Spline Based Path Planning For Unmanned Air Vehicles

Article · August 2001

DOI: 10.2514/6.2001-4238

2 authors, including:

Tim McLain
Brigham Young University - Provo Main Campus
148 PUBLICATIONS **6,621** CITATIONS

SEE PROFILE

AIAA-2001-4238

SPLINE BASED PATH PLANNING FOR UNMANNED AIR VEHICLES

Kevin B. Judd Timothy W. McLain
Department of Mechanical Engineering
Brigham Young University
Provo, UT 84602

ABSTRACT

A trajectory planning scheme that generates feasible flight routes for an unmanned air vehicle (UAV) is developed. A preliminary path is generated from a Voronoi diagram based on threat locations. This path consists of a series of straight-line segments that cannot be followed exactly by the UAV. Using a series of cubic splines to connect these straight-line segments, this path is refined into an optimum path that is flyable by the UAV. Utilizing a decomposition strategy, both the full path (coarse detail) to the target and the proximate optimum path (fine detail) near the UAV can be quickly computed. The remainder of the optimal proximate path is computed incrementally as the UAV flies the previously computed portion of the path. By decomposing the problem into many smaller problems, the computational burden for calculating optimal paths approaches the near-real-time capabilities desired for the planner. The decomposition approach also allows pop-up threats that occur along the path to be handled efficiently. Simulation results are presented that illustrate the strengths of the approach.

1 INTRODUCTION

One of the fundamental issues in UAV development is trajectory planning. A standard trajectory planning problem is to determine a set of waypoints that takes the UAV from its current location to a target location. For a trajectory planning scheme to be considered viable, it must possess certain characteristics. For military applications, one of the most basic of these characteristics is threat avoidance. The path planning algorithm must calculate a path that minimizes the UAV's exposure to threat sites. Additionally, the dynamic constraints of the UAV must be taken into account, so that the planning algorithm only commands paths that are flyable by the UAV. Examples of such constraints include minimum turning radius, and the minimum and maximum speed of the UAV. It is also desirable that the algorithm take into account the heading that the target should be attacked from, as well as the current UAV heading, so that including these requirements will not exceed dynamic constraints. Computational efficiency is essential in a path planning algorithm as it is desired that the planner be resident on the UAV itself. This gives each UAV additional autonomy as it no longer depends on outside sources for its path information. Another consideration is the possibility of pop-up threats that may be discovered during the mission. It is important for the UAV to quickly determine a new path to avoid these pop-up threats. For these two reasons, it is important for the path planning algorithm to have conservative computational needs.

While the subject of path planning has been widely studied, especially in the area of robotics, research in UAV path planning has received less attention. Goldman [5] gives a brief overview of UAV path planning issues as well as possible solution strategies. In [2], the path planning problem is likened to a dynamic system containing masses and springs. This system is placed in a field of potential functions, which is described by the threat locations, and allowed to come to a steady-state solution. The steady-state configuration describes the path the UAV should follow. Chandler, et al. [3] describes an approach in which fillets are used to create a flyable path from a series of straight line segments. In [6] a path planning scheme is described that uses the analogy of a chain to create UAV paths. Finally, [8] describes a genetic algorithm approach to solving the path planning problem.

Figure 1 shows the situation which was used to

motivate the development of the path planner. The diamond depicts the current UAV position, while the star indicates the target location. The UAV has an initial heading of zero degrees and it is desired that the attack heading at the target be zero degrees.

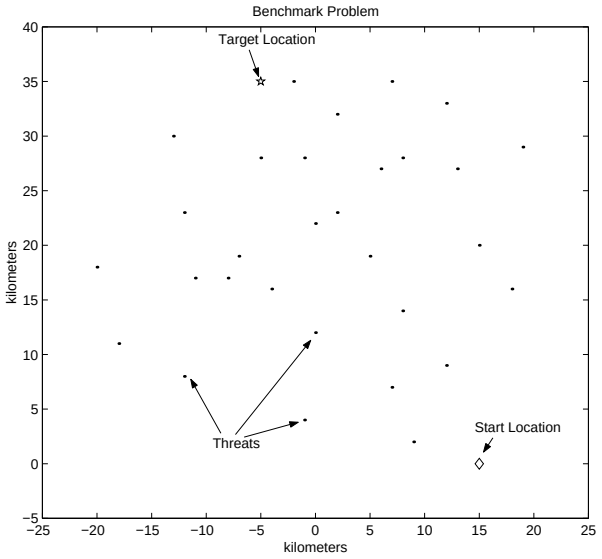

Figure 1: Mission Scenario

2 SOLUTION APPROACH

The path planning scheme developed in this work is carried out in two stages. In the first stage, a coarse path from the current UAV position to the target is constructed based on the locations of the threats. These threat locations are used to construct a Voronoi diagram [4]. This Voronoi diagram is searched to find the best path from the UAV location to the target location. While providing a good path to the target, the path is not feasible due to the sharp corners where consecutive path segments are joined. These corners are not flyable due to turning radius constraints on the UAV. The second stage uses the information from the first stage to build a path that consists of cubic splines which form an optimum path that the UAV can fly. This optimum path is built incrementally, so that detailed flight information is available quickly for the path near the UAV. Detailed information for the path farther away from the UAV is calculated as the UAV flies the initial portions of the optimal path. Both of these stages will be discussed in detail below.

2.1 Initial Path Construction

One of the most important objectives in generating a path to the target is the avoidance of threats. An efficient way to find threat avoiding paths is to use the Voronoi diagram, which is shown in Figure 2 for the problem under consideration.

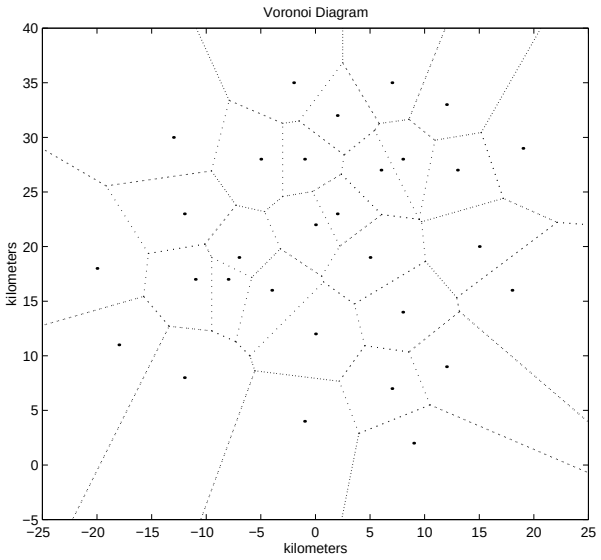

Figure 2: Voronoi Diagram

The Voronoi diagram is constructed using only threat location information. Therefore, the start location and end location need to be connected to the graph. We simply connect the start location and the end location to the nearest three nodes of the Voronoi diagram. This augmented Voronoi diagram now consists of many paths from the current UAV location to the target location. The next step is to determine which path to take. Each edge of the Voronoi diagram is assigned two costs: threat costs and fuel costs. Threat costs are based on a UAV's exposure to enemy radar. Assuming that the UAV radar signature is uniform in all directions and is proportional to $1/d^4$ (where d is the distance from the UAV to the threat), the threat cost for traveling along an edge is inversely proportional to the distance (to the threat) to the fourth power. An exact threat cost calculation would involve the integration of the cost along each edge. A faster approach is to calculate the threat cost at several locations along an edge and take the length of the edge into account. In this work, the threat cost was calculated at three points along each edge: $L_i/6$, $L_i/2$, and $5L_i/6$, where L_i is the length of edge i. The threat

cost associated with the ith edge is given by the expression:

$$J_{threat,i} = L_i \sum_{j=1}^{N} \left(\frac{1}{d_{1/6,i,j}^4} + \frac{1}{d_{1/2,i,j}^4} + \frac{1}{d_{5/6,i,j}^4} \right),$$

where N is the total number of threats, $d_{1/2,i,j}$ is the distance from the $1/2$ point on the ith edge to the jth threat. This scheme of calculating threat cost is shown in Figure 3.

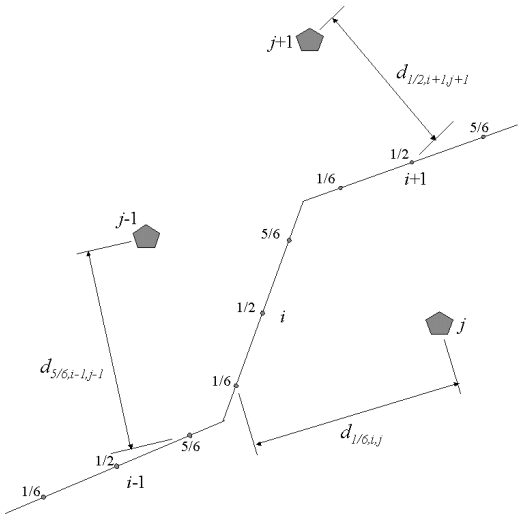

Figure 3: Threat Cost Calculation

If it is assumed that the UAVs fly at constant speed, then the fuel cost for flying along a particular edge is simply proportional to the length of the edge:

$$J_{fuel,i} = L_i.$$

The total cost for traveling along an edge comes from a weighted sum of the threat and fuel costs:

$$J_i = kJ_{fuel,i} + (1-k)J_{threat,i}.$$

The choice of k between 0 and 1 gives the designer flexibility to place weight on exposure to threats or fuel expenditure depending on the particular mission scenario. A k value closer to 1 would result in the shortest paths, with little regard for the exposure to enemy radar, while a value closer to 0 would result in paths that avoid threat exposure at the expense of longer path length.

With the cost determined for each of the Voronoi edges, the Voronoi diagram can be searched with an graph search algorithm, such as Dijkstra's algorithm [7], to find the lowest cost coarse path between the initial UAV location and the location of the target. An example of the solution to this subproblem is found in Figure 4.

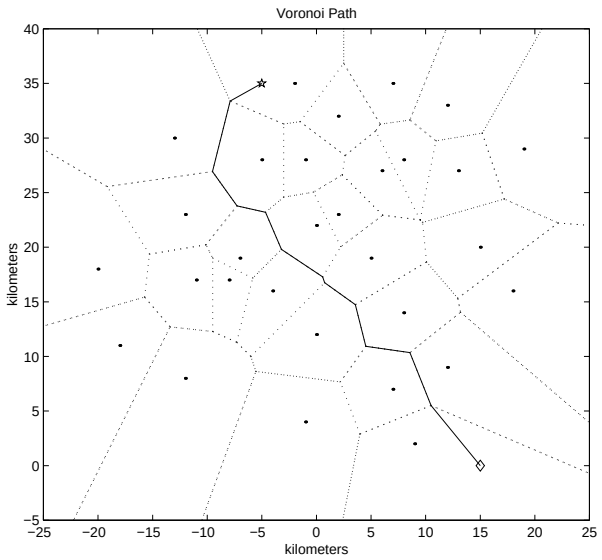

Figure 4: Initial Coarse Path

For this case, the cost weighting parameter k was chosen to give a reasonable tradeoff between proximity of the path to threats and the path length. The path shown is neither the safest possible path nor the shortest possible path, but represents a compromise between the two objectives. Both constructing the Voronoi diagram and searching it for the lowest cost path are computationally inexpensive, which is in line with our algorithm objectives. However, this series of straight line segments is unflyable by the UAV. The next step is to modify the initial coarse Voronoi path to find an optimal, yet flyable path, for the line segments nearest to the UAV.

2.2 Optimum Path Construction

The task now becomes to take this initial unflyable path and transform it into a path that is flyable by the UAV. The key idea is to build the optimal path in small pieces, so that the description of the path near the UAV is computed quickly, while the description of the path that is far away from the UAV is computed as the UAV flies the proximate segments of the path. This gives the UAV the capability to react quickly to new information, and then calculate the remainder of the path while it is flying along a

safe trajectory. To make the path dynamically feasible for the UAV, the Voronoi path is smoothed using cubic splines [1, 9] to join the path segments

2.2.1 Initial and Final Path Segments

While the initial path constructed from the Voronoi edges always generates the paths that are equidistant from the nearest threats, connecting the start and target points to the Voronoi diagram with a straight line can be less than optimal. An algorithm has been developed which takes local threat information into account when constructing these sections of the path. In this section, the procedure for constructing the initial path segment, from start point to the first Voronoi edge, will be discussed. The final path segment is created in a similar manner.

As a first step, the path is parameterized in both x and y directions as in $x = x(t), y = y(t)$. An initial spline knot is fixed at the UAV start location, while the final spline knot is fixed at the midpoint of the first Voronoi edge, as shown in Figure 5. A knot, in the context of cubic spline interpolation, is simply a point the spline is required to go through.

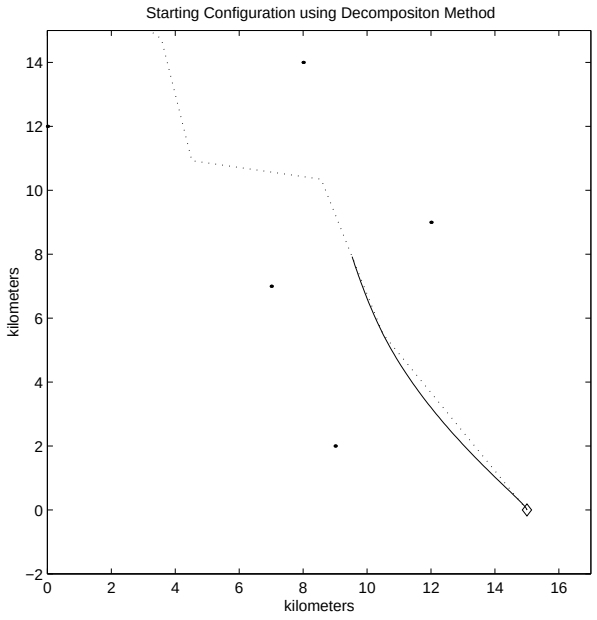

Figure 5: Initial Spline Fit from Start Point to Second leg of Initial Path

The beginning of the spline is constrained to have the specified initial heading and the end of the spline is constrained to be tangent to the Voronoi edge that it is attached to. An intermediate knot is inserted at the intersection of the two edges.

The task now is to determine how to connect these three knots. In cubic spline interpolation, each segment between knots has a unique cubic polynomial associated with it. Path continuity conditions will allow us to determine the eight coefficients that are required for each spline segment. At the interior knot, there are four conditions that must hold. Two of these are that the end of the first curve has to end on the middle knot and that the second curve has to begin on the middle knot. This gives the curve C^0 continuity. The third condition is that the first derivative of the first curve and the second curve must be equal at the knot, which allows the curve to have C^1 continuity. The fourth condition comes from the requirement that the second derivative of the first curve and the second curve must be equal at the knot, which is required for C^2 continuity. These constraints will give four conditions from which to determine the eight unknowns. The remaining four conditions come from the start and end location of the curve, and also the start and end derivatives.

The thick line in Figure 5 represents this initial spline. The goal now is to vary the location of the middle knot in such a way as to minimize the exposure of the path to the threats while not exceeding the dynamic constraints of the UAV. The exposure of the path to the threats may be quantized by measuring the distance from M discrete path locations to the N proximate threats, as shown in Figure 6.

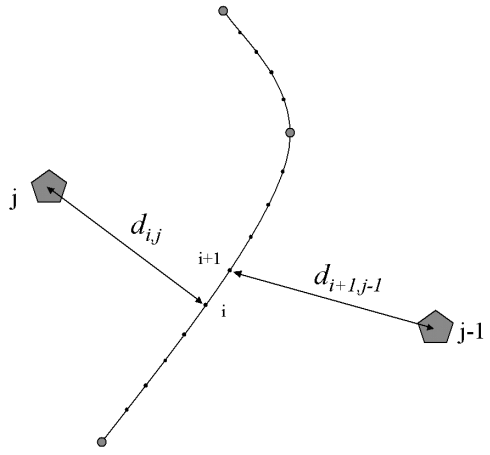

Figure 6: Cost Calculation for Initial and Final Path Segments

The threat and length cost for a path may be calculated by:

$$J_{threat} = \sum_{j=1}^{N} \sum_{i=1}^{M} \frac{1}{(d_{i,j})^4}$$

while the curvature of the path can be calculated using:

$$\kappa(t) = \frac{|\frac{dx}{dt}\frac{d^2y}{dt^2} - \frac{dy}{dt}\frac{d^2x}{dt^2}|}{((\frac{dx}{dt})^2 + (\frac{dy}{dt})^2)^{3/2}}$$

The goal of finding an optimal path that is flyable by the UAV may be posed as an optimization problem in the following form, with L representing the length of the path:

$$\begin{aligned} minimize &: kJ_{threat} + (1-k)L \\ subject\ to &: \kappa(t) - \kappa_{max} \leq 0 \end{aligned}$$

where κ_{max} represents the maximum curvature as determined from the dynamic constraints of the UAV. With this formulation, sequential quadratic programming (SQP) is used to vary the location of the middle knot in such a way as to determine a path that minimizes exposure to the threats, but also does not exceed curvature constraints. The end result of this procedure can be seen in Figure 7. In this figure, the thick line is the optimum path as determined from SQP while keeping curvature feasible within the dynamic capabilities of the UAV.

Once this step has been completed, the procedure switches into the algorithm used to compute the optimum path for the portions of the initial path that are Voronoi edges.

2.2.2 Interior Segments

The input for this algorithm consists of two adjacent Voronoi edges. The start of the spline is constrained to the midpoint of the first edge, while the end of the spline is constrained to the midpoint of the second edge. The slopes of the spline are also constrained to stay at the same slopes as their respective legs. A middle knot is inserted at the intersection of the two legs which will be moved in order to create a feasible path. The difference in this algorithm is that it uses the distance of the spline from the Voronoi path as the function to be minimized, as shown in Figure 8, in which the dashed line is the Voronoi path.

The distance from a point on the spline to the Voronoi path can be calculated using:

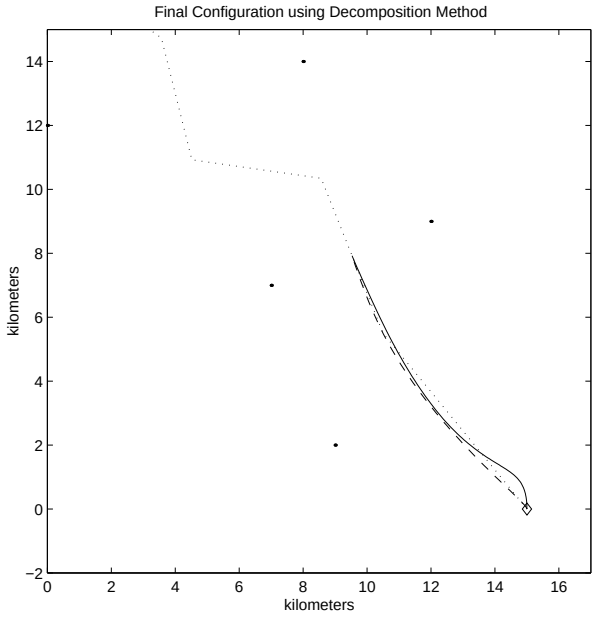

Figure 7: Final Spline Fit from Start Point to Second leg of Initial Path

$$d = \frac{\|\overrightarrow{PQ} \times \mathbf{u}\|}{\|\mathbf{u}\|}$$

where d is the distance between a point Q on the spline and the Voronoi path in which $\mathbf{u}$ is the direction vector for the Voronoi path and P is any point on the Voronoi path. For the interior segments of the initial path, the cost function becomes:

$$J_{threat} = \sum_{i=1}^{N} (d_i)^4,$$

where N is the number of discrete points on the spline. The objective is now to vary the location of the middle knot point so that the spline stays as close as possible to the Voronoi path (in a least squares sense), while not violating curvature constraints. Staying close to the Voronoi path is equivalent to avoiding the threats. While an optimization routine could be used during flight time, it is much more computationally efficient to utilize a different scheme in which the optimization is computed before flight time. At this stage of decomposition, the problem has been reduced to finding the spline that best fits two line segments, while keeping a curvature constraint feasible. Four variables are required to completely define a given case, which are the lengths

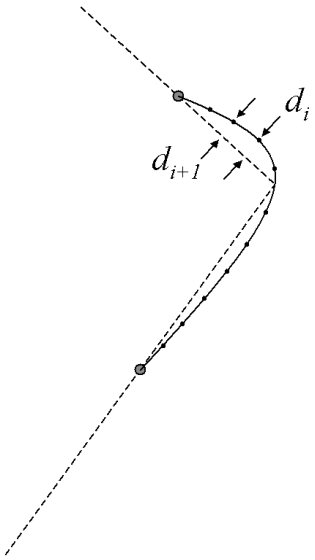

Figure 8: Cost Calculation for Interior Path Segments

of the line segments, a and b, the angle between the line segments, θ, and the allowable maximum curvature, κ_{max}, as shown in Figure 9.

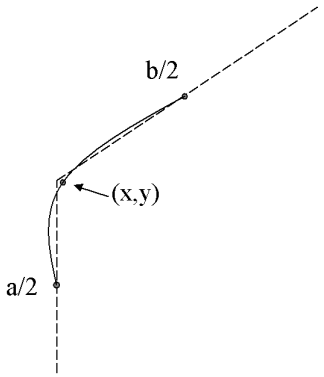

Figure 9: Decomposed Problem

The goal would be to find the (x, y) location for the middle knot that causes the spline to stay as close to the Voronoi path as possible, but does not exceed curvature constraints. New variables are defined such that L is the sum of the two leg lengths, $L = a + b$, and σ and α are used to describe the location of the middle knot, as shown in Figure 10.

It is desired to find expressions for the optimal values of $\sigma = f(a, b, \theta, r)$ and $\alpha = g(a, b, \theta, r)$. Using these relations, the optimal knot point location could be computed for a specific leg pair. Unfortunately, closed-form solutions for f and g are not available due to the complexity of the problem. Numerical solutions for σ and α are computed and stored in a look-up table. Utilizing dimensional analysis, the number of variables required for the look-up table can be reduced from four to three. If L is chosen as the repeating variable, then the problem can be translated to the nondimensional parameters:

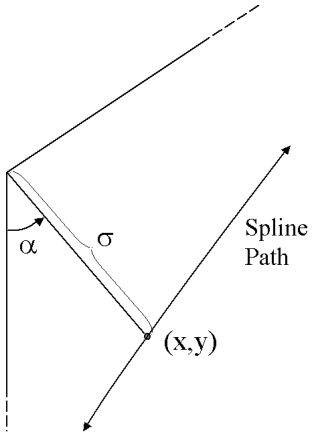

Figure 10: Definition of (x, y) Knot Point

$$\hat{\sigma} = \sigma/L \quad (1)$$

$$\hat{a} = a/L \quad (2)$$

$$\hat{r} = r_{min}/L = 1/\kappa_{max}L \quad (3)$$

This has effectively reduced the problem to finding $\hat{\sigma} = \hat{f}(\hat{a}, \theta, \hat{r})$ and $\alpha = \hat{g}(\hat{a}, \theta, \hat{r})$. A table is constructed in which for a given $\hat{a}$, $\hat{r}$, and θ, the optimum $\hat{\sigma}$ and α are calculated and stored. To use this look-up table during flight time, first the parameters a and κ are nondimensionalized into $\hat{a}$ and $\hat{r}$ using L. Then given this $\hat{a}$, $\hat{r}$, and θ, the table is used to find the optimum $\hat{\sigma}$ and α. Once this is found, it is redimensionalized to find the knot position (x, y). This procedure is summarized in Figure 11.

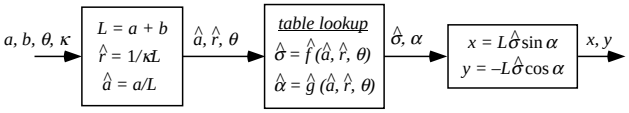

Figure 11: Summary of Table Lookup Method

The construction of this table is useful in that it now stores the optimum σ and α for any given a, b, θ and κ. When trying to determine what (x, y) should be, instead of performing a computationally costly

optimization on-line, now it is simply a matter of looking in the table to find the correct value. This speeds up the path planning algorithm considerably.

One issue with this approach is how to handle situations that are not contained in the table. For instance, if for a given maximum curvature, there is no spline that will fit between two line segments, then the lookup table will contain no solution. In order to handle this possibility, the next segment in the coarse path is used and an optimization is performed in which the objective is to avoid the threats while not exceeding curvature constraints. This is similar to the procedure used on the initial and final coarse path segments. If there is still no feasible path, then subsequent segments will be included until a feasible path can be found. This makes the method robust to line segments that have no feasible solution.

Utilizing this method, the solution using the lookup table can be seen in Figures 12 and 13. This procedure is repeated for each of the Voronoi legs until the last Voronoi leg is reached. Then the same algorithm that was used to connect the start point to the Voronoi diagram is used to connect the target location to the Voronoi diagram, taking into account the desired attack heading at the target and also local threat information. Figure 14 shows the algorithm in progress as it smoothes the Voronoi path. Figure 15 shows the completed path.

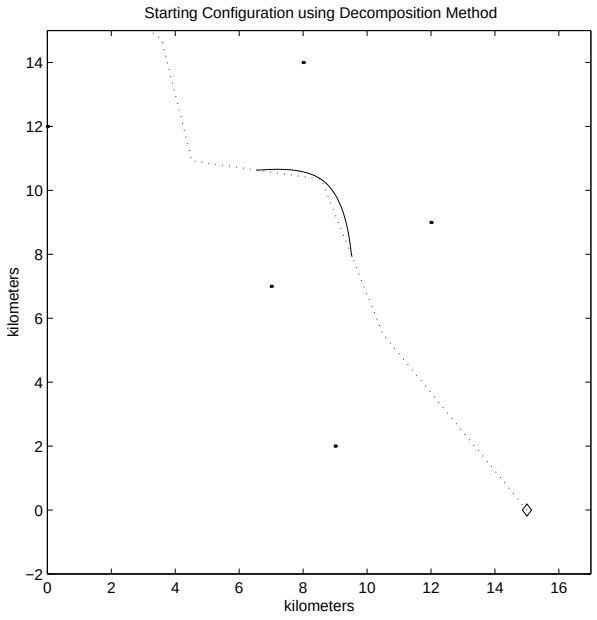

Figure 12: Initial Spline Fit for Second and Third Leg of Initial Path

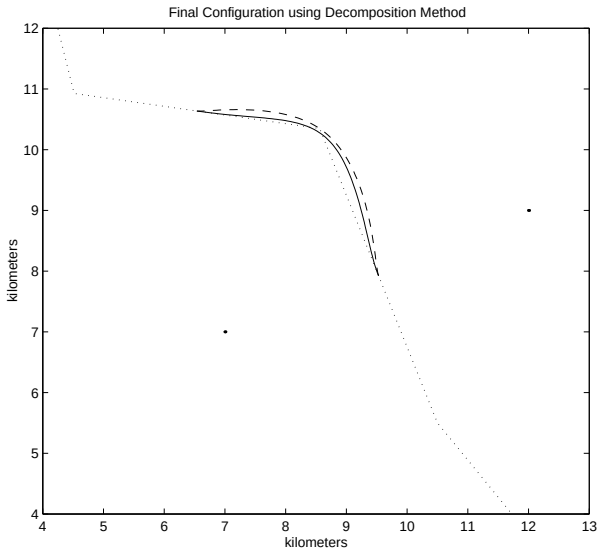

Figure 13: Final Spline Fit for Second and Third Leg of Initial Path

A particular advantage of this approach is the feature that the path nearest the UAV is calculated first, and then path details farther away from the UAV are calculated later. It is not necessary for the UAV to instantly have detailed flight information for parts of the path that are far away. Calculating the route of a UAV far into the future with high detail requires a high degree of confidence that the threat information is correct and complete. This is usually not true. Typically, new threats are discovered during the duration of the flight and the UAV must take these into account to replan a safe path. This approach quickly calculates the general flight path and detailed information that is relevant to the immediate needs of the UAV, and then uses the time that is available while flying to calculate the details of the remaining path.

3 RESULTS

The path planning strategy outlined above was tested in a simulation environment. Matlab was used on an 800 MHz PC for all the simulations. As a test case, the following scenario was proposed. As the UAV is flying along its path, previously unknown pop-up threats are detected and the UAV must quickly determine a new safe path to the target. The threat layout in Figure 15 is assumed as the initial knowledge of threats and a path is accordingly planned. However, at a point along this path, the

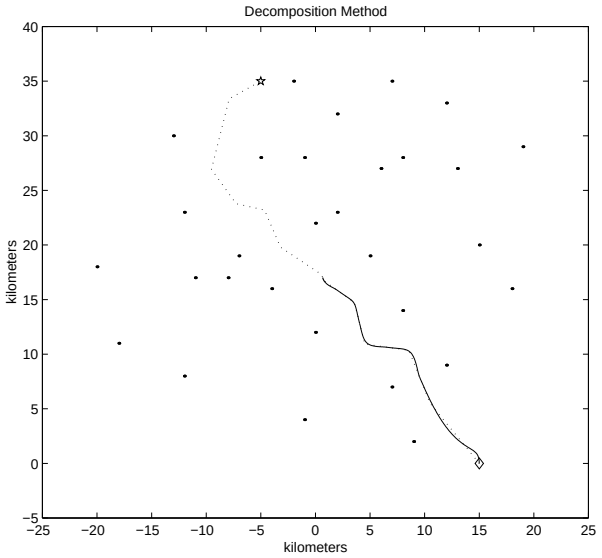

Figure 14: Smoothing Algorithm in Process

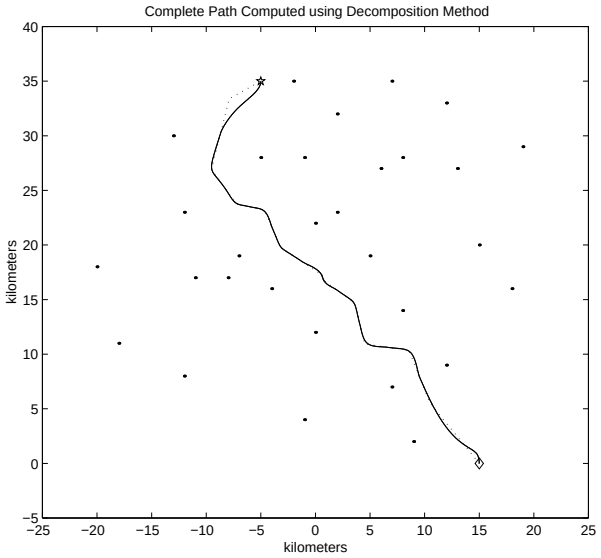

Figure 15: Smoothing Algorithm Finished

UAV learns that there are additional threats that were previously unknown. It needs to calculate a new path based on this new threat information. This scenario is shown in Figure 16.

The dotted line in Figure 16 corresponds to the original path first planned when the UAV began its mission. While the UAV (represented by the diamond) is following this path, previously unknown threats pop-up, represented by threats with circles around them. The above procedure is followed, namely the Voronoi diagram is calculated and the smoothing procedure is started. In approximately 1.3 seconds, the best initial path has been found and the first segment of that path has been smoothed by the algorithm. The UAV may now begin to fly this new optimum path. As it flies along this new optimum path, the UAV may optimize the remainder of the path. In approximately 5.8 seconds, the entire path has been optimized. The new optimum path is shown in the figure as the thick line, with the original path shown as the dotted line.

4 CONCLUSIONS

In this paper, the development of a path planner which generates a feasible path for a UAV has been presented. The key strengths of the method are: 1) the ability to find a safe path to the target while avoiding threats; 2) the paths that are generated are within the dynamic constraints imposed by the UAV; and 3) the described method is computationally efficient, allowing the algorithm to handle the possibility of pop-up threats. Using this planning approach, the ability to plan feasible paths has been demonstrated through a simulation implementing the described algorithms.

5 ACKNOWLEDGMENTS

This work was initiated at the Air Force Research Laboratory, Air Vehicles Directorate as part of the Graduate Student Summer Research Program. This work was partially supported by AFOSR grant award number F49620-01-1-0091.

References

[1] Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, Inc., 1987.

[2] Scott A. Bortoff. Path planning for unmanned air vehicles. Technical report, Air Force Research Laboratory, Air Vehicles Directorate, September 1999.

[3] P. Chandler, S. Rasmussen, and M. Pachter. UAV cooperative path planning. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, August 2000.

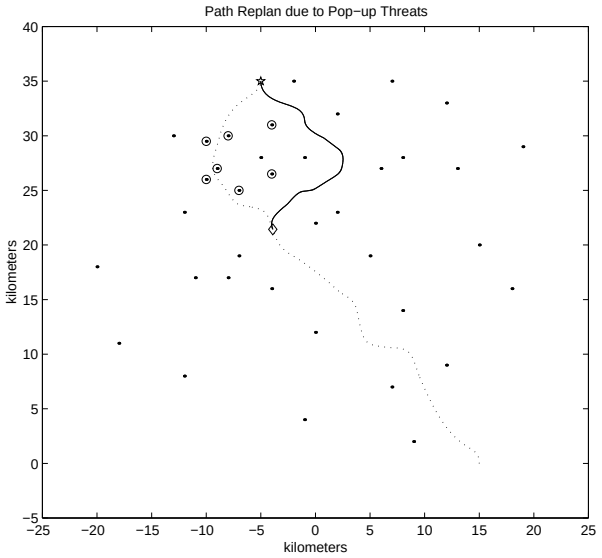

Figure 16: Path Replan due to Pop-up Threats

[4] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[5] Jeffery A. Goldman. Path planning problems and solutions. In *IEEE National Aerospace and Electronics Conference*, pages 105–108, 1994.

[6] T. McLain and R. Beard. Trajectory planning for coordinated rendezvous of unmanned air vehicles. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, August 2000.

[7] Todd K. Moon and Wynn C. Stirling. *Mathematical Methods and Algorithms*. Prentice-Hall, Inc., New Jersey, 2000.

[8] Miles B. Pellazar. Vehicle route planning with constraints using genetic algorithms. In *IEEE National Aerospace and Electronics Conference*, pages 111–119, 1998.

[9] Larry L. Schumaker. *Spline Functions: Basic Theory*. John Wiley and Sons, 1981.