International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:3, 2007

Optimal Path Planner for Autonomous Vehicles

M. Imran Akram, Ahmed Pasha, and Nabeel Iqbal

***Abstract*—**In this paper a real-time trajectory generation algorithm for computing 2-D optimal paths for autonomous aerial vehicles has been discussed. A dynamic programming approach is adopted to compute *k*-best paths by minimizing a cost function. Collision detection is implemented to detect intersection of the paths with obstacles. Our contribution is a novel approach to the problem of trajectory generation that is computationally efficient and offers considerable gain over existing techniques.

***Keywords*—**dynamic programming, graph search, path planning.

I. Introduction

THE problem of real-time trajectory generation for autonomous vehicles can be decomposed into determining paths specified by waypoints and subsequently transforming the paths into *feasible* trajectories. By feasible it is implied that the trajectories generated are smooth and satisfy the kinematics of the aerial vehicles. In addition to this tactical constraints imposed on the trajectories must be maintained. We assume that each vehicle can be modeled as a point mass and travels at a constant speed and the minimum turn radius of the vehicle is known. A waypoint is assumed to be position along the trajectory where a change in vehicle's heading is initiated.

The problem of path planning has been addressed in several works. [1] presents an algorithm that exploits characteristics of both potential field methods and the Roadmap Algorithm by an incremental construction of a roadmap of free-space. An extension of the Roadmap Algorithm is presented in [2]. The method proceeds in two phases to construct a probabilistic roadmap in the configuration-space. An optimization of the running time of the planner by minimizing the number of collision checks performed during planning is presented in [3]. Path planners reported in [4]–[8] are based on the Voronoi method [9] to generate paths to a target. Computational efficiency is accomplished at the cost of grave consequences. Optimality in terms of length is lost. This becomes more pronounced in uncluttered environments where threats are sparsely distributed. In cluttered environments the paths are no longer smooth. In comparison, grid based approaches are time consuming at fine grid spacing. Shifting to coarser grid results in irregular line segments. A geometric approach to path planning composed of a set of connected line segments with a restriction on the maximum turn angle is presented in [10]. The algorithm is extended in [11] to satisfy a probability restriction for each leg of a path. An extension to the algorithm that produces paths for multiple vehicles by enlarging the obstacles is presented in [12].

The problem of trajectory generation too has been approached in several works. Spline representations of trajectories can be found in [13], [14]. The proposed trajectory in [4], [6] introduces three turns from the time when a vehicle deviates from the straight-line segment to the time when the vehicle converges to the adjacent line segment. However, the paths never traverse the waypoints except at turn angles of zero. [7] suffers similar shortcomings. [15] generates locally optimal paths and relies on *a-priori* information about final heading towards the target and fly-by points located at obstacle boundaries.

In this paper we propose a complete solution for successful planning and computation of collision-free optimal paths from a launch site to a target satisfying constraints due to vehicle kinematics and mission strategy. Our real-time trajectory generation algorithm overcomes the inadequacies of the former approaches. In section 2 we explain the system architecture in detail. Experimental results are shown in section 3. We conclude and discuss some future work in section 4.

II. System Architecture

A detailed schematic of the system architecture is shown in Fig.1. The path planner, path sorter and the trajectory generator together generate *k*-best trajectories for the vehicles. The path planner consists of the graph generator and the modified k-best paths algorithm. Paths described by straight-line segments are transformed into feasible trajectories by the trajectory generator that satisfy the dynamics of the vehicles and meet other constraints. The path sorter receives paths from the path planner and maintains a sorted list of the required number of trajectories generated by the trajectory generator.

Manuscript received November 5, 2004.

M. Imran Akram is with the Center for Advanced Research in Engineering, Evacuee Trust Complex, Islamabad, Pakistan (phone: +92-333-520-2458; fax: +92-051-287-4614; e-mail: iakram@carepvtltd.com).

Ahmed Pasha was with Avaz Networks, Software Technology Park, 5-A Constitutional Avenue, Islamabad, Pakistan. He is now with the Department of Computer Engineering, CASE, 19 Attaturk Avenue, G-5/1, Islamabad, Pakistan (e-mail: pahmed@case.edu.pk).

Nabeel Iqbal is with the Center for Advanced Research in Engineering, Evacuee Trust Complex, Islamabad, Pakistan (e-mail: nabeel@carepvtltd.com).

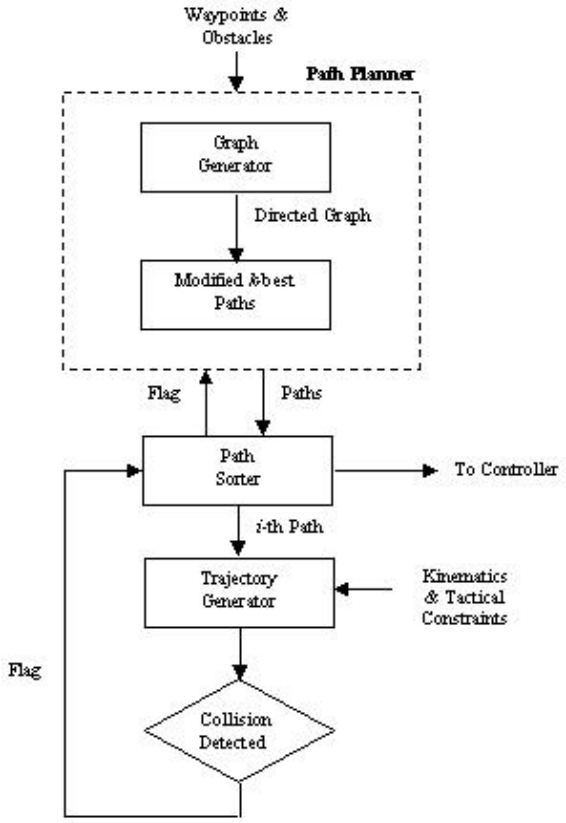

Figure 1: System Architecture

A. Path Planner

With the foci of an ellipse representing launch and target sites, a boundary is defined using the two-center *bipolar coordinate* equation:

$$r_1 + r_2 = 2a \tag{1}$$

where 2a in fig.2 equals the maximum distance covered by a vehicle during a flight. The region bounded by the ellipse defines the *mission planning area*.

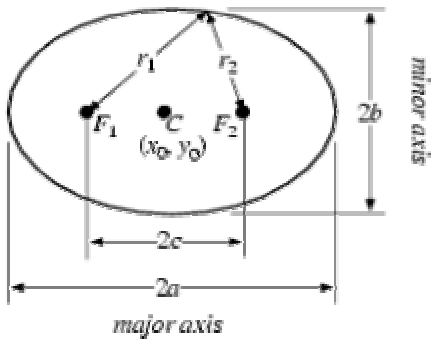

Figure 2: Mission Planning Area

Waypoints located outside the mission planning area do not contribute to graph construction. In contrast to the approach in [10] which can be applied exclusively to convex regions we maintain a general approach that can be applied to convex as well as concave regions, modeling the obstacles as *n*-sided polygons. The polygons are enlarged to account for localization errors due to GPS/INS. The enlarged polygons extending outside the mission planning area are clipped. The vertices of the polygons that remain contribute to graph generation.

B. Graph Generator

The filtering employed in the path planner reduces the complexity of the graph. The nodes of the graph comprise of waypoints and vertices of the enlarged polygons that are retained by the path planner. The nodes are arranged in order of increasing distance from a reference (x, y) that satisfies a set of three equations. Let $F_1(x_{F_1}, y_{F_1})$ represent the launch site and $F_2(x_{F_2}, y_{F_2})$ the target,

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{2}$$

$$\frac{y - y_{F_1}}{y_{F_2} - y_{F_1}} = \frac{x - x_{F_1}}{x_{F_2} - x_{F_1}} \tag{3}$$

$$\sqrt{(x - x_{F_1})^2 + (y - y_{F_1})^2} = c \tag{4}$$

The construction of the directed graph is trivial. A node is connected to all subsequent nodes in the list.

C. Modified k-best Paths

The modified *k*-best paths graph search algorithm based on Epstein's algorithm [16] is *O (m + n log n)* where *m* is the number of edges in the graph and *n* is the number of nodes, minimizes a cost function in place of edge lengths. The cost function is defined in terms of path length, node cost, and number of waypoints in a path. A node cost is the cost of inserting the node. A node whose origin is not a waypoint may have a higher cost than a node whose origin is a waypoint. The total cost of traversing an edge of the graph is expressed as:

$$J_i = \alpha J_{length,i} + \beta J_{node,i} + \chi J_{waypoint,i} \tag{5}$$

D. Path Sorter

The path sorter maintains the trajectories generated by the trajectory generator. With each new trajectory that is generated the cost of the trajectory is calculated in order to grade it. In case of path failures due to constraint violation the trajectory is rejected and additional paths are requested from the path planner to make up for the deficiency.

E. Trajectory Generator

The trajectory generator runs iteratively until the required number of trajectories is generated or the list of paths specified by straight-line segments is exhausted. In order to transform a path into a trajectory, two successive turns as opposed to three proposed in [4]–[7] are computed at each waypoint. The first turn is in the direction of the next waypoint. The second turn forces the trajectory to the original straight-line path connecting the current waypoint to the next waypoint.

As depicted in fig.3 (a) and fig.3 (b) the two cases need to be considered separately. With P as the current waypoint the locations of Q and R need to be determined.

Case 1: Absolute difference between current and next heading is less than $\frac{\pi}{2}$

Let Ω = minimum turn radius, $\phi = \angle O_1PS$, $\alpha = \angle SO_1T$

Consider ΔO_1PS, ΔO_1ST and ΔO_2RT in fig.3 (a).

$$PS = \Omega\cos\phi \tag{6}$$

$$O_1S = \Omega\sin\phi \tag{7}$$

$$ST = O_1S\tan\alpha = \Omega\sin\phi\tan\alpha \tag{8}$$

$$TR = \Omega\cos\alpha \tag{9}$$

$$O_1S = O_1T\cos\alpha = \Omega\sin\phi \tag{10}$$

$$O_1T + \frac{\Omega}{\cos\alpha} = 2\Omega \tag{11}$$

$$\alpha = \cos^{-1}\left[\frac{1+\sin\phi}{2}\right] \tag{12}$$

$$PR = PS + ST + TR$$
$$PR = \Omega(\cos\phi + \sin\phi\tan\alpha + \cos\alpha) \tag{13}$$

Using the great distance formula, R is recovered. Since O_2R is perpendicular to PS and O_1 is perpendicular to the current leg at a distance Ω from P, O_1 and O_2 are also obtained. Obtaining Q becomes trivial. The vehicle heading at Q is,

$$Vehicle\_heading\_at\_Q = O_1Q\_heading + \frac{\pi}{2} \tag{14}$$

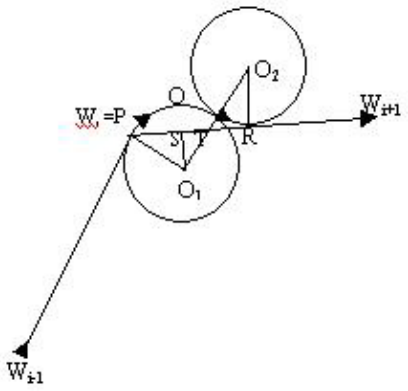

Figure 3(a): Change in heading less than $\frac{\pi}{2}$

Case 2: Absolute difference between current and next heading is greater than $\frac{\pi}{2}$

Let

$$\theta = \angle O_1PT = next\_heading - current\_heading - \frac{\pi}{2}$$

$$\phi = \angle O_2O_1S$$

$$O_2S = \Omega(1-\sin\theta) \tag{15}$$

$$\phi = \sin^{-1}\left[\frac{1-\sin\theta}{2}\right] \tag{16}$$

$$\therefore TR = O_1S = 2\Omega\cos\phi \tag{17}$$

R is recovered from

$$PR = \Omega + 2\Omega\cos\phi \tag{18}$$

Since, O_2R is perpendicular to PR and has length Ω, O_2 is obtained. The heading at Q becomes:

$$PO_1\_heading = next\_heading - \theta$$
$$O_1Q\_heading = next\_heading - \theta - \phi \tag{19}$$

$$\therefore vehicle\_heading\_at\_Q = O_1Q\_heading - \frac{\pi}{2} \tag{20}$$

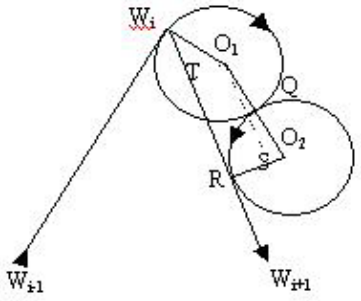

Figure 3(b): Change in heading greater than $\frac{\pi}{2}$

F. Collision Detector

In order to fully ascertain safety of the vehicle the corridor along the trajectory must be safe to account for the localization errors due to GPS/INS This involves the computationally demanding *polygon-polygon overlap* algorithm. Alternatively, the polygons can be enlarged to compensate for the error in localization. This involves the much cheaper *line-polygon intersection* algorithm. The straight-line segment and the subsequent turns are tested for intersection with the polygons. The straight-line segment is tested for intersection with the sides of the polygons. Incremental steps along the turns are tested for intersection with the sides of polygons. In case of an occurrence of an intersection the path is rejected.

III. Experimental Results

The results of three test cases generated are shown in the figures that follow. Test case one has three circular obstacles of varying radii. Test case two has five obstacles composed of a combination of three convex polygons and two circular obstacles of varying radii and test case three has four obstacles composed of a set of two convex polygons, one concave polygon and one circular obstacle. The test cases generated are similar in complexity to the test problems 1-8 in [10].

Fig.4 (a) displays four from a total of sixty-nine trajectories that were generated. The time to generate sixty-nine trajectories was less than thirty-seven seconds on a P4 2.5

GHz system. The average time to generate one trajectory is 0.536 seconds in contrast to less than one minute for [10]. Fig.4 (b) shows four trajectories from a total of sixty-four trajectories generated and fig.4 (c) shows five from a total of eighty-nine trajectories. For all three test cases the trajectory in red is the optimal trajectory. The remaining trajectories were randomly selected. Also shown along the trajectories are their respective INS corridors.

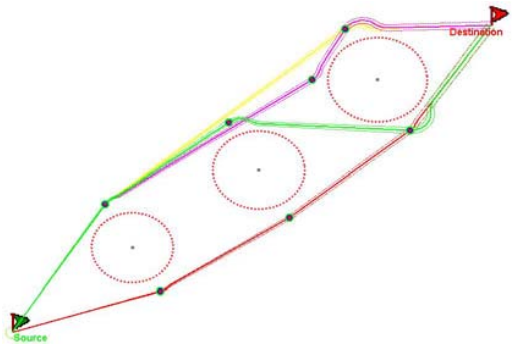

Figure 4(a): Test case 1

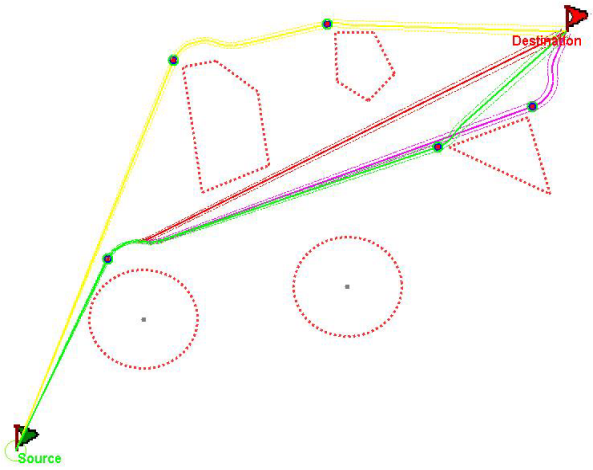

Figure 4(b): Test case 2

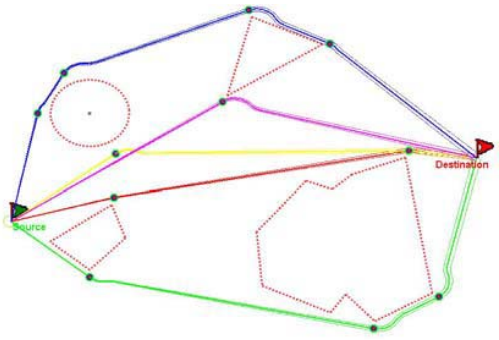

Figure 4(c): Test case 3

IV. Conclusion

This paper presents an algorithm to automatically generate smooth trajectories for autonomous aerial vehicles that avoid collisions with obstacle modeled as *n*-sided polygons. Several trajectories have been planned and further experiments are planned to evaluate the algorithm. One extension to the algorithm can be the generation of 3-D optimal trajectories for low-flying autonomous aerial vehicles.

References

[1] J. F. Canny and M. C. Lin, "An opportunistic global path planner," *Algorithmica*, vol. 10, 1993, pp 102–120.
[2] L. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration space," *IEEE Trans. on Robotics and Automation*, 12(4): 1996, pp. 566–580.
[3] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proceedings of the International Conference on Robotics and Automation*, vol. 1, 2000, pp 521–528.
[4] R. W. Beard, T. W. McLain, M. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, December 2002, pp. 911–922.
[5] T. McLain and R. Beard, "Cooperative rendezvous of multiple unmanned air vehicles," in *Proc. AIAA Guidance, Navigation and Control Conf.*, Denver, CO, Aug. 2000, AIAA Paper 2000–4369 (CD-ROM).
[6] W.B. Randal, W. M. Timothy, "Coordinated target assignment and intercept for Unmanned Air Vehicles," in *proceedings of the 2002 IEEE International Conference On Robotics & Automation*, Washington, DC. May 2002.
[7] E. P. Anderson, R.W. Beard, "An algorithmic implementation of constrained extremal control for UAVs," *AIAA Guidance, Navigation, and Control Conference and Exhibit* 5-8 August 2002, Moneterey, California.
[8] P. Chandler, S. Rasumussen, and M. Pachter, "UAV cooperative path planning," in *Proc. AIAA Guidance, Navigation, and Control Conf.*, Denver, CO, Aug. 2000, AIAA Paper 2000–4370.
[9] R. Sedgewick, *Algorithms*, 2nd ed. Reading, MA: Addison-Wesley, 1988.
[10] R. V. Helgason, J. L. Kennington, K. R. Lewis, "Cruise missile mission planning: A heuristic algorithm for automatic path generation," *Journal of Heuristics*, vol. 7, 2001, pp. 473–494.
[11] R. Helgason, J. Kennington, and K. Lewis. (1997a), "Cruise missile mission planning with a probability side constraint," Technical Report 97-CSE-3, Department of Computer Science and Engineering, SMU, Dallas, TX 75275-0122.
[12] R. Helgason, J. Kennington, and K. Lewis. (1997a), "Cruise missile strike planning: Automatic multiple path generation," Technical Report 97-CSE-24, Department of Computer Science and Engineering, SMU, Dallas, TX 75275-0122.
[13] M. B. Milam, K. Mushambi, and R. M. Murray, "A computational approach to real-time trajectory generation for constrained mechanical systems," in *Proc. IEEE Conf. Decision and Control*, Sydney, Australia, 2000, pp. 845–851.
[14] S. Sun, M. B. Egerstedt, and C. F. Martin, "Control theoretic smoothing splines," *IEEE Trans. Automat. Contr.*, vol. 45, Dec. 2000, pp. 2271–2279.
[15] G. Yang, V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints", *Proceedings of the 41st IEEE Conference on Decision and Control* Las Vegas, Nevada USA, December 2002.
[16] D. Eppstein, "Finding the k shortest paths," *SIAM Journal of Computing*, vol. 28, no. 2, 1998, pp. 995–1020.