Practical Search Techniques in Path Planning for Autonomous Driving

Dmitri Dolgov
AI & Robotics Group
Toyota Research Institute
Ann Arbor, MI 48105
ddolgov@ai.stanford.edu

Sebastian Thrun
Computer Science Department
Stanford University
Stanford, CA 94305
thrun@ai.stanford.edu

Michael Montemerlo
Computer Science Department
Stanford University
Stanford, CA 94305
mmde@ai.stanford.edu

James Diebel
Computer Science Department
Stanford University
Stanford, CA 94305
diebel@stanford.edu

Abstract

We describe a practical path-planning algorithm that generates smooth paths for an autonomous vehicle operating in an unknown environment, where obstacles are detected online by the robot's sensors. This work was motivated by and experimentally validated in the 2007 DARPA Urban Challenge, where robotic vehicles had to autonomously navigate parking lots. Our approach has two main steps. The first step uses a variant of the well-known A* search algorithm, applied to the 3D kinematic state space of the vehicle, but with a modified state-update rule that captures the continuous state of the vehicle in the discrete nodes of A* (thus guaranteeing kinematic feasibility of the path). The second step then improves the quality of the solution via numeric non-linear optimization, leading to a local (and frequently global) optimum. The path-planning algorithm described in this paper was used by the Stanford Racing Teams robot, Junior, in the Urban Challenge. Junior demonstrated flawless performance in complex general path-planning tasks such as navigating parking lots and executing U-turns on blocked roads, with typical full-cycle replaning times of 50–300ms.

Introduction and Related Work

We address the problem of path planning for an autonomous vehicle operating in an unknown environment. We assume the robot has adequate sensing and localization capability and must replan online while incrementally building an obstacle map. This scenario was motivated, in part, by the DARPA Urban Challenge, in which vehicles had to freely navigate parking lots. The path-planning algorithm described below was used by the Stanford Racing Team's robot, Junior in the Urban Challenge (DARPA 2007). Junior (Figure 1) demonstrated flawless performance in complex general path-planning tasks—many involving driving in reverse—such as navigating parking lots, executing U-turns, and dealing with blocked roads and intersections with typical full-cycle replanning times of 50–300ms on a modern PC.

One of the main challenges in developing a practical path planner for free navigation zones arises from the fact that the space of all robot controls—and hence trajectories—is continuous, leading to a complex continuous-variable optimization landscape. Much of prior work on search algorithms for path planning (Ersson and Hu 2001; Koenig and Likhachev 2002; Ferguson and Stentz 2005; Nash et al. 2007) yields fast algorithms for discrete state spaces, but those algorithms tend to produce paths that are non-smooth and do not generally satisfy the non-holonomic constraints of the vehicle. An alternative approach that guarantees kinematic feasibility is forward search in continuous coordinates, e.g., using rapidly exploring random trees (RRTs) (Kavraki et al. 1996; LaValle 1998; Plaku, Kavraki, and Vardi 2007). The key to making such continuous search algorithms practical for online implementations lies in an efficient guiding heuristic. Another approach is to directly formulate the path-planning problem as a non-linear optimization problem in the space of controls or parametrized curves (Cremean et al. 2006), but in practice guaranteeing fast convergence of such programs is difficult due to local minima.

Our algorithm builds on the existing work discussed above, and consists of two main phases. The first step uses a heuristic search in continuous coordinates that guarantees kinematic feasibility of computed trajectories. While lacking theoretical optimality guarantees, in practice this first

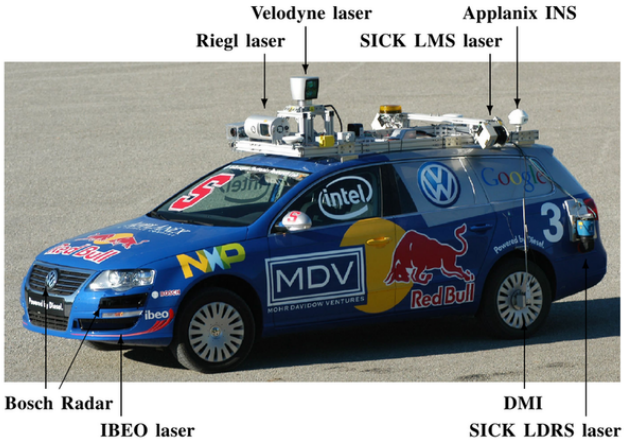


Figure 1: Junior, our entry in the DARPA Urban Challenge, was used in all experiments. Junior is equipped with several LIDAR and RADAR units, and a high-accuracy inertial measurement system.

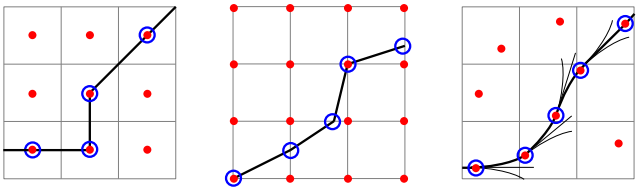


Figure 2: Graphical comparison of search algorithms. Left: A* associates costs with centers of cells and only visits states that correspond to grid-cell centers. Center: Field D* (Ferguson and Stentz 2005) and Theta* (Nash et al. 2007) associate costs with cell corners and allow arbitrary linear paths from cell to cell. Right: Hybrid A* associates a continuous state with each cell and the score of the cell is the cost of its associated continuous state.

step typically produces a trajectory that lies in a neighborhood of the global optimum. The second step uses conjugate gradient (CG) descent to locally improve the quality of the solution, producing a path that is at least locally optimal, but usually attains the global optimum as well.

Another practical challenge is the design of a cost function over paths that yields the desired driving behavior. The difficulty stems from the fact that we would like to obtain paths that are near-optimal in length, but at the same time are smooth and keep a comfortable distance to obstacles. A common way of penalizing proximity to obstacles is to use a potential field (Andrews and Hogan 1983; Khatib 1986; Pavlov and Voronin 1984; Miyazaki and Arimoto 1985). However, as has been observed by many researchers (Tilove 1990; Koren and Borenstein 1991), one of the drawbacks of potential fields is that they create high-potential areas in narrow passages, thereby making those passages effectively untraversable. To address this issues, we introduce a potential that rescales the field based on the geometry of the workspace, allowing precise navigation in narrow passages while also effectively pushing the robot away from obstacles in wider-open areas.

Hybrid-State A* Search

The first phase of our approach uses a variant of the well-known A* algorithm applied to the 3D kinematic state space of the vehicle, but with a modified state-update rule that captures continuous-state data in the discrete search nodes of A*. Just as in conventional A*, the search space (x, y, θ) is discretized, but unlike traditional A* which only allows visiting centers of cells, our hybrid-state A* associates with each grid cell a continuous 3D state of the vehicle, as illustrated in Figure 2.

As noted above, our hybrid-state A* is *not* guaranteed to find the minimal-cost solution, due to its merging of continuous-coordinate states that occupy the same cell in the discretized space. However, the resulting path is guaranteed to be drivable (rather than being piecewise-linear as in the case of standard A*). Also, in practice, the hybrid-A* solution typically lies in the neighborhood of the global optimum, allowing us to frequently arrive at the globally optimal solution via the second phase of our algorithm (which uses gradient descent to locally improve the path, as described below).

The main advantage of hybrid-state A* manifests itself in maneuvers in tight spaces, where the discretization errors become critical.

Our algorithm plans forward and reverse motion, with penalties for driving in reverse as well as switching the direction of motion.

Heuristics Our search algorithm is guided by two heuristics, illustrated in Figure 3. These heuristics do not rely on any properties of hybrid-state A* and are also applicable to other search methods (e.g., discrete A*).

The first heuristic—which we call "non-holonomic-without-obstacles"—ignores obstacles but takes into account the non-holonomic nature of the car. To compute it, we assume a goal state of $(x_g, y_g, \theta_g) = (0, 0, 0)$ and compute the shortest path to the goal from every point (x, y, θ) in some discretized neighborhood of the goal, assuming complete absence of obstacles.[1] Clearly, this cost is an admissible heuristic. We then use a max of the non-holonomic-without-obstacles cost and 2D Euclidean distance as our heuristic. The effect of this heuristic is that it prunes search branches that approach the goal with the wrong headings. Notice that because this heuristic does not depend on runtime sensor information, it can be fully pre-computed offline and then simply translated and rotated to match the current goal. In our experiments in real driving scenarios, this heuristic provided close to an order-of-magnitude improvement in the number of nodes expanded over the straightforward 2D Euclidean-distance cost.

The second heuristic is a dual of the first in that it ignores the non-holonomic nature of the car, but uses the obstacle map to compute the shortest distance to the goal by performing dynamic programming in 2D. The benefit of this heuristic is that it discovers all U-shaped obstacles and dead-ends in 2D and then guides the more expensive 3D search away from these areas.

Both heuristics are mathematically admissible in the A* sense, so the maximum of the two can be used.

Analytic Expansions The forward search described above uses a discretized space of control actions (steering). This means that the search will never reach the exact continuous-coordinate goal state (the accuracy depends on the resolution of the grid in A*). To address this precision issue, and to further improve search speed, we augment the search with analytic expansions based on the Reed-Shepp model (Reeds and Shepp 1990). In the search described above, a node in the tree is expanded by simulating a kinematic model of the car—using a particular control action—for a small period of time (corresponding to the resolution of the grid).

In addition to children generated in such a way, for some nodes, an additional child is generated by computing an optimal Reed-and-Shepp path from the current state to the goal (assuming an obstacle-free environment). The Reed-and-Shepp path is then checked for collisions against the current obstacle map, and the children node is only added to

[1] We used a 160x160 grid with 1m resolution in x-y and 5° angular resolution.

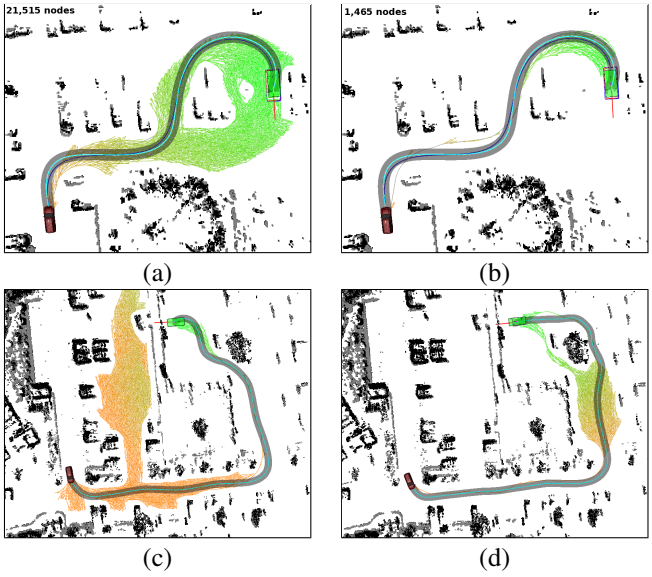


Figure 3: A* heuristics. Euclidean distance in 2D expands $21,515$ nodes (a). The non-holonomic-without-obstacles heuristic is a significant improvement: it expands $1,465$ nodes in (b), but can lead to wasteful exploration of dead-ends in more complex settings: $68,730$ nodes in (c). This is rectified by using the latter in conjunction with the holonomic-with-obstacles heuristic: $10,588$ nodes in (d).

the tree if the path is collision-free. For computational reasons, it is not desirable to apply the Reed-Shepp expansion to every node (especially far from the goal, where most such paths are likely to go through obstacles). In our implementation, we used a simple selection rule, where the Reed-Shepp expansion is applied to one of every N nodes, where N decreases as a function of the cost-to-goal heuristic (leading to more frequent analytic expansions as we get closer to the goal).

A search tree with the Reed-Shepp expansion is shown in Figure 4. The search tree generated by the short incremental expansion of nodes is shown in the yellow-green color range, and the Reed-Shepp expansions is shown as the single purple line leading to the goal. We found that this analytic extension of the search tree leads to significant benefits in both accuracy and planning time.

Path-Cost Function Using the Voronoi Field

We use the following potential field, which we call the Voronoi Field, to define the trade off between path length and proximity to obstacles. The Voronoi Field is defined as follows:

$$\rho_V(x,y) = \Big(\frac{\alpha}{\alpha + d_{\mathcal{O}}(x,y)}\Big)\Big(\frac{d_{\mathcal{V}}(x,y)}{d_{\mathcal{O}}(x,y) + d_{\mathcal{V}}(x,y)}\Big) \frac{(d_{\mathcal{O}} - d_{\mathcal{O}}^{max})^2}{(d_{\mathcal{O}}^{max})^2}, \quad (1)$$

where $d_{\mathcal{O}}$ and $d_{\mathcal{V}}$ are the distances to the nearest obstacle and the edge of the Generalized Voronoi Diagram (GVD),

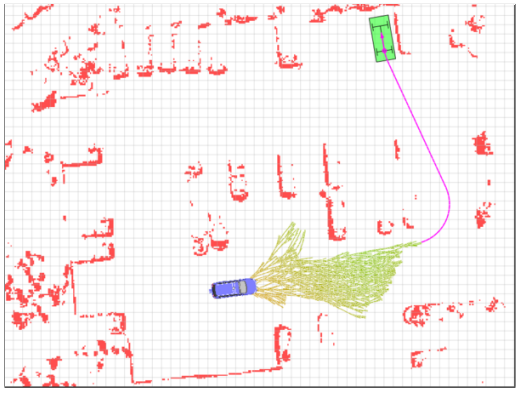


Figure 4: Analytic Reed-and-Shepp expansion. The search-tree branches corresponding to short incremental expansions are shown in the yellow-green color range, and the Reed-Shepp path is the purple segment leading towards the goal.

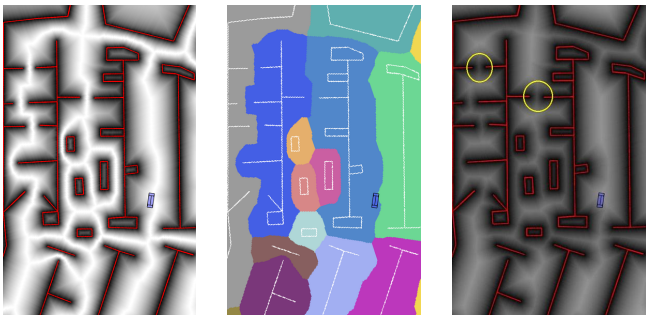


Figure 5: (a) Voronoi field in a simulated parking lot. (b) The corresponding Voronoi diagram. (c) A standard potential field with high-potential regions in narrow passages.

respectively, and $\alpha > 0, d_{\mathcal{O}} > 0$ are constants that control the falloff rate and the maximum effective range of the field. The expression in (1) is for $d_{\mathcal{O}} \leq d_{\mathcal{O}}^{max}$; otherwise, $\rho_V(x,y) = 0$.

This potential has the following properties: i) it is zero when $d_{\mathcal{O}} \geq d_{\mathcal{O}}^{max}$; ii) $\rho_V(x,y) \in [0,1]$ and is continuous on (x,y) since we cannot simultaneously have $d_{\mathcal{O}} = d_{\mathcal{V}} = 0$; iii) it reaches its maximum only within obstacles. iv) it reaches its minimum only on the edges of the GVD.

The key advantage of the Voronoi field over a conventional potential fields is the fact that the field value is scaled in proportion to the total available clearance for navigation. As a result, even narrow openings remain navigable, which is not always the case for standard potential fields.

Figure 5 illustrates this property. Figure 5a shows the 2D projection of the Voronoi field, and Figure 5b gives the corresponding generalized Voronoi diagram. Notice that narrow passages between obstacles that are close to each other are not blocked off by the potential, and there is always a continuous $\rho_V = 0$ path between them. Compare this to a naïve potential field $\rho(x,y) = \alpha(\alpha + d_{\mathcal{O}}(x,y))^{-1}$ shown in Figure 5c, which has high-potential regions in narrow pas-

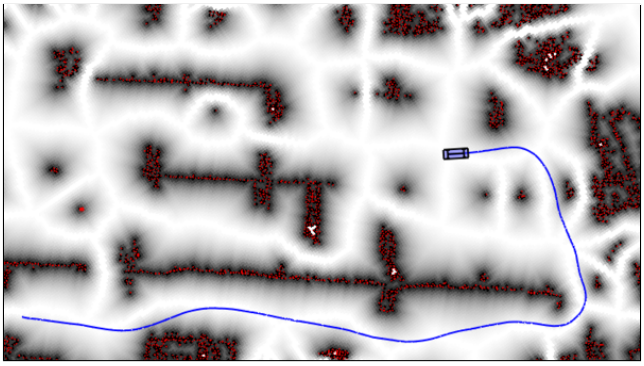


Figure 6: Voronoi Field and a trajectory driven by Junior in a real parking lot.

sages between obstacles.

Figure 6 shows the Voronoi Field and a driven trajectory for a real parking lot.

We should note that the use of Voronoi diagrams and potential fields has long been proposed in the context of robot motion planning. For example, Voronoi diagrams can be used to derive skeletonizations of the free space (Choset and Burdick 2000). However, navigating along the Voronoi graph is not possible for a non-holonomic car.

Navigation functions (Koditschek 1987; Rimon and Koditschek 1992) and Laplace potentials (Connolly, Burns, and Weiss 1990) are also similar to our Voronoi Field in that they construct potential functions free of local minima for global navigation. We do not use the Voronoi Field for global navigation. However, we observe that for workspaces with convex obstacles, the Voronoi Field can be augmented with a global attractive potential, yielding a field that has no local minima and is therefore suitable for global navigation.

Local Optimization and Smoothing

The paths produced by hybrid-state A* are often still sub-optimal and worthy of further improvement. Empirically, we find that such paths are drivable, but can contain unnatural swerves that require unnecessary steering. We therefore post-process the hybrid-state A* solution by applying the following two-stage optimization procedure. In the first stage, we formulate a non-linear optimization program on the coordinates of the vertices of the path that improves the length and smoothness of the solution. The second stage performs non-parametric interpolation using another iteration of conjugate gradient with higher-resolution path discretization.

Given a sequence of vertices $\mathbf{x}_i = (x_i, y_i),\ i \in [1, N]$, we define several quantities: $\mathbf{o}_i$, the location of the obstacle nearest to the vertex; $\Delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$, the displacement vector at the vertex; $\Delta\phi_i = |\tan^{-1}\frac{\Delta y_{i+1}}{\Delta x_{i+1}} - \tan^{-1}\frac{\Delta y_i}{\Delta x_i}|$, the change in the tangential angle at the vertex. The objective

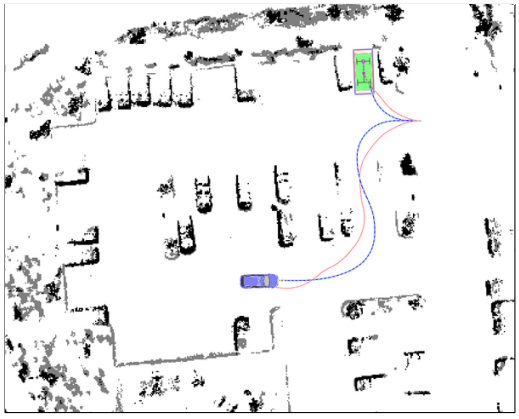


Figure 7: Hybrid-A* and CG paths for a complicated maneuver, which involves reversing into a parking spot. Hybrid-state A path (red), and the conjugate-gradient solution (blue).

function is:

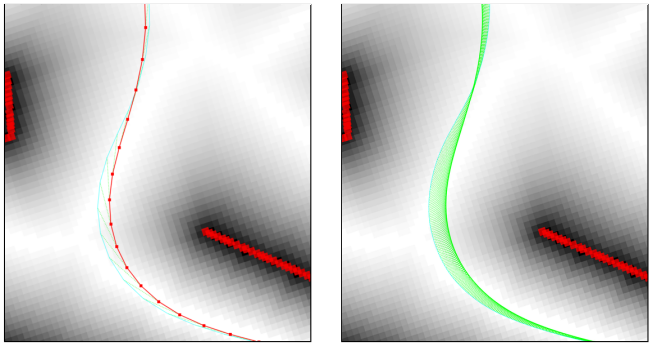
$$w_\rho \sum_{i=1}^{N} \rho_V(x_i, y_i) + w_o \sum_{i=1}^{N} \sigma_o\left(|\mathbf{x}_i - \mathbf{o}_i| - d_{\max}\right) +$$
$$w_\kappa \sum_{i=1}^{N-1} \sigma_\kappa\left(\frac{\Delta\phi_i}{|\Delta\mathbf{x}_i|} - \kappa_{max}\right) + w_s \sum_{i=1}^{N-1} (\Delta\mathbf{x}_{i+1} - \Delta\mathbf{x}_i)^2,$$

where ρ_V is the Voronoi field; κ_{max} is the maximum allowable curvature of the path (defined by the turning radius of the car), and σ_o and σ_κ are penalty functions (empirically, we found simple quadratic penalties to work well); $w_\rho, w_o, w_\kappa, w_s$ are weights.

The first term of the cost function effectively guides the robot away from obstacles in both narrow and wide passages. The second term penalizes collisions with obstacles. The third term upper-bounds the instantaneous curvature of the trajectory at every node and enforces the non-holonomic constraints of the vehicle. The fourth term is a measure of the smoothness of the path.

The gradient of the above cost function is computed in a straightforward manner as described below. For the Voronoi-field term, we have when $d_\mathcal{O} \leq d_\mathcal{O}^{max}$:

$$\begin{aligned}
\frac{\partial\rho_V}{\partial\mathbf{x}_i} &= \frac{\partial\rho_V}{\partial d_\mathcal{O}}\frac{\partial d_\mathcal{O}}{\partial\mathbf{x}_i} + \frac{\partial\rho_V}{\partial d_\mathcal{V}}\frac{\partial d_\mathcal{V}}{\partial\mathbf{x}_i}, \\
\frac{\partial d_\mathcal{O}}{\partial\mathbf{x}_i} &= \frac{\mathbf{x}_i - \mathbf{o}_i}{|\mathbf{x}_i - \mathbf{o}_i|}, \\
\frac{\partial d_\mathcal{V}}{\partial\mathbf{x}_i} &= \frac{\mathbf{x}_i - \mathbf{v}_i}{|\mathbf{x}_i - \mathbf{v}_i|}, \\
\frac{\partial\rho_V}{\partial d_\mathcal{V}} &= \frac{\alpha}{\alpha + d_\mathcal{O}}\frac{(d_\mathcal{O} - d_\mathcal{O}^{max})^2}{(d_\mathcal{O}^{max})^2}\frac{d_\mathcal{O}}{(d_\mathcal{O} + d_\mathcal{V})^2}, \\
\frac{\partial\rho_V}{\partial d_\mathcal{O}} &= \frac{\alpha}{\alpha + d_\mathcal{O}}\frac{d_\mathcal{V}}{d_\mathcal{O} + d_\mathcal{V}}\frac{(d_\mathcal{O} - d_\mathcal{O}^{max})}{(d_\mathcal{O}^{max})^2} \\
&\quad \left[\frac{-(d_\mathcal{O} - d_\mathcal{O}^{max})}{\alpha + d_\mathcal{O}} - \frac{d_\mathcal{O} - d_\mathcal{O}^{max}}{d_\mathcal{O} + d_\mathcal{V}} + 2\right],
\end{aligned}$$



(a) (b)

Figure 8: Interpolation of the CG path. The input path is shown in (a), the result of the interpolation is shown in (b). The planned paths of both the front and the rear axles are shown.

where $\mathbf{v}_i$ is a 2D vector of coordinates of the point on the edge of the Generalized Voronoi Diagram (GVD) that is closest to vertex i. We compute the nearest obstacle $\mathbf{o}_i$ and the nearest GVD-edge point $\mathbf{v}_i$ by maintaining a kd-tree of all obstacle points and GVD-edge points and updating the nearest neighbors of vertices at every iteration of conjugate gradient.

For the collision penalty with a quadratic σ_o, we have if $|\mathbf{x}_i - \mathbf{o}_i| \leq d_{max}$:

$$\frac{\partial \sigma_o}{\partial \mathbf{x}_i} = 2(|\mathbf{x}_i - \mathbf{o}_i| - d_{max}) \frac{\mathbf{x}_i - \mathbf{o}_i}{|\mathbf{x}_i - \mathbf{o}_i|}.$$

For the maximum-curvature term at vertex i, we have to take the derivatives with respect to the three points that affect the curvature at point i: $i-1$, i, and $i+1$. For this computation, the change in the tangential angle at node i is best expressed as

$$\Delta\phi_i = \cos^{-1} \frac{\Delta\mathbf{x}_i^T \Delta\mathbf{x}_{i+1}}{|\Delta\mathbf{x}_i||\Delta\mathbf{x}_{i+1}|}, \tag{2}$$

and the derivatives of the curvature $\kappa_i = \Delta\phi_i / |\Delta\mathbf{x}_i|$ with respect to the coordinates of the three nodes are then:

$$\frac{\partial \kappa_i}{\partial \mathbf{x}_i} = -\frac{1}{|\Delta\mathbf{x}_i|} \frac{\partial \Delta\phi_i}{\partial \cos(\Delta\phi_i)} \frac{\partial \cos(\Delta\phi_i)}{\partial \mathbf{x}_i} - \frac{\Delta\phi_i}{(\Delta\mathbf{x}_i)^2} \frac{\partial \Delta\mathbf{x}_i}{\partial \mathbf{x}_i},$$
$$\frac{\partial \kappa_i}{\partial \mathbf{x}_{i-1}} = -\frac{1}{|\Delta\mathbf{x}_i|} \frac{\partial \Delta\phi_i}{\partial \cos(\Delta\phi_i)} \frac{\partial \cos(\Delta\phi_i)}{\partial \mathbf{x}_{i-1}} - \frac{\Delta\phi_i}{(\Delta\mathbf{x}_i)^2} \frac{\partial \Delta\mathbf{x}_i}{\partial \mathbf{x}_{i-1}},$$
$$\frac{\partial \kappa_i}{\partial \mathbf{x}_{i+1}} = -\frac{1}{|\Delta\mathbf{x}_i|} \frac{\partial \Delta\phi_i}{\partial \cos(\Delta\phi_i)} \frac{\partial \cos(\Delta\phi_i)}{\partial \mathbf{x}_{i+1}},$$

where

$$\frac{\partial \Delta\phi_i}{\partial \cos(\Delta\phi_i)} = \frac{\partial \cos^{-1}(\cos(\Delta\phi_i))}{\partial \cos(\Delta\phi_i)} = \frac{-1}{(1 - \cos^2(\Delta\phi_i))^{1/2}}.$$

The derivative of $\cos(\Delta\phi_i)$ with respect to the coordinates of the three vertices is easiest expressed in terms of orthogonal complements:

$$\mathbf{a} \perp \mathbf{b} = \mathbf{a} - \frac{\mathbf{a}^T\mathbf{b}}{|\mathbf{b}|} \frac{\mathbf{b}}{|\mathbf{b}|}. \tag{3}$$

Introducing the following normalized orthogonal complements:

$$\mathbf{p}_1 = \frac{\mathbf{x}_i \perp (-\mathbf{x}_{i+1})}{|\mathbf{x}_i||\mathbf{x}_{i+1}|}; \qquad \mathbf{p}_2 = \frac{(-\mathbf{x}_{i+1}) \perp \mathbf{x}_i}{|\mathbf{x}_i||\mathbf{x}_{i+1}|}, \tag{4}$$

we can then express the derivatives as:

$$\begin{aligned} \frac{\partial \cos(\Delta\phi_i)}{\partial \mathbf{x}_i} &= -\mathbf{p}_1 - \mathbf{p}_2; \\ \frac{\partial \cos(\Delta\phi_i)}{\partial \mathbf{x}_{i-1}} &= \mathbf{p}_2; \qquad \frac{\partial \cos(\Delta\phi_i)}{\partial \mathbf{x}_{i+1}} = \mathbf{p}_1. \end{aligned} \tag{5}$$

Figure 7 shows the effect of the second optimization and smoothing step: the red line is the A* solution, and the blue line is the path obtained by CG optimization.

Using the CG smoothing described above, we obtain a path that is much smoother than the A* solution, but it is still piecewise linear, with a significant distance between vertices (around 0.5m–1m in our implementation). This can lead to very abrupt steering on a physical vehicle. Therefore, we further smooth the path using interpolation between the vertices of the CG solution. Many parametric interpolation techniques are very sensitive to noise in the input and exacerbate any such noise in the output (e.g., cubic splines can lead to arbitrarily large oscillations in the output as input vertices get closer to each other).

We therefore use non-parametric interpolation, where we super-sample the path by adding new vertices, and using CG to minimize curvature of the path, while holding the original vertices fixed. The result of interpolating the path in Figure 8a is shown in Figure 8b.

Results

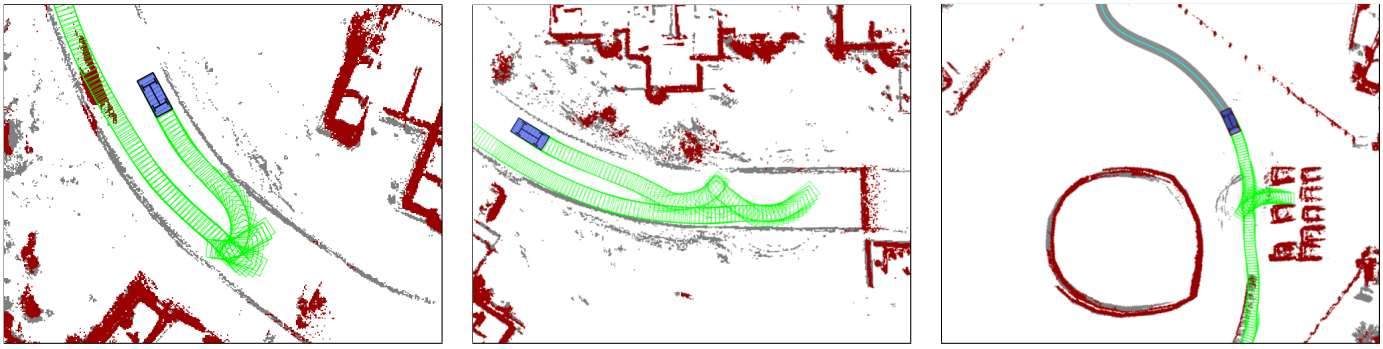
Figure 9 depicts several trajectories driven by Junior in the DARPA Urban Challenge. Figure 9a–c show U-turns on blocked roads, Figure 9d shows a task involving navigation in a parking lot.

A solution to a more complex maze-like environment computed in simulation is shown in Figure 10. A video showing the robot replanning as it incrementally detects obstacles and builds an obstacle map in scenario of Figure 10 is available at http://ai.stanford.edu/ ddolgov/gpp_maze.avi .

We used the following parameters for our planner: the obstacle map was of size 160m×160m with 0.15cm resolution; A* used a grid of size 160m×160m×360° with 0.5m x-y resolution and 5° resolution for the heading θ. Typical running times for a full replanning cycle involving the hybrid A* search, CG smoothing, and interpolation were on the order of 50–300ms.

Acknowledgments

We would like to thank Dirk Haehnel, Jesse Levinson, and other members of the Stanford Racing Team for their help with implementing and testing our path planner on the vehicle. We would also like to thank Michael James and Michael Samples for useful discussions related to this work.



(a) (b) (c)

Figure 9: Examples of trajectories generated by our planner and driven by Junior (Figure 1) in the DARPA Urban Challenge. (a) and (b) show U-turns on blocked roads; (c) shows a parking task.

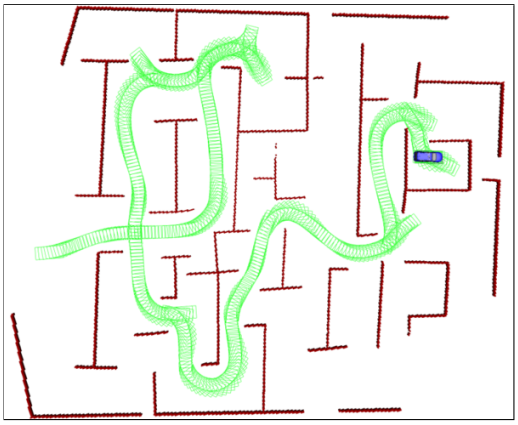


Figure 10: The shown path was generated in simulation. Note that in all cases the robot had to replan in response to obstacles being detected by its sensors (via a simulated planar rangefinder); this explains the sub-optimality of the trajectory. A video of this planning problem is available at: http://robot.cc/gpp_maze.avi

References

Andrews, J., and Hogan, N. 1983. Impedance control as a framework for implementing obstacle avoidance in a manipulator. *Control of Manufacturing Processes and Robotic Systems* 243–251.

Choset, H., and Burdick, J. 2000. Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research* 19.

Connolly, C.; Burns, J.; and Weiss, R. 1990. Path planning using laplace's equation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2102–2106.

Cremean, L. B.; Foote, T. B.; Gillula, J. H.; Hines, G. H.; Kogan, D.; Kriechbaum, K. L.; Lamb, J. C.; Leibs, J.; Lindzey, L.; Rasmussen, C. E.; Stewart, A. D.; Burdick, J. W.; and Murray, R. M. 2006. Alice: An information-rich autonomous vehicle for high-speed desert navigation. *Journal of Field Robotics*.

DARPA. 2007.

Ersson, T., and Hu, X. 2001. Path planning and navigation of mobile robots in unknown environments. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Ferguson, D., and Stentz, A. 2005. Field d*: An interpolation-based path planner and replanner. In *Proceedings of the Int. Symp. on Robotics Research (ISRR)*.

Kavraki, L.; Svestka, P.; Latombe, J.-C.; and Overmars, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4).

Khatib, O. 1986. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research.* 5(1):90–98.

Koditschek, D. E. 1987. Exact robot navigation by means of potential functions: Some topological considerations. In *IEEE Int. Conf. on Robotics and Autom.*

Koenig, S., and Likhachev, M. 2002. Improved fast replanning for robot navigation in unknown terrain. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*.

Koren, Y., and Borenstein, J. 1991. Potential field methods and their inherent limitations for mobile robot navigation. In *ICRA*.

LaValle, S. 1998. Rapidly-exploring random trees: A new tool for path planning.

Miyazaki, F., and Arimoto, S. 1985. Sensory feedback for robot manipulators. *Journal of Robotic Systems* 2(1):53–71.

Nash, A.; Daniel, K.; Koenig, S.; and Felner, A. 2007. Theta*: Any-angle path planning on grids. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 1177–1183. AAAI Press.

Pavlov, V., and Voronin, A. N. 1984. The method of potential functions for coding constraints of the external space in an intelligent mobile robot. *Soviet Automatic Control* 17(6):45–51.

Plaku, E.; Kavraki, L.; and Vardi, M. 2007. Discrete search leading continuous exploration for kinodynamic motion planning. In *Robotics: Science and Systems*.

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

Rimon, E., and Koditschek, D. E. 1992. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation* 8(5):501–518.

Tilove, R. 1990. Robotics and automation. In *IEEE International Conference on Robotics and Automation*, volume 1, 566 – 571.