OBSTACLE-BASED ROUTE PLANNING FOR A STEERING CONSTRAINED AUTONOMOUS VEHICLE

Steve Rowe
Charles Jacobus, Ph. D
Douglas Haanpaa
Cybernet Systems Corporation
Ann Arbor, MI

Abstract:

The Team Cybernet vehicle for the 2007 DARPA Urban Challenge[1] incorporated a route planning approach that uses sensed obstacles in the environment as the basis for potential turn placement prior to performing path search. The path search is confined to finding a set of straight-line tangents that connect circles of maximum curvature that are constructed adjacent to sensed obstacles. This approach is substantially different from traditional approaches in that the complexity of the search space is not based on the length of the path, but rather on the number of obstacles in the field. For sparse obstacle fields, this approach allows for very fast plan generation and results in paths that are guaranteed by construction to not violate steering constraints.

[1] DISCLAIMER: *The information contained in this paper does not represent the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA) or the Department of Defense. DARPA does not guarantee the accuracy or reliability of the information in this paper.*

Introduction

In the 2007 DARPA Urban Challenge, one of the qualification criteria was the ability for the autonomous vehicle to traverse bounded, open "zones" containing obstacles and parking spaces. A typical zone mission involved entering the zone from a roadway, navigating to a parking space within the zone, pulling into a parking space, backing out of the parking space, and then proceeding to a defined zone exit point (see Figure 1).

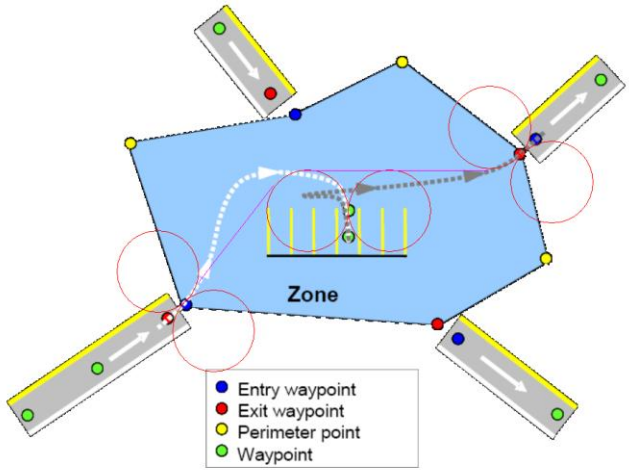

Figure 1: A Typical Zone Mission. The dotted line represents a potential vehicle path, where each dot represents one node in the search tree. The circles and the lines that link them show a path generated with our approach.

Our over-arching vehicle design philosophy was to solve exactly the problems we had to, and not go any further. Scoring for this event was based on avoiding obstacles (including lines on parking spaces), providing a cushion of at least 1 meter around all other vehicles, reaching the destination checkpoints, and not pausing for more than 5 seconds[2]. This guided our choice of solution. We made certain assumptions about the context in which the planning was to take place. We assumed that the area would be typical of a "real" parking lot, i.e. large (100x50 meters was a working baseline), flat (no obstacles obscured by the ground), and most importantly that the nature of the obstacles would be islands that needed to be driven around, rather than forming a maze that had to be driven through. We also assumed that DARPA might add a cul-de-sac to the zone to test the vehicle's ability to plan out of local minima.

Figure 2: Team Cybernet Urban Challenge Vehicle

This motivated the path planning algorithm to be described in this paper. The algorithm constrains the planning process to find paths that achieve the minimum obstacle avoidance distance and that do not require turns smaller than the vehicle's minimum turning radius to follow. The two constraints can be used to generate a reduced planning graph space to be searched as compared to more traditional discrete grid quantization of the space, and because the space is not quantized, allow any possible smooth path that meets the constraints to be generated.

Previous Work

Before beginning implementation, we reviewed previous work for route planning of nonholonomic (steering constrained) vehicles. Fundamental inspiration for our planner came from Dubins [1] and Reeds and Shepp [2] who provided clear pictorial

[2] The complete scoring documentation for the 2007 Urban Challenge can be found at http://www.darpa.mil/GRANDCHALLENGE/rules.asp.

illustrations of their underlying mathematics[3]. However, these were concerned with reachability and optimal path length and did not discuss the problem of obstacles in the field. Considerable work has been done to extend this work to address obstacles. Backer [3] considers obstacles, but in order to restrict search space (and make the problem complexity polynomial) he only considers paths inside a restrictive narrow corridor. Others, including Esquivel [4], Graf [5], Jiang [6], and Agarwal [7], perform path planning by first finding the shortest unrestricted path, and then attempt to fix the path by applying curvature constraints. This two-step process is done to guarantee finding the shortest steering-constrained path. Boissonnat [8] gives a polynomial time algorithm for constructing a path amid "moderate" obstacles[4] and proves that the path is the shortest feasible path. The restriction to moderate disjoint obstacles does not allow for one of our key assumptions, which is the cul-de-sac obstacle. One can create a moderate obstacle by constructing an artificial perimeter around an existing non-moderate obstacle, but in doing so may eliminate the only feasible path.

One other approach that dispenses with all of the constraints on obstacle shape is to build a search tree incrementally with nodes that represent a maximal left turn, maximal right turn, or straight line segment over some short time interval. Such a search tree will provide an arbitrarily close to optimum solution, but the size of the search space grows as the length of the path (and not as a function of obstacle field complexity). The algorithmic performance of searching this space is highly sensitive to the quality of the heuristics employed. Unlike the polynomial-time algorithms cited above, the search tree algorithm has a runtime complexity that is exponential in path length.

Our approach

Our planning process builds a graph consisting of nodes representing arc sections, and edges representing straight line segments that connect the arcs at tangent points. We then perform traditional graph search. The process is given below:

Step 1: For the starting configuration, construct two "navigation circles" of minimum turn radius, tangent to the starting point. One of the circles is directed clockwise, the other counter-clockwise so that the direction of travel is preserved.

Step 2: The same is done for the goal configuration.

Step 3: Scan the area with our laser range finder (working range out to about 80 meters).

Step 4: Convert the points detected by the scanner into line segments.

Step 5: Construct "pad circles" with a radius of the minimum clearance allowed centered on the endpoints of each line segment (Figure 3). The Urban Challenge rules stipulated a 1-meter clearance between vehicles and all obstacles.

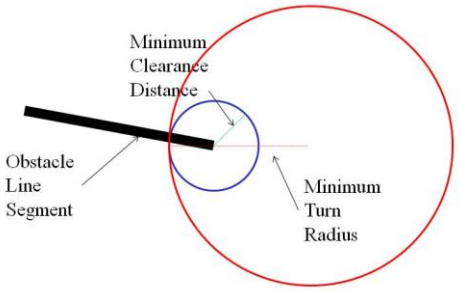

Figure 3: Construction of navigation circles around the end of an obstacle line segment.

Step 6: For each pad circle, construct 6 navigation circles tangent to the pad circle, and evenly distributed around it (Figure 4). The number of navigation circles was chosen empirically as providing a good balance between providing enough potential travel paths and expanding the size of the search space.

[3] We did not use the extensions for reverse driving found in [2] – Our vehicle only used reverse for special maneuvers, and did not need to plan reverse paths in general.

[4] The definition of "moderate" obstacles is convex and with a boundary that is differentiable and made up of line segments and circular arcs of unit radius.

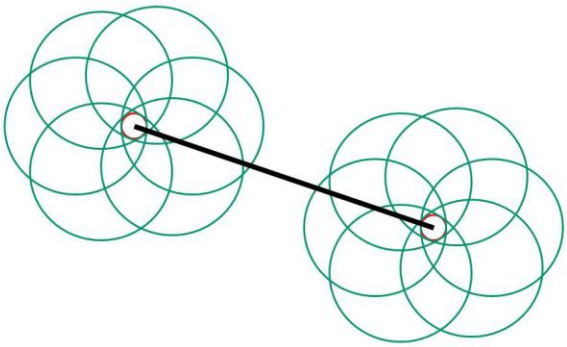

Figure 4: Completed construction of navigation circles. Each of the circles shown represents both a clockwise and counter-clockwise circle.

Step 7: For each vertex of the polygon that makes up the boundary of the zone, create a navigation circle sufficiently far inside the zone so that the nearest points of the circle to the boundary are <minimum clearance distance> away.

Step 8: For each navigation circle created in the previous steps, construct arcs by intersecting the circles with obstacle and perimeter line segments. Any arcs that lie outside the perimeter are discarded. In Figure 5, the circle has been divided into two arcs by cutting it with the obstacle line segment.

Step 9: For each arc, attempt to find a permissible *tangent* line segment to every other arc. A segment is permissible if it does not intersect any of the pad circles and does not intersect any of the obstacle or perimeter line segments. Store each of these tangents in a list associated with the source arc. Note that the set of arcs and permissible tangents forms a directed graph (digraph) of potential paths that the vehicle can follow. Not all of these paths are legal, however; In Figure 5, a path starting with segment 1 could not leave the arc on segment 2, because segment 2's tangent point lies "upstream" of segment 1's tangent point. The path entering on segment 1 and exiting on segment 3 is allowed. The path entering on segment 1 and exiting on segment 4 is not allowed, because segment 4 is actually part of a completely separate arc.

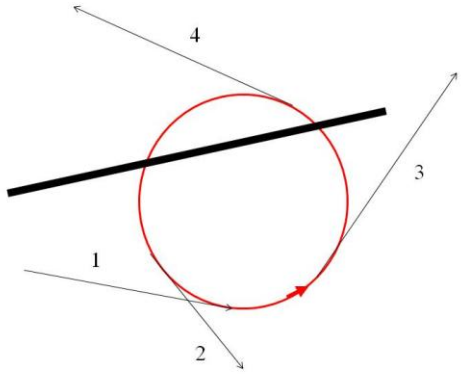

Figure 5: Tangents into and out of a pair of directed arcs. Note that it is not valid to enter the arc on segment 1 and exit on 2 or 4.

Step 10: Beginning at one of the arcs (either clockwise or counter-clockwise) tangent to the start point, search the digraph created in the previous step. Then repeat for the other (counter-clockwise or clockwise) beginning arc. The search succeeds when the last arc in the path is tangent to the goal point and the goal point lies "downstream" from the point where the segment entered the arc. We used a depth-first search with iterative deepening.

Analysis

The graph construction process (steps 1-9) is polynomial in the number of obstacle and perimeter points as follows:

Steps 1,2, 3, and 4 are constant time operations. Steps 5 and 6 are linear in the number of line segments (n) detected. Step 7 is linear in the number of perimeter points (m) that describe the boundary of the zone. Step 8 is linear in (n+m). Step 9 is quadratic in (n+m). So the graph to search can be constructed in polynomial time $O((n+m)^2)$.

The search (step 10) is a traditional tree search, and so is exponential in search depth (d), with a branching factor (b) on the order of (n+m). Search depth is the number of arcs that must be visited to reach the goal, excluding the starting arc. That is equivalent to saying that the search depth is the number of obstacles (or zone perimeter segments) that must be avoided to reach the goal configuration from the start configuration.

Results

In practice, the number of line segments created from a scan was less than 30. The number of perimeter points for the zones varied from 4 to 14. The number of obstacles that was actually in the way of our goal configuration was typically around 3. This means that our entire search process was on the order of (44)^3 = 85,000 operations. In the National Qualifying Event (NQE), three zones were navigated. We successfully traversed these, and did not incur any "stop and stare" penalties. Our algorithm was implemented in Java on a 2 GHz dual core Pentium system.

Figures 6 and 7 show the algorithm in progress (imagery reconstructed from stored path telemetry from Cybernet's second pass through Test Area B at the 2007 Urban Challenge in Victorville, California).

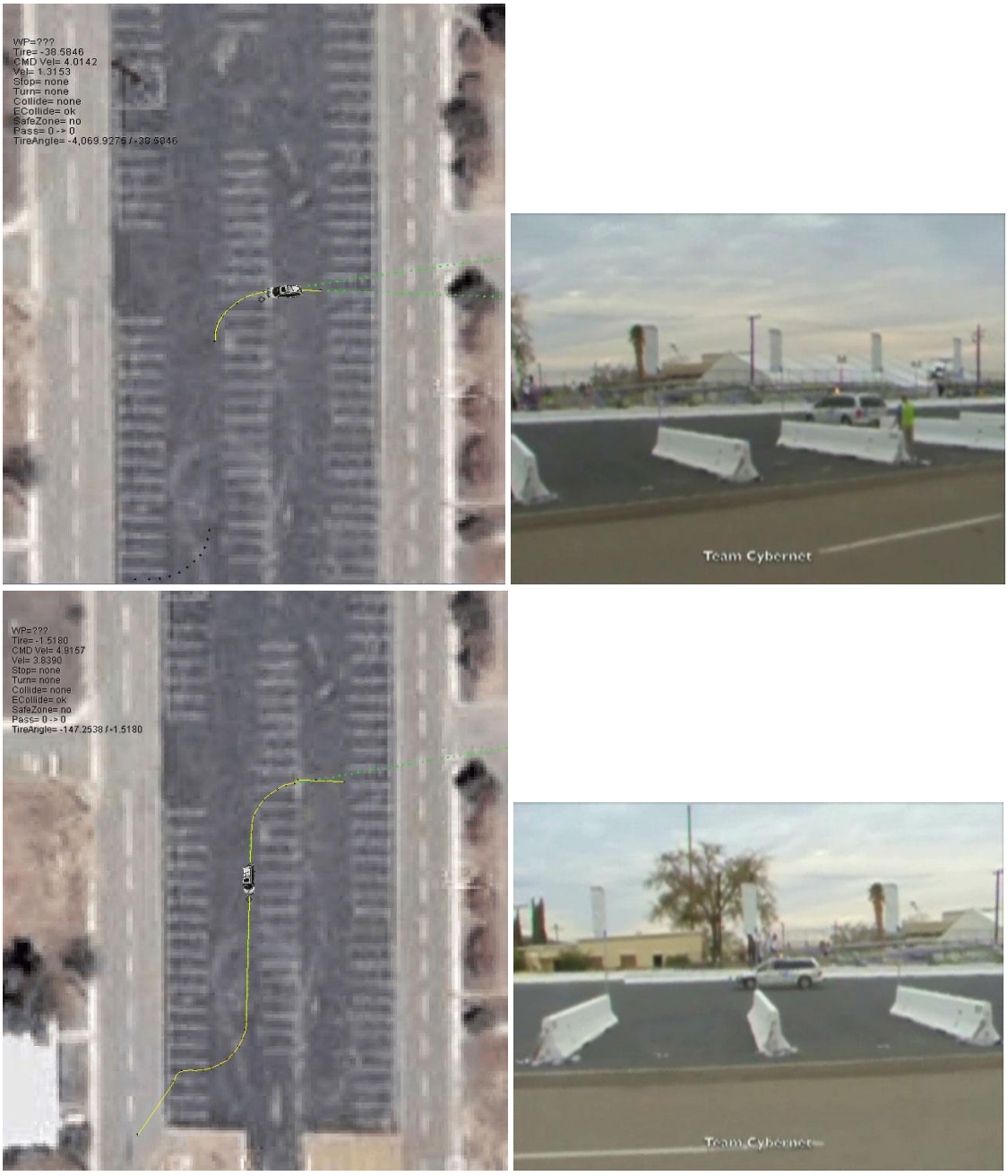

Figure 6: Path planned through the starting zone at the 2007 Urban Challenge Test Area B.

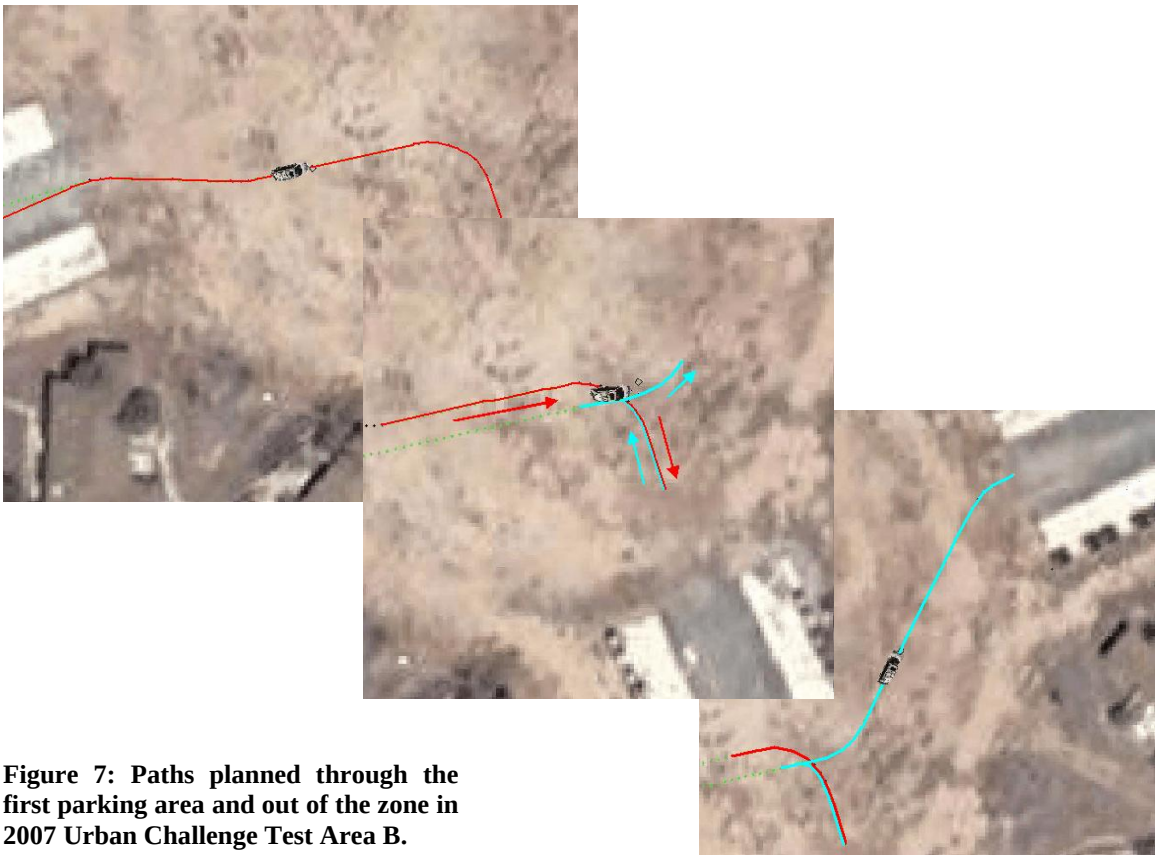

Figure 7: Paths planned through the first parking area and out of the zone in 2007 Urban Challenge Test Area B.

Shortcomings

Our approach is not complete. There may be a feasible path that our planner does not find, because it only plans along circles that are generated from the obstacles and perimeter. Because the zones presented in the Urban Challenge were uncomplicated and uncluttered, we never failed to find a solution.

Our approach is not optimal. Although we select the shortest path of those that we consider, because we do not generate all possible paths, we most likely do not generate the shortest path[5]. The lack of optimality was tolerable because we tend to generate paths that are not far from optimal.

Our approach still contains an exponential term in the computational complexity. The search space of our approach grows exponentially with the number of turns that must be made to reach the goal. For the simple zones given in the Urban Challenge, a depth of 5 was adequate. For more general obstacle fields, this will not be the case and we will need to make assumptions about the obstacles so that we can avail ourselves of one of the polynomial time algorithms.

Conclusion and Future Work

We have presented our implementation of a model-based path planner that is able to plan around obstacles that can be represented as line segments. The planner generates plans that do not violate curvature constraints, and can generate plans

[5] In the absence of obstacles, and when there is a direct path from start configuration to end configuration that does not intersect the perimeter of the zone, the planner trivially generates the CSC curve that Dubins proved was optimal. This is a special case.

sufficiently fast to avoid being penalized under the DARPA "stop-and-stare" rules.

As a future project, we will implement the Boissonnat algorithm and run it on data sets gathered during the Urban Challenge to see how it compares to what we did. This will require that we make some decisions about how the perimeter of the zone is treated (it is not necessarily convex). We will also investigate the conversion of the obstacles that we encountered into moderate obstacles so that the algorithm can be applied.

REFERENCES

[1] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *Amer. J. Math.*, 79:497–516, 1957.

[2] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific J. Math.*, 145(2):367–393, 1990.

[3] J. Backer, D. Kirkpatrick, "A Polynomial-Time Algorithm for Finding a Bounded-Curvature Path in a Narrow Corridor",

[4] W. Esquivel, L. Chiang, "Nonholonomic Path Planning Among Obstacles Subject to Curvature Restrictions", Robotica, vol. 20, 49-58, Jan 2002.

[5] B. Graf, J. Wandosell, C. Schaeffer, "Flexible Path Planning for Nonholonomic Mobile Robots",

[6] K. Jiang, L. Seneviratne, "A Shortest Path Based Path Planning Algorithm for Nonholonomic Mobile Robots", in Journal of Intelligent and Robotic Systems, 24th Ed. (1999), pp. 347-366.

[7] P. K. Agarwal, P. Raghavan, and H. Tamaki. "Motion planning for a Steering Constrained Robot Through Moderate Obstacles". In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 343–352, New York, NY, USA, 1995.

[8] J.-D. Boissonnat and S. Lazard. "A polynomial-time algorithm for computing a shortest path of bounded curvature amidst moderate obstacles" (extended abstract). In *SCG '96: Proceedings of the twelfth annual symposium on Computational geometry*, pages 242–251, New York, NY, USA, 1996. ACM Press.