV, while performing acceptably in all cases, does better at the higher speeds. Also looking at the Apache's separation after just 1 second shows effectively no movement, owing to the need to use some of the first second to roll.

Figure 9. Shot from Behind and Below in the Open (SBBO), Closest Munition Distance

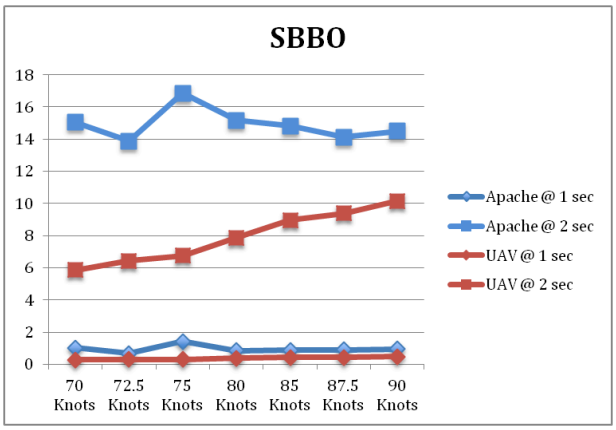

Figure 10. Shot from Behind and Below in the Open (SBBO), 1 and 2 second separations from initial trajectories

In the Shot from 45 degrees in Gorge (S45G) scenario, from just the merits of the geography, either a climb or turn maneuver would seem appropriate. But because of the faster reaction time for a climb, this is the one chosen, as evidenced by how similar Fig. 11 and 12 are to Fig. 5 and 8.

Figure 11. Shot from 45 degrees in Gorge (S45G), Closest Munition Distance

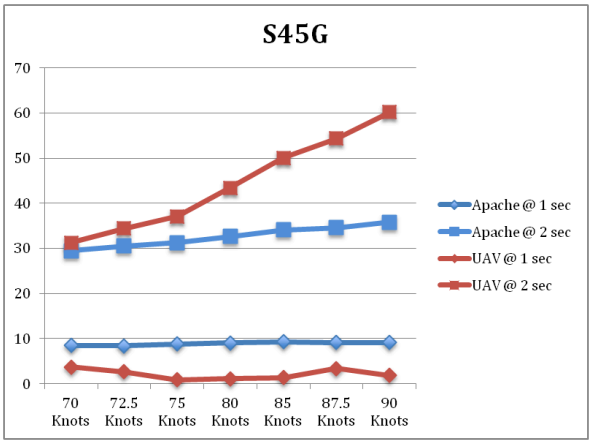

Figure 12. Shot from 45 degrees in Gorge (S45G), 1 and 2 second separations from initial trajectories

Finally, to illustrate that multiple munitions could be handled simultaneously, the results of the Simultaneous shots in Gorge (SSG) scenarios are shown below. Figure 13 is the closest distance of the closest munition. Since the threshold standard is 20 feet, there was no advantage for the Apache to exceed this difference and hence different choices that both lead to exceeding a 20-foot separation are arbitrary. Clearly from Fig. 13 and 14, in two cases a turn was selected as the first maneuver versus a climb which was selected in the other cases.

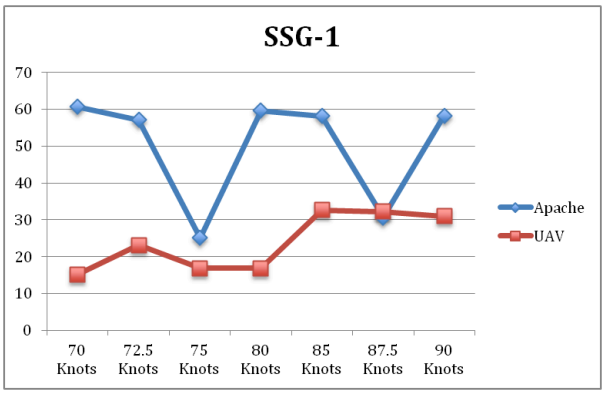

Figure 13. Simultaneous shots in Gorge (SSG), Closest Munition Distance

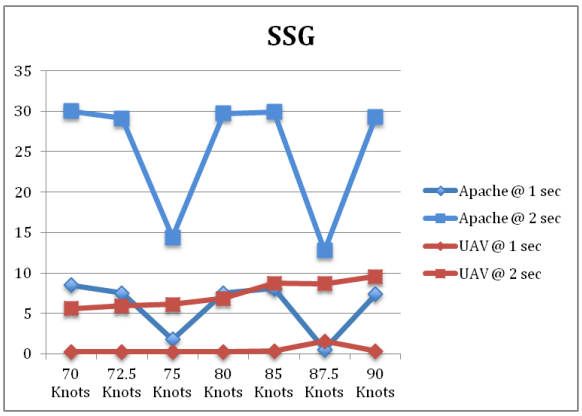

Figure 14. Simultaneous shots in Gorge (SSG), 1 and 2 second separations from initial trajectories

VII. Future Work

There are a wide diversity of follow-on efforts that should be pursued, owing to the generality of the planning techniques employed in this effort. This effort did successfully implement and demonstrate a real-time, high-quality, aggressive maneuver planner for the Apache that is suitable for avoiding hostile fire while avoiding terrain and moving obstacles. However, to be fielded, this trajectory planner would need either to be paired with an autopilot that could follow the trajectory that it produces, or a visualization heads-up display system that instantly showed the pilot the recommended route. In the latter case, the stored control inputs could be used to automatically begin the maneuver, perhaps for the first 0.5 seconds, then the pilot could fly the rest, which could perhaps be displayed as a wire model of a tunnel he should fly through. Once such an autopilot or display system existed, there would be several steps required before this aggressive route planner could be made operational. The first step would be to interface it to a hostile fire indicator system and some means of providing it with accurate terrain and own-position data. The second step would be to more rigorously define under what conditions the path planner would be expected to operate (requirements analysis) in an operational setting and the general and specific issues it should consider beyond avoiding the terrain, popup obstacles, specific munitions, and other moving objects. Then the maneuver library would need to be fleshed out for all applicable conditions (speed and weight ranges) and with less aggressive maneuvers so that these are available to the planner when the most aggressive maneuvers are not necessary, preferably through use of expert Apache pilots, although manual editing of maneuvers could also be used. The combined system would need to be thoroughly simulation tested with a high-fidelity model of the operational equipment, with variations introduced in the model parameters to ensure that the planner was robust across all reasonable operational variants in fielded units. The combined software would need to be flight certified and flight tested in a variety of conditions. It should be noted that control inputs stored with each maneuver might be useful inputs to a closed loop aggressive maneuver autopilot, since they are approximately the correct control inputs required to fly the maneuver.

Versions of the planner also exist for the Kiowa Warrior and a generic UAV based on the Predator. For these more stable platforms, it appears that the control inputs stored with the maneuvers could be used directly for a short-term autopilot, but that would require extensive study in it own right. Otherwise the process of transition for these two platforms would be similar to the Apache with the addition that higher-fidelity models would first need to be acquired than were used so far.

A major finding of the work was how easily the specific planner developed in this effort could be applied to additional aircraft (in this case the fixed wing UAV). Therefore, with a relatively small effort, the software developed under this project could be applied to any aircraft where there was a requirement for aggressive aircraft maneuvering. The proliferation of UAVs and the ease with which they can currently be shot down by a more capable adversary than the insurgents in Iraq and Afghanistan may present many such new requirements. It would be straightforward to add the use of counter measures to the aggressive maneuver planner (e.g., in the case of a guided threat, release flares or chaff while aggressive maneuvering).

The general Route Planner can be applied to planning problems besides just aircraft. In fact, this has already occurred. During this effort, the general planner was applied to the problem of planning multiple simultaneous contiguous routes of circular beam active areas over the surface of a geodesic dome phased array antenna (GDPAA).

We are always looking for additional applications for the PRM planner developed under this effort and fully expect to see it applied. Beyond UAVs and other aircraft, strong possibilities include ground robots and autonomous spacecraft.

VIII. Conclusions

Overall, the highest-level finding was probably the flexibility of PRM concept paired with the maneuver library. This flexibility was important to provide mechanisms for overcoming each of the challenges that we discovered during the course of the project. For example, when we determined that there would not be any expert Apache pilots available, we were able to edit and create maneuvers for the library by hand. When we determined that the recorded dynamics associated with each maneuver in the library were not accurate enough for planning, we could simply, off-line, record the resulting dynamics of running each maneuver at 1 knot increments. When we realized that for testing, there was not an adequate autopilot to follow the planned trajectories and that the Apache was not stable enough to only use the prerecorded control inputs stored with the maneuvers, we found that we could use slices from the maneuvers in the library to effect a kind of real-time control and keep the helicopter stable. When during pre-recording of the results of running each maneuver off-line in 1 knot increments, we discovered that some of the results deviated too far from the nominal attitudes, we could reuse the same slice idea, offline, to "fix" the maneuver in advance. The entire description above effectively consists of solutions based on the flexibility of the combined PRM planner and maneuver library system to overcome unanticipated challenges.

Other findings were more anticipated. One anticipated benefit of the combination was the fact that it effectively precomputed and cached (off-line) a lot computation (i.e., the high-fidelity FlightLab aircraft model simulation to ensure the paths were realizable without violating structural and other dynamic constraints), information, and intelligence, which could be easily and quickly used in real-time by the planner. We did successfully implement and demonstrate a real-time, high-quality, aggressive maneuver planner for the Apache that is suitable for avoiding hostile fire while avoiding terrain and moving obstacles and realizing a wide variety of mission objectives, when possible. Versions of the planner were also created for the Kiowa Warrior and a generic UAV based on the Predator. A major finding was how easily the specific planner developed in this effort could be applied to additional aircraft (in this case the fixed wing UAV). It was very easy to adapt to a radically different platform (fixed wing UAV versus rotary wing manned aircraft). It also proved to be true that the general Route Planner developed as part of this effort could be applied to planning problems besides just aircraft.

A final set of conclusions relate to the specifics of the project. Not surprisingly, for both aircraft, maximum initial trajectory separation was achieved by Dives and Climbs when compared to flat turns. This was expected since a flat turn requires a roll first. An operational system should include maneuvers with various levels of aggressiveness as opposed to only having maximally aggressive maneuvers. Depending on the stability and predictability of the airframe, it may or may not be possible to use the control inputs stored with each maneuver as a sort of short-term, aggressive maneuver autopilot. Even in cases where it is not possible, the maneuvers would likely be useful as part of a real-time control scheme.

References

[1]Stottler, R., Barton, C., and Breeden, D., "Highly Reactive, Real-Time, Planner for Aggressive 3D Aircraft Maneuvers to Avoid Unguided Threats," *Proceedings of the Infotech@Aerospace 2012 Conference*, Vol.3, AIAA, Reston, VA, 2012, pp. 1725-1735.

Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *AIAA Journal of Guidance, Control, and Dynamics* Vol., 25, No. 1, 2000, pp. 116-129.

Kavraki, L. E., Švestka, P., Latombe, J.-C., and Overmars, M. H., "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, 1996, pp. 566-580.

Kindel, R., Hsu, D., Latombe, J.-C., and Rock, S., "Kinodynamic Motion Planning Amidst Moving Obstacles," *Proceedings of ICRA 2000, IEEE International Conference on Robotics and Automation*, Vol. 1, Omnipress, Madison, WI, 2000, pp. 537–543.

Latombe, J.-C., *Robot Motion Planning*, Kluwer Academic, Boston, MA, 1991.

LaValle, S. M., and Kuffner, J. J., "Randomized Kinodynamic Planning," *The International Journal of Robotics Research*, Volume 20, No. 5, 1999, pp. 378-400.