AN EVOLUTION BASED PATH PLANNING ALGORITHM FOR AUTONOMOUS MOTION OF A UAV THROUGH UNCERTAIN ENVIRONMENTS

David Rathbun, Ph.D., Sean Kragelund, Anawat Pongpunwattana, University of Washington, Seattle, WA

Brian Capozzi, Ph.D., Metron Aviation, Inc., Herndon, VA

Abstract

Adaptive and intelligent on-board path planning is a required part of a fully autonomous UAV. In controlled airspace, such a UAV would have to interact with other vehicles moving though its environment. The locations of obstacles (other vehicles) that form obstructions in the environment may only be known with limited accuracy. Evolutionary algorithms (EA) have been successfully used to compute near-optimal paths through obstructed, dynamically changing environments. Explicitly accounting for the uncertainty of the obstacles can result in the survival of "best" paths which differ from those that would be favored in a purely deterministic environment. In this paper, we consider the application of evolution-based path planning to the motion of an unmanned air vehicle (UAV) through a field of obstacles at uncertain locations. We begin with the static form of the EA algorithm for generating a path at a single point in time. We describe the algorithm and show its behavior, specifically how it responds differently based on the known accuracy of the predictions of the environment. We then show how the static structure can be extended to consider the uncertainties which change with time. We demonstrate by application to path planning through a field of moving obstacles whose future motion is uncertain.

Introduction

The use of Unmanned Air Vehicles (UAVs) has been increasing recently. Such aircraft, when operating fully autonomously, are capable of planning their own trajectory for transitioning between desired locations, and often also for avoiding any fixed obstacles in their path. A number of authors have suggested algorithms for autonomous path planning. These include methods such as potential fields [1], graph search methods like A* and D* ([2],[3]), and evolutionary algorithmic (EA) techniques ([4],[5]).

Traditionally, however, UAVs have been restricted to operating in areas that do not contain any other vehicles outside the control of the authority in charge of the UAV. If we desire to move the use of UAVs outside of that restriction and into areas of general use, the UAV must be able to plan a path that not only avoids fixed obstacles, but can effectively deal with moving obstacles (other vehicles) as well.

Changing the planning problem from fixed obstacles to moving ones changes the problem from a geometric one ("don't enter this area") to a dynamic one ("don't enter this area at this time while considering your ability to change your location at that time"). Equally importantly, it also changes it from a deterministic problem to a *stochastic problem.*

To plan a path which avoids a moving obstacle, we need to avoid the location occupied by the obstacle (other vehicle) at some future time. But for most non-trivial cases, the future location of the obstacle (other vehicle) can only be estimated. There will be some degree of uncertainty associated with that motion. A truly effective path planner is one that can deal directly with the combination of the expected motion of the obstacles, the uncertainty in their location, and how that uncertainty changes with time.

Depending on the nature of the specific situation, the degree of uncertainty in the future location of another vehicle can vary. A commercial aircraft following a flight plan (or failing to follow) has a different level of uncertainty than an uncontrolled personal aviation aircraft. For a

0-7803-7367-7/02/$17.00 © 2002 IEEE

military scenario, an intelligent adversarial aircraft would have an even higher level of uncertainty. All of these situations can be thought of as the same general problem, but with different levels of growth of uncertainty.

One approach to handling environmental uncertainty is the Markov localization of Fox ([6],[7]). For evolution-based methods, uncertainty handling was presented in a simple fashion by Latourell [8]. A more complete approach for static obstacles was presented by Rathbun [9].

The uncertainty in an obstacle's location will typically grow as we try to predict further into the future. Because of that growth, there will come a point during actuation (flying) of distant parts of a planned path when the current (e.g. sensed) uncertainty of an obstacle location will be smaller than the uncertainty used in the earlier plan. Therefore, at some point it is appropriate to re-plan. Taking this to the extreme (plan over 2000s, fly first 2s, plan over 2000s, fly 2s, ... etc.), we transform from a *static* planner into a *dynamic* planner.

By considering this growth of uncertainty with time in conjunction with re-planning, we are predicting a motion so that the near term motion will place the vehicle into a location so as to maximize the *likelihood* of success of the far term motion (e.g. maximize the likelihood that the plan can be executed without change). This does not imply that such a planner would take the place of an ultra-fast, rule based, obstacle avoidance safety system (i.e. TCAS). At low levels of integration, the path planner considers a long time horizon to minimize the likelihood that the avoidance safety system is used. At higher levels of integration, the avoidance safety system might condition its conflict resolution based on the "outer" loop path planner downstream recommendations.

For faster updated rates of a dynamic planner, performance (speed of response) may become an issue. Ideally, the problem being solved is a nearby problem to the original planning problem – potentially admitting a nearby solution (e.g. one that does not require a drastic change in UAV trajectory). A desired feature of a path planner is the ability to increase performance by using as much as possible of the previous plan when computing the new plan.

In this paper we present a design for such a dynamic, stochastic path planner based on evolutionary algorithmic (EA) techniques. The EA-based methods have a number of desirable features, making them appropriate for this problem. One is their ability to consider non-linear performance metrics, which will allow for joint consideration of geometric avoidance, time based dynamics, and vehicle performance limitations. Another desirable feature is the EA structure of iterative solution improvements over an internal state. This will allow for the condition at the end of one planning cycle to more easily be used as a good starting condition for the next cycle.

In section 2, we present the structure of the EA-based path planner that we have chosen. In section 3, we show how the planner deals with the uncertainty in the obstacle locations and how it deals with the change in that uncertainty over time. In section 4, we show how the results of a previous planning solution can be incorporated into a re-planning to form a true dynamic planner. In section 5, we present some results of the planner, showing its behavior while flying through a field of moving obstacles whose future motion is uncertain.

Structure of Path Planner

We will begin in this section with a description of the static form (starting location and time does not change) of an evolutionary algorithm (EA) based method for solving the desired path planning problem. Those who desire a more in-depth description of general EA-based path search algorithms or their behaviors should see ([10],[11]).

EA-based path planning algorithms use a population of solutions and random modifications to those solutions to form a structured, stochastic search. See Figure 1. The population is seeded randomly. The algorithm is then run in a loop. At each step (or generation), additional paths (members) are first added to the population by modifying current members (a mutation) or by mixing two or more current members (reproduction). Then the cost function is used to score (the fitness of) all members of the larger population. Based on those scores, the population is reduced (the selection) back to its initial size. Those paths that score well are likely to be retained; those that score poorly are likely to be dropped.

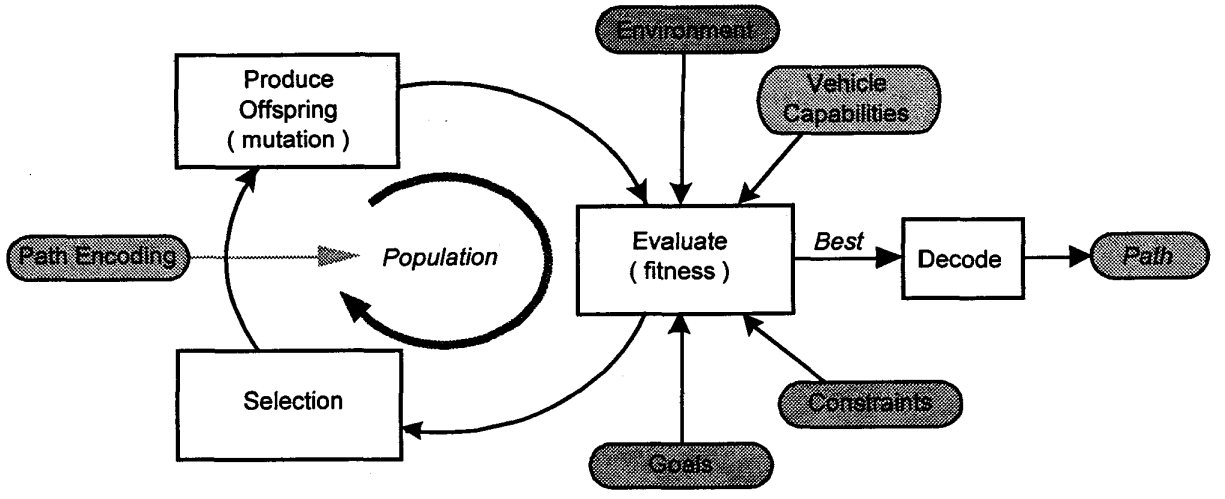

Figure 1. Overview of Evolutionary Search

Our encoding of the paths is a sequence of simple splines (termed *segments*) chained end-to-end. There are two types of splines: straight lines and constant radius curves. See Figure 2. Mutation parameters include: length, radius, and end speed. Speed and turn radius parameters are limited to keep motion within the vehicle capabilities. For line segments, the speed along the segment transitions linearly from start to end speed. We enforce continuity between the joining end and beginning point positions, headings, and speeds.

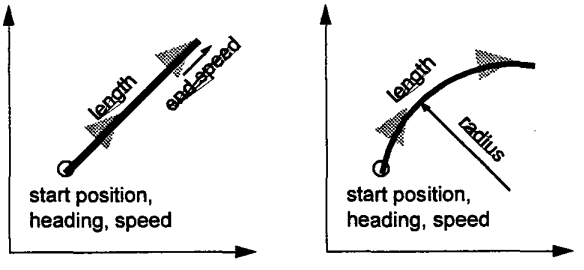

Figure 2. Elemental Path Segment Types

We have four mutation mechanisms:

- **Mutate and Propagate** – randomly changes the parameters of one or more segments, and then re-locates all following segments to enforce the end point constraints. The first segment to be mutated and the number of segments to be mutated are chosen at random. See Figure 3(a). The dashed line is the path before the mutation. The solid line is the path after mutation. The red segments are those that have been mutated.
- **Crossover** – takes the starting segments of one path and the ending segments of another, and matches them up using a *point-to-point-join function.* See Figure 3(b). The two parent paths are the black and blue lines. The resultant path is the solid line. The point-to-point-join function is the two red segments.
- **Go to Goal** – chooses a random segment near the end point and uses the point-to-point-join function to match directly from the start location of that segment to the goal location. See Figure 3(c).
- **Mutate and Match** – changes the parameters of one or more segments, computes the new resultant end point for those segments (similar to Mutate and Propagate), and then connects back to the start of another segment of the path further along. The beginning segment and the segment joined to are both chosen at random. See Figure 3(d). The solid line is the resultant path. The blue segments are the mutated segments. The red segments are the result of the point-to-point-join function.

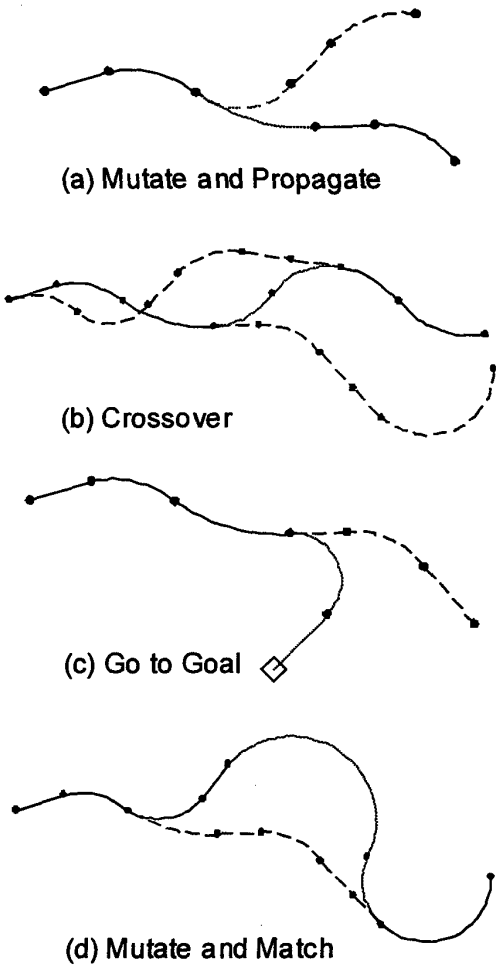

Figure 3. Path Mutation Mechanisms

For each generation, one of the mutation mechanisms chosen at random is applied to each path in the population (the original path is retained in the population). The population size is started at 20 paths, and then mutated up to 40 paths. The broad range of mutation types is included in an effort to give the path search algorithm an improved chance of avoiding local minima.

Several of the mutation mechanisms are enabled by a *point-to-point-join function*. This function takes a starting location (position and heading) and determines the parameters for two segments that will link to an ending location, while meeting the continuity requirements. Without this function, continuity would not be enforced. An illustration of the nature of the geometric solution required to make these joins is shown in Figure 4. There are two possible solutions, depending on whether the curvature of the two arc segments is in the same or opposite directions. For creating a join, both types are calculated and the one with the shortest total length is used.

In the left-most figure, the construction begins with lines emanating from the endpoints of each path segment, perpendicular to the segment direction. At a distance R from the segment end point along these lines would be the join segment center points. The two resultant center points must be a distance $2R$ from each other. Solving this geometrical constraint for R gives the solution. The common endpoint for each join segment is the point half way between each center point.

In the right-most figure, the construction begins with the line connecting the two path segment endpoints, and the two rays tangent to the path segment endpoints. Moving a distance A along one ray and then a distance A along a line parallel to the connection line gives the possible common endpoint to the join segments. Doing the same from the other endpoint with a distance of B must give the same common point. This geometrical constraint can be solved for A and B. The center of the arc is the point of intersection of a line perpendicular to the connection line through the common point, and a line perpendicular to one of the segment endpoints. This solution is derived from Kosugi ([12]).

Note that the solution must be constrained by the minimum turn radius of the vehicle. In the event that either radii for the join is beyond the vehicle capability, the join is considered to have failed and a new mutation (chosen at random) is applied to the parent path.

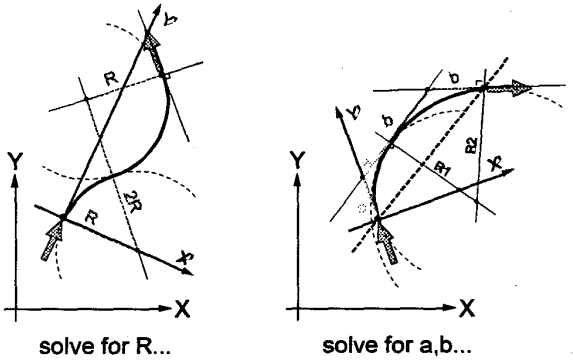

Figure 4. Geometric Constructions Used for Point-to-Point Join Function

The selection process for the larger population is based on a round robin tournament. Random pairs of paths are chosen to compete. The winner is selected based on relative fitness, but includes a random component to occasionally let a lower-performing individual win (e.g. survive). Those paths with the greatest number of wins at the end of the tournament are passed on to the next generation. The process is elitist, in that the path with highest fitness is always passed on to the next generation.

We define the objective of the simple search problem as determination of a path that reaches the goal location, minimizes the probability of intersecting an obstacle, while minimizing the fuel required. This is subject to the constraints that the path is continuous from the start location to the goal location and does not violate minimum or maximum speeds or minimum turn radii. We will present the discussion in a two dimensional (2-D) context. We will mention the three dimensional (3-D) problem only when it requires something beyond a simple extension of the 2-D concepts.

We use a cost function consisting of a linear combination of the separate objectives of interest:

$$J = a_1 G(P_i) + a_2 F(P_i) + a_3 B_1(P_i) \qquad (1)$$

where $G(P)$ represents a measure of the proximity of the path end point to the goal location, $F(P)$ is a measure of the percentage of fuel used, $B_1(P)$ represents the probability that the path intersects an obstacle, and a_1, a_2, and a_3 are relative weighting factors. The EA-based path planner attempts to find a path that minimizes this cost function. The continuity and motion constraints are enforced by the path encoding and mutation mechanisms.

Obstacle Intersection Probability and Timing

To compute the cost function (fitness value) for the EA path search algorithm given above, we need an approximation of the probability that a given path will intersect with one or more obstacles. To get this probability, we will characterized an obstacle (O) in terms of an expected position (C) an expected velocity (v), a safe approach radius (D), an uncertainty specification for the location (σ), and an uncertainty specification for the velocity (τ).

For stationary obstacles, it was shown in [9] that for a path P_i and one obstacle O_j the probability of intersection of the path/obstacle pair can be approximated as a summation over the path length of a *probability density field* (β) that surrounds the obstacle.

$$B_2(P_i, O_j) = \prod_k \beta\left(\left|P_i(s_k) - C_j\right|, \sigma_j\right)\Delta s \qquad (2)$$

where B_2 is the probability of intersection, C_j is the expected location of obstacle O_j, s is the distance along the path, and Δs is the path length between points $P_i(s_k)$ and $P_i(s_{k-1})$. The probability density field β is defined as

$$\beta(r,\sigma) = \frac{p(r+D,\sigma) - p(r-D,\sigma)}{2\pi r} \qquad (3)$$

where $p(r,\sigma)$ is the probability that the actual obstacle location is within a distance r of the expected location C (i.e. the probability distribution).

$$p(r,\sigma) = 2\pi \int_0^r y\rho(y,\sigma)\,dy \qquad (4)$$

where $\rho(y,\sigma)$ is the probability density function for the possible locations of the obstacle. For $r < 0$ we define:

$$p(r,\sigma) = -p\left(|r|,\sigma\right) \quad r < 0 \qquad (5)$$

For an obstacle probability density with a uniform distribution (that used in the simulations), the distribution function used to construct the field as defined in equation (3), is given as,

$$p(r,\sigma) = \begin{cases} r^2/\sigma^2 & r \le \sigma \\ 1 & r > \sigma \end{cases} \qquad (6)$$

One advantage of the approximation to the probability of intersection of equation (2) is the ease with which it can be extended to consider moving obstacles. It is the form of the solution – a summation over a parametrically defined function – that allows for the simple inclusion of time into the equations. Equation (2) simply becomes:

$$B_2(P_i, O_j) = \prod_k \beta\left(\left|P_i(s_k) - C_j(T_k)\right|, \sigma_j(T_k)\right)\Delta s \qquad (7)$$

where T_k is the time at which

$$P(s) = P(s_k) \tag{8}$$

To compute the obstacle location at a given time, $C(T_k)$, without any other indication of its intent, we will use a simple dynamic propagation

$$C(T_k) = C(T_0) + v(T_k - T_0) \tag{9}$$

If some other indication of vehicle intent is known (filed flight plan, expectation of strategy from gaming theory, etc.) then that could be used in place of Equation (9).

There are a number of terms that must be considered for propagation of the location uncertainty, $\sigma(T_k)$. Since the current obstacle location and velocity are *estimated* quantities, there will be an initial uncertainty in the position. The uncertainty in the initial velocity will also couple through the position propagation via equation (9). So, for a vehicle with no action to change its trajectory from a straight line, the minimum uncertainty propagation becomes

$$\sigma(T_k) = \sigma(T_0) + \tau_0(T_k - T_0) \tag{10}$$

Additional terms need to be added to account for action taken by the vehicle (whether intentional or not). A simple model is to assume that the vehicle can apply a fixed acceleration, a, in a random direction. The uncertainty would then grow as

$$\sigma(T_k) = \sigma(T_0) + \tau_0(T_k - T_0) + \frac{a}{2}(T_k - T_0)^2 \tag{11}$$

This may be a good model for a hostile aircraft, whose motion is unpredictable. For a commercial flight following a known trajectory (perhaps badly), a better model might be an uncertainty which grows, but is limited to some upper value. There are many other possible ways to compute future uncertainty. While the structure of the EA-based planner allows for inclusion of nearly any method, we will not provide any further discussion of possible estimation techniques. For all the simulation results presented, the uncertainty propagation model is that of equation (11).

If we can assume that the location of each obstacle is independent of the location of all other obstacles, we can express the total probability of intersection of a path with one or more obstacles as,

$$B_1(P_i) = 1 - \prod_j \{1 - B_2(P_i, O_j)\} \tag{12}$$

Figure 5 through Figure 7 illustrate the behavior of this static planner (captured via snapshots at three different generations) as it considers the motion of the obstacle over time. Here, the UAV (blue outlined vehicle) is heading to the right, trying to reach the goal location (circle, shown in green). The obstacle vehicle is heading vertically up the page. The dashed circles correspond to the planner's estimate of uncertainty at different times – which is seen to grow the further the planner plans (e.g. as the obstacle vehicle moves further up the page). In this case, if the UAV were to travel straight to the goal, it would exactly collide with the expected center of the moving obstacle.

The fitness function for this simulation allows the UAV to use 7.2kg of fuel before incurring any penalty, where the straight line path to the goal would have required 7.9kg. With such a low penalty on fuel usage, the UAV path planner is conservative, choosing a path that has a minimal encroachment onto the obstacle uncertainty region.

Figure 8 and Figure 9 show the time history of the motion after the planning has been completed. The UAV initially heads to its right to come around behind the moving obstacle. It then heads almost straight to the goal, in so doing only entering the uncertainty region of the obstacle by a very small amount.

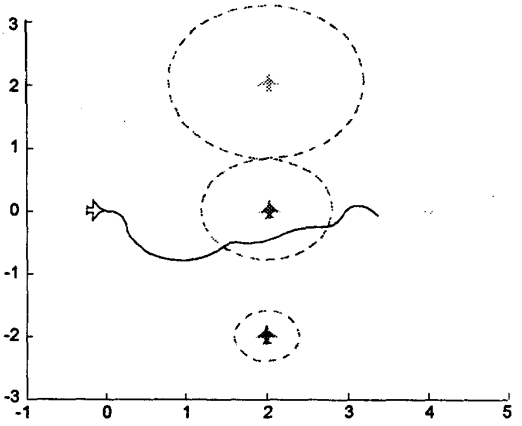

Figure 5. Static Path Planning Showing The Effect Of Moving, Uncertain Obstacles (At Generation 20)

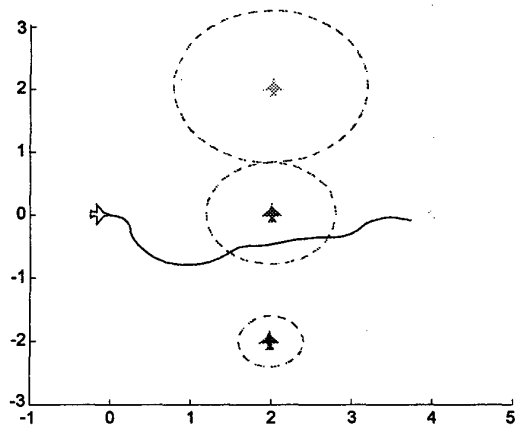

Figure 6. Static Path Planning Showing The Effect Of Moving, Uncertain Obstacles (At Generation 40)

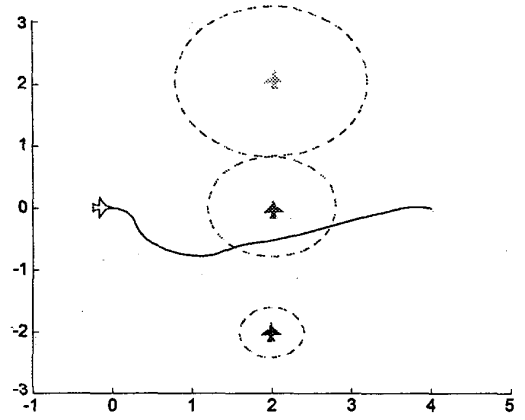

Figure 7. Static Path Planning Showing The Effect Of Moving, Uncertain Obstacles (At Generation 60)

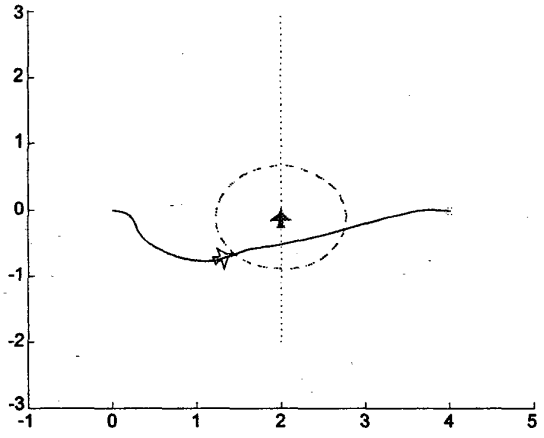

Figure 8. Moving Through Time (Time T1)

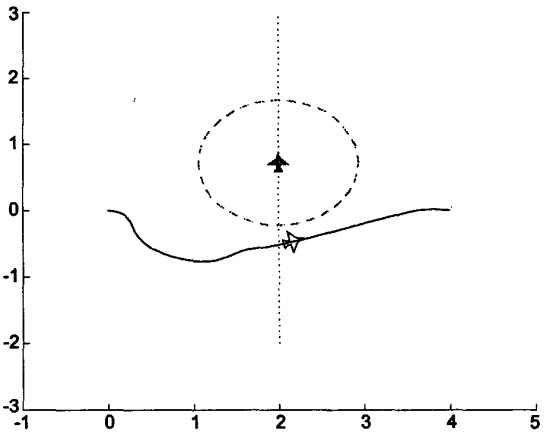

Figure 9. Moving Through Time (Time T2)

Re-Planning

Since the uncertainty in the location of obstacles in the UAV's environment grows over time, after the path has been actuated for a period of time, the measurements of the obstacle locations will diverge from the predicted locations. This calls for re-planning the vehicle path.

Planning over time requires an iterative solution with the following steps:

1. Plan path with "best probability" of success
2. Move along path a given (small?) distance
3. Update estimates of obstacle locations and motion. Update vehicle locations
4. Use the previous solution to seed initial conditions for a new search
5. Re-plan path
6. go to Step (2)

Step (4) is an attempt to save the knowledge gained from the previous planning cycle for the next search. If done well, it has the possibility of improving the performance for some types of the search algorithms. This is of particular importance for EA-based planners, since they must converge to an "acceptable" (e.g. flyable and collision-free) solution in a finite time.

For the EA-based path planner, step (4) will be a mapping of the population from the previous cycle into a valid population for the next cycle:

$$\Gamma_i \rightarrow \Gamma_{i+1} \tag{13}$$

The objective of the mapping is to create a population of paths that meet the constraints of the search algorithm; (a) all start from the same location (position/heading), and (b) are made from segments that meet the vehicle performance constraints – while retaining as much as possible of the characteristics of the original population; (c) a high level of fitness, and (d) wide "variability". We will do so with the following steps (see Figure 10):

1. Choose the path from the population of iteration i with the best fitness value. This is depicted as the black path in the left-most figure.
2. Remove the first segment from that path $S[1]_i$. This is the segment which the vehicle will travel along while iteration $i+1$ is being computed. This is depicted as the magenta segment at the start of the black path in both figures.
3. The new start point for the population of iteration $i+1$ is the end point of segment $S[1]_i$.
4. For all other members of the population
   - Remove a number of segments from the start of the path (equal to the number of segments required to form a join plus one). These are depicted as the dashed parts of the green and blue paths in the left-most figure.
   - Add to the path a join between the end of $S[1]_i$ and the start of the (now shorter) path. These are depicted as the red segments in the right most figure.

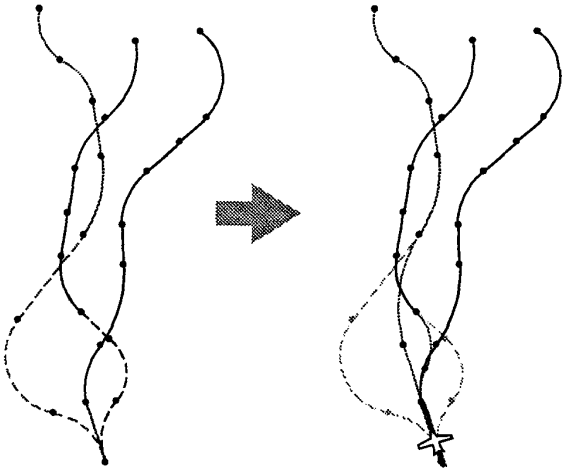

Figure 10. Reset Of Path To New Search Criteria

Note that the plan update rate must be on the order of that required to fly a given trajectory segment. This implies a tradeoff between the numbers of segments describing a trajectory (and thus freedom and flexibility of possible motions) and computational time required to plan the remaining trajectory.

Example EA Simulation

In this section we present the results of simulations which demonstrate the behavior of the EA-based path planner defined in this paper.

We initialize the simulation with a single UAV heading towards its final goal location at a distance of 4km. There are two other aircraft in the area, each considered as an obstacle for the UAV. One obstacle is initially heading roughly parallel to the UAV, but later turns to cross over the UAVs straight line path to the goal. A second obstacle is heading crosswise to the UAVs path, and later shifts its path to directly follow the UAVs straight line path to the goal. The UAV is restricted to avoid the other aircraft by a distance of 0.3km.

The EA-based path planner is initialized with a set of random paths. It is given an initial 20 generations of evolution before time begins. Thereafter, it is allowed 20 generations of evolution each iteration before it must provide a segment for the UAV to fly.

The measurement uncertainty for the location of the obstacle aircraft is set to 0.1km. The

measurement uncertainty for the velocity is set to 5km/s. It is assumed that no knowledge about the intent of the obstacle is available, and the uncertainty growth is set to 0.001 km/s/s.

The UAV is restricted to speeds between 21m/s and 34m/s. The turn radius is restricted to be above 0.18km. The UAV has enough initial fuel to travel 1.52 times the initial distance to the goal location and is not penalized for using the first 91% of that fuel.

Figure 11 through Figure 17 show the results of the simulation at various times (time *Tn* represents the time at the beginning of the *n*'th planning cycle). They depict the current situation, as well as line representing the possible future motion. The UAV is depicted as the blue outlined vehicle. The path that it will fly as the planning process takes place is depicted as the thick, solid blue line. The path that the UAV has already flown is depicted as a thin, solid blue line. The dashed, blue line shows the path of the population with the highest fitness from the previous iteration. The goal location is depicted as a green circle.

The obstacles are depicted as solid, red vehicles. They are following the dashed, red lines. These paths are unknown to the UAV path planner. The dotted, red circles depict the estimate given to the path planner of the vehicle location at the time that the UAV will reach the end of the solid blue line (the end of the planning cycle). The radius of the circle includes the uncertainty at that time, as well as the radius of the required separation. The center of the circle is derived from the current location and velocity of the obstacle with random noise added to simulate estimation error. The dotted, red lines represent the increase in uncertainty of the obstacle locations for times past the end of the planning cycle.

Figure 12 also displays a typical challenge for a path planner that allows the uncertainty of obstacle locations to grow with time: the level of that uncertainty must be weighed against other costs. The uncertainty can grow very large, and completely cover all paths to the goal location. Because of this, a simple solution of avoiding all possible locations of the obstacle results in a problem with no solution. The best solutions are those that can trade off a small probability of obstacle intersection against other performance constraints.

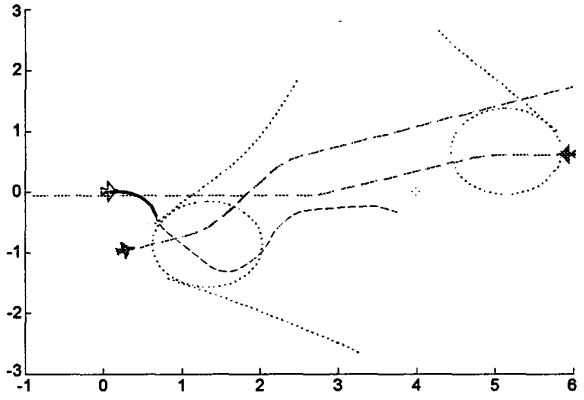

Figure 11. Simulation State At Time T3

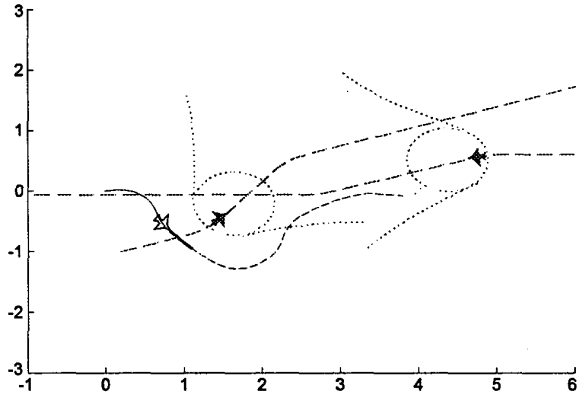

Figure 12. Simulation State At Time T5

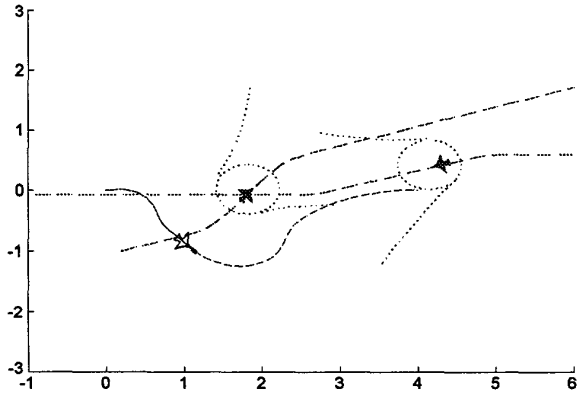

Figure 13. Simulation State at Time T7

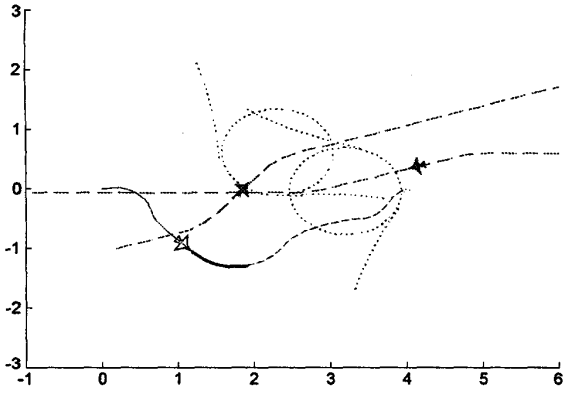

Figure 14. Simulation State At Time T9

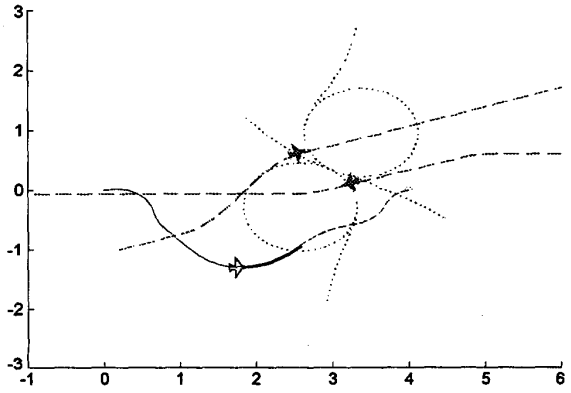

Figure 15. Simulation State At Time T10

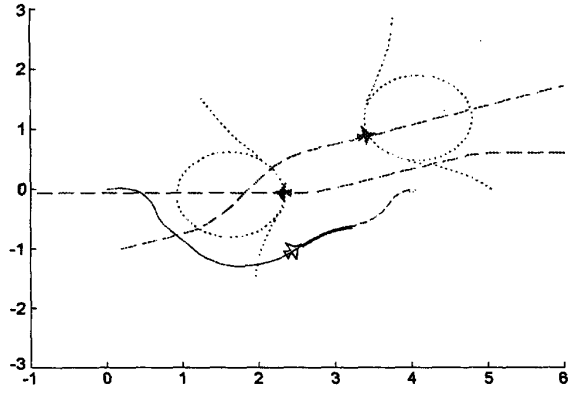

Figure 16. Simulation State At Time T11

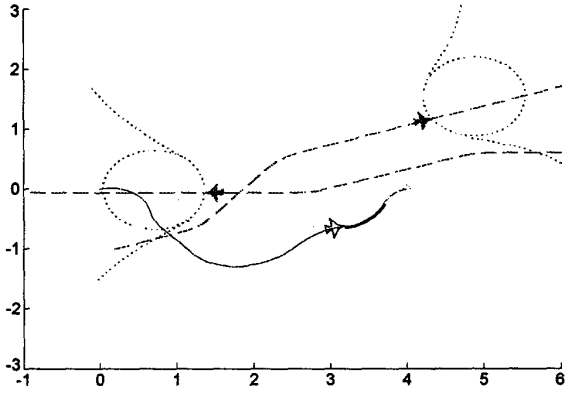

Figure 17. Simulation State At Time T12

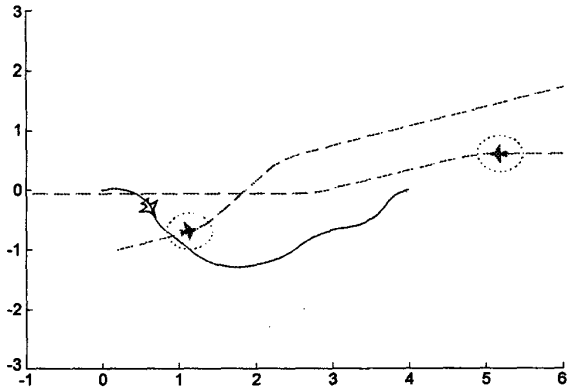

Figure 18. Moving Through Time (T4)

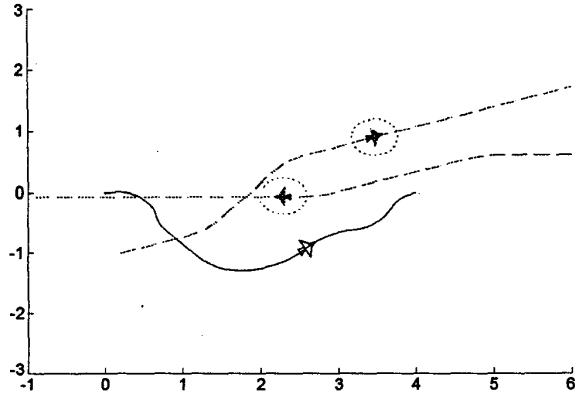

Figure 19. Moving Through Time (T14)

Figure 18 and Figure 19 show the final path flown by the UAV (without depiction of any future uncertainty of the vehicles) at two different times. This is what would have been seen by an outside observer. The dotted, red circles around the obstacle vehicles represent the size of the restricted approach distance. As can be seen, the UAV successfully avoids the two obstacle UAVs without any knowledge of their future intent, while maintaining all vehicle speed and fuel use constraints.

Summary and Conclusions

We have presented a simple framework for path planning using evolution based algorithms, which allows for consideration of uncertainty in estimates of the parameters describing obstacles in the environment, as well as changes in those parameters over time.

We have presented a method for construction of a dynamic planner from the static version through the addition of a state update function.

The structure of the planner was such that the path generated was continuous in space and within the speed and maneuverability constraints of the UAV.

We demonstrated the behavior of the path planning algorithm by application to scenarios where the UAV was required to modify its path to avoid a number of other aircraft flying in its vicinity. The UAV was able to successfully avoid all other aircraft without the need for aggressive avoidance maneuvers or any previous knowledge of the future trajectories of the other aircraft. The planner was able to effectively balance the likelihood of collision against the fuel required to implement overly cautious avoidance behavior.

Acknowledgments

This research for this paper was completed under funding from the DARPA Mixed Initiative Control of Automa-Teams (MICA) project. The authors would like to thank both DARPA and our contracting agency, SPAWAR Systems Center San Diego, CA, (contract N66001-01-C-0089) for their support that made this work possible.

References

[1] Khatib, O., 1986, "Real-Time Obstacle Avoidance for Manipulator and Mobile Robots, International Journal of Robotics Research, Vol. 5, No. 1, pp. 90-98.

[2] Mitchell, J.S.B., and D.M. Keirsey, 1984, "Planning Strategic Paths through Variable Terrain Data", Proc. of the SPIE Conference on Applications of Artificial Intelligence, Vol. 485, Arlington, VA, 1984, pp. 172-179.

[3] Stentz, A., 1994, "Optimal and Efficient Path Planning for Partially-Known Environments", Proc. of the 1994 International Conference on Robotics and Automation, Vol. 4, Los Alamitos, CA., pp. 3310-3317.

[4] Fogel, D.B., and L.J. Fogel, 1990, "Optimal Routing of Multiple Autonomous Underwater Vehicles through Evolutionary Programming, Proc. of the 1990 Symposium on Autonomous Underwater Vehicle Technology, Washington, D.C., pp. 44-47.

[5] Capozzi, B.J., and J. Vagners, 2001, "Evolving (Semi)-Autonomous Vehicles", Proc. of the 2001 AIAA Guidance, Navigation and Control Conference, Montreal, Canada.

[6] Thrun, S., D. Fox, and W. Burgard, S. Thrun and D. Fox and W. Burgard}, 1998, "A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots", Machine Learning and Autonomous Robots, Vol. 31.

[7] Fox, D., W. Burgard, and S. Thrun, 1999, "Markov Localization for Mobile Robots in Dynamic Environments", Journal of Artificial Intelligence Research, Vol. 11.

[8] Latourell, J., B. Wallet, and B. Copeland, 1998, "Genetic Algorithm to Solve Constrained Routing Problem with Applications for Cruise Missile Routing ", SPIE Proceedings, Applications and Science of Computational Intelligence, Vol. 3390, pp. 490-500.

[9] Rathbun, D., and B.J. Capozzi, 2002, "Evolutionary Approaches to Path Planning through Uncertain Environments", Proc. of AIAA's 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles, Systems, Technologies, and Operations, Portsmouth, VA.

[10] Xiao, J. et al., 1997, "Adaptive Evolutionary Planner/Navigator for Mobile Robots," IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 18-28.

[11] Capozzi, B.J., 2001, "Evolution-Based Path Planning and Management for Autonomous Vehicles", Ph.D. Thesis, University of Washington.

[12] Kosugi, M. and T. Teranishi, 1977, "Construction of a Curve Segment with Two Circular Arcs", Transactions of the IECE of Japan, section E, vol. E60, No 11, p. 684