where $\ddot{q}^i$ represents the coordinates φ,θ,ψ and where the standard summation convention applies. The expression on the left-side of the equal sign in Eq. (19) gives the contravariant components of the generalized acceleration.

In the case of the rigid-body motion expressed in Euler angles, five of the 27 Γ^k_{ij} symbols are identically zero, and the 22 others are given in Appendix B. Similar results were given in the simplified case $A=B$ by Luré.[5]

The different formulas and tensor components that have been given here were derived by hand as well as by our Poisson series manipulation system on the CDC-6600 computer.[4] To verify the algebra, the hand results were entered in the computer and subtracted symbolically from the computer results. These operations require the availability of three polynomial variables (A,B,C) and three harmonic variables (φ,θ,ψ). It is also necessary that the system be able to handle the negative exponents of the polynomial variables A,B,C, which occur frequently in the present application.

Appendix A: First-Kind Christoffel Symbols for the Rotating Rigid-Body Motion

$$\Gamma_{121}=\Gamma_{211}=\left[A\sin^2\psi+B\cos^2\psi-C\right]\sin\theta\cos\theta$$

$$\Gamma_{123}=\Gamma_{213}=-\tfrac{1}{2}\left[(A-B)(\cos^2\psi-\sin^2\psi)+C\right]\sin\theta$$

$$\Gamma_{131}=\Gamma_{311}=(A-B)\sin^2\theta\sin\psi\cos\psi$$

$$\Gamma_{132}=\Gamma_{312}=\tfrac{1}{2}\left[(A-B)(\cos^2\psi-\sin^2\psi)+C\right]\sin\theta$$

$$\Gamma_{231}=\Gamma_{321}=\tfrac{1}{2}\left[(A-B)(\cos^2\psi-\sin^2\psi)-C\right]\sin\theta$$

$$\Gamma_{232}=\Gamma_{322}=-(A-B)\sin\psi\cos\psi$$

$$\Gamma_{112}=-\left[A\sin^2\psi+B\cos^2\psi-C\right]\sin\theta\cos\theta$$

$$\Gamma_{113}=-(A-B)\sin^2\theta\sin\psi\cos\psi$$

$$\Gamma_{221}=(A-B)\cos\theta\sin\psi\cos\psi$$

$$\Gamma_{223}=(A-B)\sin\psi\cos\psi$$

Appendix B: Second-Kind Christoffel Symbols for the Rotating Rigid-Body Motion

$$\Gamma^1_{11}=\frac{(A-B)(A+B-C)}{AB}\cos\theta\sin\psi\cos\psi$$

$$\Gamma^1_{12}=\Gamma^1_{21}=\frac{(A+B-C)}{2}\frac{\cos\theta}{\sin\theta}\left[\frac{\sin^2\psi}{A}+\frac{\cos^2\psi}{B}\right]$$

$$\Gamma^1_{13}=\Gamma^1_{31}=\frac{(A+B-C)(A-B)}{2AB}\sin\psi\cos\psi$$

$$\Gamma^1_{23}=\Gamma^1_{32}=\frac{1}{2AB\sin\theta}\left[A(A-B-C)-(A-B)(A+B-C)\sin^2\psi\right]$$

$$\Gamma^2_{11}=\frac{-\sin\theta\cos\theta}{AB}\left[B(B-C)+(A-B)(A+B-C)\sin^2\psi\right]$$

$$\Gamma^2_{21}=\Gamma^2_{12}=\frac{(B-A)(A+B-C)}{2AB}\cos\theta\sin\psi\cos\psi$$

$$\Gamma^2_{31}=\Gamma^2_{13}=\frac{\sin\theta}{2AB}\left[B(A-B+C)-(A-B)(A+B-C)\sin^2\psi\right]$$

$$\Gamma^2_{32}=\Gamma^2_{23}=\frac{(B-A)(A+B-C)}{2AB}\sin\psi\cos\psi$$

$$\Gamma^3_{11}=\frac{B-A}{ABC}\sin\psi\cos\psi\left[(A-C)(C-B)\cos^2\theta+AB\right]$$

$$\Gamma^3_{21}=\Gamma^3_{12}=\frac{-(A+B-C)\cos^2\theta}{2AB\sin\theta}\left[A+(B-A)\sin^2\psi\right]-\frac{\sin\theta}{2C}\left[(A-B)(\cos^2\psi-\sin^2\psi)\right]-\frac{\sin\theta}{2}$$

$$\Gamma^3_{22}=\frac{A-B}{C}\sin\psi\cos\psi$$

$$\Gamma^3_{31}=\Gamma^3_{13}=\frac{-(A-B)(A+B-C)}{2AB}\cos\theta\sin\psi\cos\psi$$

$$\Gamma^3_{32}=\Gamma^3_{23}=\frac{-\cos\theta}{2AB\sin\theta}\left[-(A-B)(A+B-C)\sin^2\psi+A(A-B-C)\right]$$

$$\Gamma^1_{22}=\Gamma^1_{33}=\Gamma^2_{22}=\Gamma^2_{33}=\Gamma^3_{33}=0$$

References

[1]Goldstein, H., *Classical Mechanics,* Addison-Wesley, Reading, MA, 1957, p. 134.

[2]Desloge, E. A., *Classical Mechanics,* Wiley, New York, 1982, pp. 654, 658.

[3]Meirovitch, L., *Methods of Analytical Mechanics,* McGraw-Hill, New York, 1970, pp. 138, 157.

[4]Broucke, R., and Garthwaite, K., "A Programming System for Analytical Series Expansions on a Computer," *Celestial Mechanics,* Vol. 1, 1969, pp. 271-284.

[5]Luré, L., *Mécanique Analytique,* Masson, Paris, 1968, p. 640.

Navigation Path Planning for Autonomous Aircraft: Voronoi Diagram Approach

Jimmy Krozel* and Dominick Andrisani II†

Purdue University, West Lafayette, Indiana 78712

Introduction

FOR autonomous aircraft and intelligent pilot aids, navigation path planning may be performed by computer rather than by humans. In this Note, we consider the task of planning a path from some start location to some finish location in mountainous terrain. The technique investigated employs searching for the best paths among topologically unique paths. Candidate paths are depicted by a search graph generated using a computerized geometric construct.

In reviewing approaches to navigation path planning, one must consider path-planning research involving robot manipulator arms, mobile robot and autonomous land vehicle path planning, and existing planning algorithms for pilot aid and autonomous aircraft.[1-6] For instance, the Dynapath algorithm[3] and Beaton et al.[4] generate path plans in fine detail us-

Received May 26, 1988; revision received Feb. 5, 1989.

*Hughes Doctoral Fellow, School of Aeronautics and Astronautics. Student Member AIAA.

†Associate Professor, School of Aeronautics and Astronautics. Member AIAA.

ing tree branching and grid-based approaches, respectively. We do not use a grid approach. However, a search graph is constructed using a Voronoi diagram so that only *one* path between each neighboring mountain is represented. This represents a higher level of abstraction for the path-planning problem in our work compared to Refs. 3 and 4. A second-stage path refinement is suggested for the paths generated with our technique.

Voronoi diagrams have also been used in path-planning problems in robotics.[5] In Voronoi-based methods, the Voronoi diagram of obstacle polygon line segments is used to create a diagram of straight and parabolic arcs connected to form a search graph. Meng[6] uses such a technique for autonomous aircraft path planning. We use the Voronoi diagram of a set of points, rather than a set of line segments, to generate a search graph for path planning.

Problem Statement

The simplified navigation path-planning problem addressed here is to find a path from a start location S to a finish location F on a mountainous terrain map while minimizing the path length and the exposure to threats. The problem considered here is constrained to a constant altitude. Mountains are described by a terrain contour map and threats are described by danger regions on the contour map. For any segment of a trajectory that crosses a threat area, it is assumed that a cost associated with danger can be assigned to that segment.

Constructing a Voronoi Diagram Search Graph

A path-planning search graph depicts possible paths through the terrain/threat search space. A good search graph includes at least one path between all neighboring mountain obstacles. Paths are represented within a graph of nodes connected by arcs, where the arcs represent possible paths to be flown. The search environment can be separated into two spaces: free space and obstacle space. Free space, through which the vehicle is free to move, is unoccupied by obstacles, and obstacle space is occupied by obstacles. We will proceed to model free space with a feasible search graph, a graph including only nodes and arcs that are in free space.

Frequently, the planning objective is to minimize the path length and threat exposure. For this purpose, path length and threat cost are weighted and summed according to their relative importance; a final cost is assigned to each graph arc. This requires that the threat cost can be expressed in terms of the cost of path length.

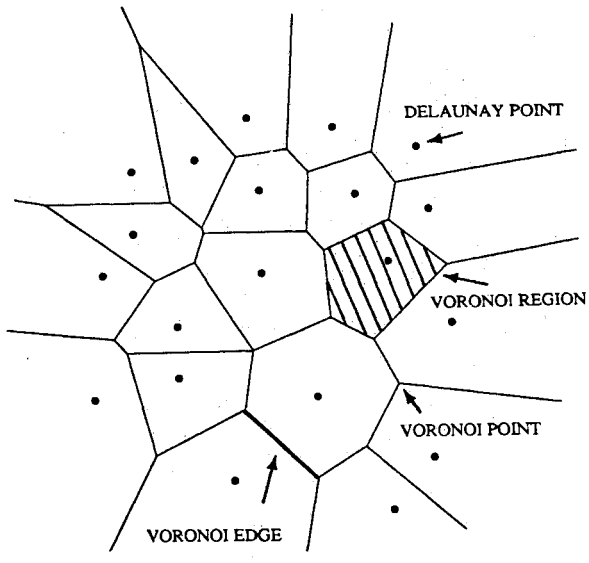

Fig. 1 Voronoi diagram for a set of 20 points.

The Voronoi diagram of a set of points is used for the purpose of creating a search graph for path planning. First, the definition of a Voronoi diagram is given to introduce this geometric construct. Then, with a simple example, the use of Voronoi diagrams for constructing terrain search graphs is presented.

A Voronoi diagram for a set of N points p_i, $1 \le i \le N$, in the Euclidean plane is a partitioning of the plane into N polygonal regions, one region associated with each point p_i. Figure 1 shows the Voronoi diagram for a set of points. A point p_i is referred to as a Delaunay point. The Voronoi region $V(p_i)$ associated with point p_i is the locus of points closer to p_i than to any of the other $N-1$ points. The Voronoi edge separating $V(p_i)$ from $V(p_j)$ is composed of the points equidistant from p_i and p_j. Not all Voronoi edges are bounded; some extend to infinity. The intersection of Voronoi edges occurs at vertices called Voronoi points. The construction of Voronoi diagrams is reviewed in Refs. 1 and 7.

A procedure is now presented for creating a search from a Voronoi diagram. Given a terrain contour map, we first polygonize the contour data, as shown in bold lines in Fig. 2. Vertices from mountain polygon obstacles are used as Delaunay point locations to model obstacles. The vertices of obstacles are points that should be avoided when traversing around polygon obstacles. Voronoi edges, by definition, maximally avoid these points. A Voronoi diagram is constructed for the set of Delaunay points describing obstacle vertices. The Voronoi edges that extend infinitely are bounded by a region that encloses the entire terrain map, creating path search arcs around the boundry of the terrain map. Figure 2 illustrates such a Voronoi diagram. The complete Voronoi diagram is shown in dashed and solid edges. Next, Voronoi edges defined by two neighboring Delaunay points of the same obstacle are removed. These edges are shown in dashed lines. Start and finish search nodes can be added to the search graph by connecting these point locations to the closest Voronoi edge. The resulting search graph is described by the remaining solid edges.

This method gives a good representation for free space using a fairly small amount of nodes. Note that only one path results between neighboring obstacles. Thus, all paths depicted by the search graph are topologically different. Another salient feature of this method is that it guarantees that the resultant search graph will depict only feasible paths, which are completely in the free space. Feasible paths are guaranteed based on parameter of the polygonization: Let the closest distance from any vertex of an obstacle to a neighboring obstacle be defined as δ. If obstacle polygonization is performed using polygon sides no greater in length than δ, then the resultant Voronoi search graph will depict only feasible paths. A proof of the feasibility of search graphs generated with this method is given in Ref. 1.

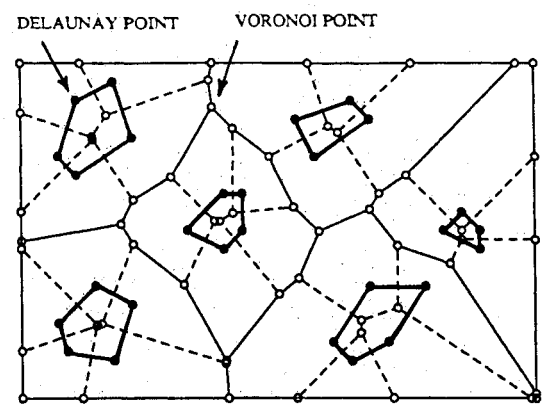

Fig. 2 Voronoi diagram graph for polygon obstacles modeled with Delaunay points at the vertices of the obstacles.

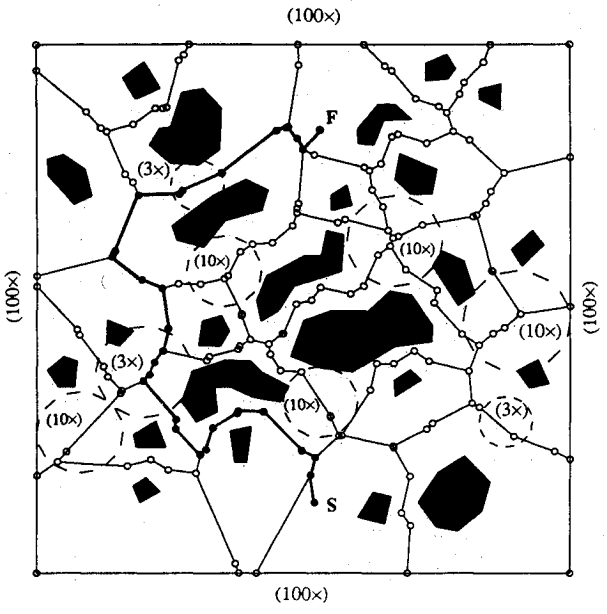

Fig. 3 Voronoi search graph for a mountainous terrain environment with threats.

Navigation Path-Planning Example

Consider a set of polygons that represent mountains at a constant altitude. Figure 3 shows a set of mountain polygon obstacles as solid black regions. The vertices of polygon mountain boundaries are used to generate the search graph illustrated. Graph arcs from map boundaries eliminate any arcs from infinite Voronoi edges, and start and finish nodes are added to the graph by connecting them to the closest Voronoi point nodes. The threat environment consists of a barrier of threats so that no path free of threat exists between the start and finish nodes. Dashed circular regions indicate threat regions.

A solution path should have minimal exposure to threats. To account for this, arcs of the search graph are assigned a threat cost in addition to a length cost. For this example, the cost assigned to each arc in the search graph is the Euclidean distance between the nodes forming the arc. Within threat regions arcs are assigned a threat cost in addition to the length cost. The threat cost is either 3 or 10 times the cost for traversing an arc of the same length in a nonthreat region. These levels of danger are indicated in parentheses as (3×) and (10×) in Fig. 3. In addition to these threat regions, a constraint is added so that the solution path does not use the boundary arcs to pass around the barrier of threats. This is established by assigning a cost of 100 times the length cost for using an arc on the boundary. This is labeled as (100×) in Fig. 3.

The Voronoi diagram search graph of Fig. 3 is searched to find the minimum cost path from the start node *S* to finish node *F*. The A^* algorithm[8] is used for this purpose. The A^* search technique uses problem-dependent information to reduce the number of nodes investigated. The heuristic function[8] is the Euclidean distance to the finish node *F*. This is an admissible heuristic,[8] and thus, the results of the A^* algorithm are optimal. The optimal path plan is shown with bold edges in Fig. 3. Since no path free of threats exists, the solution path penetrates the barrier of threats in the least dangerous region. Notice that the optimal path penetrates two (3×) threat regions. The first region is the least costly location to penetrate the initial barrier of threats. The second region is penetrated because the alternative path around this region contains a boundary arc in the search graph. Boundary arcs are heavily weighted, and thus the path through the (3×) region results.

Path Refinement

The method described provides a solution path from the start point to the finish point. While this path is guaranteed to avoid mountains, it may not represent the best path when compared to other topologically similar paths. For example, the resultant path may not penetrate individual threats at their weakest points. The next step in path planning may be to consider solutions near the Voronoi diagram path. One possible procedure for searching for path plans near the Voronoi diagram path is to search a fine grid in the region around this path. Another possible procedure is to use an iterative improvement procedure such as the path relaxation method of Thorpe.[9] Finally, when refinement is performed further information may be included in the cost function. For example, the cost function may account for the aircraft's performance capabilities or limitations. Large changes in heading angle may be punished, as well as paths that come too close to mountain boundaries.

Conclusions

A technique for generating a search graph depicting topologically unique paths around mountain boundaries at a constant altitude is presented. Our technique requires a description of mountain boundaries as polygons and generates the search graph using a geometric construct. All nodes and arcs of the search graph are guaranteed to lie in free space, thus avoiding mountain obstacles. A solution path plan is generated by searching the graph for the optimal path from a start location to a finish location.

Acknowledgments

This research was conducted at the NASA Ames Research Center, Moffett Field, California, under NASA Grant NCC 2-367. The first author is currently supported by the Artificial Intelligence Center, Hughes Research Laboratories, Malibu, California, as a Hughes Doctoral Fellow.

References

[1]Krozel, J. A., "Search Problems in Mission Planning and Navigation of Autonomous Aircraft," M. S. Thesis, Purdue Uiversity, West Lafayette, IN, May 1988.

[2]Mitchell, J. S. B., "Planning Shortest Paths," Ph.D. Thesis, Stanford University, Stanford, CA, Aug. 1986.

[3]Denton R., Jones, J. E., and Froeberg, P. L., "Demonstration of an Innovative Technique for Terrain Following/Terrain Avoidance, The Dynapath Algorithm," *IEEE National Aerospace and Electronics Conference,* Inst. of Electrical and Electronics Engineers, New York, May 1985, pp. 522-529.

[4]Beaton, R. M., Adams, M. B., and Harrison, J. V. A., "Real-Time Mission and Trajectory Planning," *IEEE Conference on Decision and Control,* Inst. of Electrical and Electronics Engineers, New York, Dec. 1987 pp. 1954-1959.

[5]Schwartz, J. and Yap, C. K., *Advances in Robotics,* Lawrence Erlbaum Associates, Hillside, NJ, 1986, pp. 187-228.

[6]Meng, A., "Flight Path Planning Under Uncertainty for Robotic Air Vehicles," *IEEE National Aerospace and Electronics Conference,* Inst. of Electrical and Electronics Engineers, New York, May 1987, pp. 359-366.

[7]Lee, D. T. and Schachter, B. J., "Two Algorithms for Constructing a Delaunay Triangulation," *International Journal of Computer and Information Sciences,* Vol. 9, No. 3, June 1980, pp. 219-242.

[8]Nilsson, N. J., *Principles of Artificial Intelligence,* Tioga Publishing, Palo Alto, CA, 1980, pp. 53-97.

[9]Thorpe, C. F., "Path Relaxation: Path Planning for a Mobile Robot," The Robotics Institute, Carnegie-Mellon Univ., Pittsburgh, PA, CMU-RI-TR-84-5, April 1984.