Abstract—We consider the problem of optimal path planning of a forward moving simple car with a minimum turning radius (a Dubins’ car), in a known environment possibly containing impenetrable obstacles. We model this problem in the Hamilton-Jacobi partial differential equation framework, and provide a finite difference numerical method to solve it. Furthermore, we introduce and demonstrate a scheme based on this framework to steer a micro-car on a testbed.

I. INTRODUCTION

We consider the problem of generating the shortest curve from an initial position $x_0$ and a tangent direction $\theta_0$ to a target position $x_f$ (and possibly also a tangent direction $\theta_f$), with a lower bound on the curvature. This problem arises when optimally steering a simple car—a vehicle with fixed rear axle and pairs of turning front wheels, as in Fig. 1. If the wheels have a maximum turning angle constraint, $|\phi| \leq \phi_{\text{max}}$, it can be shown that, equivalently, the curvature of a forward-moving path is bounded below by $\rho_{\text{min}} = L/\tan(\phi_{\text{max}})$, where $L$ is the distance between the front and rear axles [1, Chapter 13]. Steering such a vehicle is a non-trivial task, especially in the presence of obstacles and/or the freedom of reversing direction.

A. Previous Work for Curvature Constrained Path Planning

Traditionally, combinatorial methods based on path geometry have been the primary tools for constrained path-planning problems. The problem of finding the shortest path under the curvature constraint was first introduced in the pioneering work of Dubins [2], where he characterized such paths in two dimensions in the absence of obstacles. In the robotics community, a Dubins’ car refers to a vehicle with a minimum turning radius that only moves forward. Reeds and Shepp [3] investigated its variant (still in the absence of obstacles), called the Reeds-Shepp’s car, allowing for reversal. Both Dubins’ and Reeds-Shepp’s cars have optimal paths that could be classified into known permutations of a sequence consisting of bang-bang controls; that is, traveling straight, turning fully right or fully left, and, for the latter car, reversing direction. Reachability problems of such cars have been studied in [4], [5].

There are several approaches for computing curvature-constrained optimal paths. Barraquand and Latombe [6] considered creating a reachability tree assuming that, for example, a Dubins’ car would locally only move straight, fully right or fully left for a short distance. The tree is grown from the target point backwards until one of the leaves reaches the starting point. Unnecessary cluttering of leaves can be avoided by partitioning the domain into cells and restricting to one leaf per cell [1, Chapter 14].

In the presence of obstacles, global approaches for the Dubins’ car problem have extended the analysis of characterizing path geometries. For $n$ polygonal obstacles, Wang and Agarwal [7] gave a $O((n^2/\epsilon^2) \log n)$ algorithm to construct a path that is no longer than $(1 + \epsilon)$ times the shortest $\epsilon$-robust path. Informally, a path is $\epsilon$-robust if perturbations by $\epsilon/2$ is still feasible. For similar results for a more general class of obstacles see [8] and [9].

Jacobs and Canny [10] presented a grid search algorithm (also computing $\epsilon$-robust paths), where certain nodes are strategically placed along the obstacle edges. Their setup is somewhat similar to the one proposed in this paper in that each node represents both the position and orientation of the vehicle. Two nodes are adjacent if there is a collision free trajectory connecting them. Then Dijkstra’s algorithm [11] is applied to find the shortest path among ordered sequences of adjacent nodes.

Reif and Wang [12] developed a non-uniform grid algorithm in higher dimensions using the obstacle-free optimal paths as building blocks to construct optimal paths among obstacles. Lee et. al. [13] considered the problem of finding an optimal curvature-bounded path that passes through a sequence of way-points known a priori. For the Reeds-Shepp’s car, a two phase algorithm of Laumond and Jacobs [14] first finds a feasible path ignoring curvature constraints, then modifies the path to satisfy them; the method is efficient, but the resulting paths need not be optimal.

Our approach is very similar to that of [15]: first determine a cost-to-go function (the value function (6)), then use it to compute individual trajectories. In [15], the cost-to-go func-
The performance of this algorithm is validated in a testbed setting.

In comparison to traditional methods, we introduce a practical scheme for steering an actual vehicle in a bounded domain without obstacles. We argue that the aforementioned HJ formulation is particularly suited for such a problem. In section III-C, the performance of this algorithm is validated in a testbed setting.

**II. THE HAMILTON-JACOBI FORMULATION**

**A. Definitions and Setup**

Let $\Omega \subset \mathbb{R}^2$ be a bounded, connected domain, which we call a *map*. The map is partitioned into its *free space* and *obstacles*: $\Omega = \Omega_{\text{free}} \cup \Omega_{\text{obs}}$. We will denote by $x = (x, y) \in \Omega$ a point (the *location*) on the map. Let the *pose* be the ordered triple of the location and orientation (in radians) of the vehicle: $(x, \theta) = (x, y, \theta) \in \Omega \times [0, 2\pi)$. Define a *path* or a *trajectory* as a curve $(x(\cdot), \theta(\cdot)) : [0, \infty) \to \Omega \times [0, 2\pi)$ parametrized by arc-length in $\Omega$:

$$|\dot{x}(s)| = 1, \quad s > 0. \quad (2)$$

Note the relations

$$\theta(s) = \tan^{-1}(y(s)/x(s)), \quad (3)$$

$$\dot{x}(s) = (\cos(\theta(s)), \sin(\theta(s))), \quad (4)$$

where ever $x(\cdot)$ is smooth. We say that a path is *feasible* if it is contained in $\Omega_{\text{free}} \times [0, 2\pi)$. The *average curvature* between two parametrization variables $s_1, s_2 > 0$ of a path $(x(\cdot), \theta(\cdot))$ is defined as $AC_x(s_1, s_2) = |\theta(s_2) - \theta(s_1)|/|s_1 - s_2|$.

Throughout this article, we assume that the vehicle travels with unit speed (which is automatically implied by (2)) and has a turning angle constraint that induces the constraint $AC_x \geq \rho_{\text{min}} > 0$, where $\rho_{\text{min}}$ is the *minimum turning radius*. It is easy to see that if $\theta(\cdot)$ is differentiable, the constraint is equivalent to

$$|\dot{\theta}(s)| \leq \rho_{\text{min}}^{-1}, \quad s > 0. \quad (5)$$

Denote $A_{\text{ad}, x_0, \theta_0}$ the set of *admissible path from the pose* $(x_0, \theta_0)$, that is, feasible paths that satisfy (2), (5) and $(x(0), \theta(0)) = (x_0, \theta_0)$. Such paths are precisely the trajectories generated by the Dubins’ car among impenetrable obstacles.

Finally, given a target point $x_f$, we define the *controlling value function* $u : \Omega \times [0, 2\pi) \to \mathbb{R}^+ \cup \{0\}$:

$$u(x, \phi) = \inf\{t : (x(\cdot), \theta(\cdot)) \in A_{\text{ad}, x_0, \theta_0}, x(t) = x_f\}. \quad (6)$$

In case the final orientation of the vehicle is specified, there is an additional condition $\theta(t) = \theta_f$. In other words, the function $u$ is the optimal cost-to-go for the Dubins’ car problem with given constraints, an initial pose, and a target position (or a pose).

**B. Formal Derivation of the Hamilton-Jacobi Equation**

Using (6), we start with the dynamic programming principle for our problem:

$$u(x, \phi) = \inf\{u(x(\Delta t), \theta(\Delta t)) + \Delta t : (x(\cdot), \theta(\cdot)) \in A_{\text{ad}, x, \phi}\}. \quad (7)$$

Rearranging the terms, dividing by $\Delta t$, and taking $\Delta t \to 0$, we have

$$-1 = \inf\{\nabla u \cdot (\dot{x}, \dot{\theta}) : |\dot{x}| = 1, |\dot{\theta}| \leq \rho_{\text{min}}^{-1}\}. \quad (8)$$
Furthermore, by applying (4) we obtain the Hamilton-Jacobi-Bellman equation [19]:
\[-1 = \cos(\theta)u_x + \sin(\theta)u_y + \inf_{\theta' \in [\theta - 1, \theta + 1]} \{ \theta u_\theta \}, \quad (9)\]

where subscripts denote partial derivatives. Finally, the infimum in the last term can be eliminated by assuming the bang-bang principle
\[\bar{\theta} = \pm \rho_{\min}^{-1}.\quad (10)\]

Thus, we arrive at the HJ equation
\[-1 = \cos(\theta)u_x + \sin(\theta)u_y - |u_\theta|/\rho_{\min}.\quad (11)\]

Since the target has zero cost-to-go, we have the following boundary conditions: \(u(x_f, \theta_f) = 0\), or \(u(x_f, \theta) = 0\) for \(\theta \in \{0, 2\pi\}\), if no final orientation is given. The obstacles can be interpreted as the locations of infinite cost-to-go: \(u(\Omega_{\text{obs}}, \theta) = \infty\) for \(\mathbf{x} \in \Omega_{\text{obs}}, \theta \in \{0, 2\pi\}\).

It can be shown that the bang-bang control (10) does not characterize optimal trajectories if the obstacle boundaries have curvature between 0 and 1 [9]. If such obstacle boundaries exist, the bang-bang control will induce infinite switching of controls to trace the boundaries. Such a phenomenon is referred to as chattering control [19]. In a numerical setting, the number of switchings is \(O(1/\rho_{\min})\), where \(\rho_{\min}\) is the grid refinement. As \(\rho_{\min} \rightarrow 0\), the resulting numerical path approaches the true optimal path, even when (10) is implemented.

C. Numerical Implementation

To solve (11), we propose an upwind, monotone finite difference discretization, and a fast sweeping update scheme. Set up a three dimensional uniform Cartesian grid with refinement \((h_x, h_y, h_\theta)\). Let \(u_{i,j,k} \approx u(ih_x, jh_y, kh_\theta)\) be the approximation of the solution at the grid nodes. Denote \(\xi_k = \text{sgn}(\cos(\theta_k))\) and \(\nu_k = \text{sgn}(\sin(\theta_k))\). We approximate the derivatives by upwind discretizations
\[
\begin{align*}
\cos(\theta)u_x &_{i,j,k} = -|\cos(\theta)|\frac{u_{i+\xi_k,j,k} - u_{i,j,k}}{h_x}, \quad (12) \\
\sin(\theta)u_y &_{i,j,k} = -|\sin(\theta)|\frac{u_{i,j+\nu_k,k} - u_{i,j,k}}{h_x}, \quad (13)
\end{align*}
\]

and a monotone discretization [28]
\[
\left( \frac{|u_\theta|}{\rho_{\min}} \right)_{i,j,k} = \max \left\{ \frac{u_{i,j,k+1} - u_{i,j,k}}{\rho_{\min} h_\theta}, \frac{u_{i,j,k-1} - u_{i,j,k}}{\rho_{\min} h_\theta}, 0 \right\}. \quad (14)
\]

For simplicity, we set \(h_x = h_y = h\). Substituting (12), (13), and (14) into (11), and solving for \(u_{i,j,k}\), we have
\[
u_k \begin{cases} u_{i,j,k} = |\cos(\theta)|u_{i+\xi_k,j,k} + |\sin(\theta)|u_{i,j+\nu_k,k} + h \max\{u_{i,j,k+1}/(\rho_{\min} h_\theta) - h\} & \text{if } (|u_\theta|/\rho_{\min})_{ij,k} \neq 0, \\
+ h \max\{u_{i,j,k-1}/(\rho_{\min} h_\theta) - h\} & \text{if } (|u_\theta|/\rho_{\min})_{ij,k} = 0, \\
\end{cases}
\]

\[
u_k \begin{cases} u_{i,j,k} = |\cos(\theta)|u_{i+\xi_k,j,k} + |\sin(\theta)|u_{i,j+\nu_k,k} - h & \text{if } (|u_\theta|/\rho_{\min})_{ij,k} \neq 0, \\
+ h \max\{u_{i,j,k+1}/(\rho_{\min} h_\theta) - h\} & \text{if } (|u_\theta|/\rho_{\min})_{ij,k} = 0, \\
\end{cases}
\]

if \(u_{i,j,k} = 0\) at the nearest nodes to \(x_f\) (and \(\theta_f\) for a final orientation constraint) and \(u_{i,j,k} = \infty\) at nodes inside \(\Omega_{\text{obs}}\). The boundary of the domain is also set to infinity.

To solve the system of nonlinear equations (15), (16), we apply the fast sweeping algorithm [24]. This involves the update scheme (denoting the iteration by superscripts)
\[
u_k \begin{cases} u_{i,j,k}^{n+1} = \min \{u_{i,j,k}^n + u_{i,j,k}^* + u_{i,j,k}^{**} \} & \text{if } (|u_\theta|/\rho_{\min})_{ij,k} = 0. \\
\end{cases}
\]

The boundary conditions are implemented by setting \(u_{i,j,k} = 0\) at the nearest nodes to \(x_f\) (and \(\theta_f\) for a final orientation constraint) and \(u_{i,j,k} = \infty\) at nodes inside \(\Omega_{\text{obs}}\). The boundary of the domain is also set to infinity.

The values of \(u\) not on the nodes are approximated by a nearest-neighbor interpolation.
E. Numerical Results

For all computations in this section we use $\Omega = [-1, 1]^2$ discretized uniformly by a $200 \times 200$ grid. The $\theta$ values are discretized with 200 uniform nodes. We use $\rho_{\text{min}} = 0.2358$. The time step used for the forward Euler scheme in computing the trajectories was set to the grid refinement $h = 0.01$. Typically about 5 to 7 iterations (each iteration involves 8 sweeps) were sufficient for convergence, depending on the domain shape.

Fig. 3 shows the 0.5 and 0.9 level sets of the value function $u_{i,j,k}$, with $x_f = (0, 0)$ and no final orientation constraint. No obstacles are present in this example for simplicity of presentation. The level sets of $u_{i,j,k}$ define the equal cost-to-go poses to the target. Note that the turning radius constraint is noticeable near the central axis in the 0.5 level set; for locations of larger cost-to-go, this becomes less evident.

Fig. 4. Optimal paths, among obstacles. The target (same for both cases) is shown by a ‘+’, the obstacles by black blobs and the buffer region in gray. Left: no final orientation constraint, path length is 4.9648. Right: with final orientation constraint $\theta_f = \pi/4$, path length is 5.5980.

In contrast, the option of reversibility can alleviate such a difficulty. Below, we introduce a scheme for navigating a reversible simple car that uses the HJ formulation, and demonstrate its performance in a robot testbed environment.

A. A Semi Real-time Correcting Scheme

We first propose a scheme that adjusts the optimal path of the vehicle as it deviates from its originally optimal path due to realistic factors.

Initially, set $m = 0$.

Step 1. compute the optimal path $P_m \subset \Omega_{\text{free}}$ from the pose $(x_m, \theta_m)$ to the target point $x_f$ (or pose $(x_f, \theta_f)$).

Step 2. steer the vehicle for a ‘short distance’ along $P_m$.

Step 3. record the vehicle pose as $(x_{m+1}, \theta_{m+1})$.

Step 4. let $m \leftarrow m + 1$ and repeat from step 1.

We shall call this the semi real-time correcting (SRC) scheme. Note that in step 1, one only needs to compute $P_m$ that is necessary for step 2. We assume a mechanism to track the current pose of the vehicle (step 3).

Next, to add the option of reversibility, we add another candidate path in step 1 of the SRC scheme:

Step 1. (modified) Compute the optimal path $P^1_m \subset \Omega_{\text{free}}$ from pose $(x_m, \theta_m)$, and another optimal path $P^2_m \subset \Omega_{\text{free}}$ from the the pose $(x_m, \theta_m + \pi)$. Let $P_m$ be the shorter of $P^i_m, i = 1, 2$.

Although one may be tempted to believe that this solves an approximation to the Reeds-Shepp’s car problem [3], it does not. Rather, we are solving a Dubins’ car problem from each recorded vehicle pose $(x_m, \theta_m)$, allowing the vehicle to start forward or backwards.

B. HJ-based SRC Scheme

The HJ approach to optimal path planning for the Dubins’ car problem computes optimal trajectories efficiently if the value function is provided. Indeed, the value function contains information of all optimal paths to the target location $x_f$ (or pose $(x_f, \theta_f)$). Consequently, the bulk of the computational cost is in computing the value function.

To exploit the full advantage of the HJ formulation, we consider the following situation: suppose that the map $\Omega$, the
target location $x_f \in \Omega_{free}$, and possibly the final orientation $\theta_f \in [0, 2\pi)$ are provided. Then, the corresponding value function $u_{i,j,k}$ can be computed offline prior to steering the vehicle. We outline the HJ based SRC scheme: given a map $\Omega = \Omega_{free} \cup \Omega_{obs}$, a minimum turning radius $\rho_{min}$, and a desired target location $x_f$ (or pose $(x_f, \theta_f)$).

1) precompute the value function $u_{i,j,k}$ (section II-C),
2) perform the SRC scheme (section III-A); use the method in section II-D to compute the paths $P_m$.

Strictly speaking, since the turning radius of a simple car differs for forward and backward motion, one must precompute two value function. However, we have found that the algorithm performs well in practice by using only the value function of the larger turning radius.

C. Experimental Results

We validated the HJ based SRC scheme on an autonomous simple car, using the testbed introduced in [29]. In [30] our scheme is applied to steer an autonomous vehicle while it searches for a signal source in a bounded domain with obstacles.

We used the same map as in section II-E including the buffer region, but scaled to a 140 x 140 cm square. The car size was $7 \times 3.8 \times 4.6$ cm (L x W x H) and had a minimum turning radius of $\rho_{min} = 16.51$ cm. The value function was computed on a 200 x 200 x 200 grid. The car location was sampled and recorded approximately every 15 cm travelled; this was based on the smallest distance the car can maneuver with reasonable accuracy.

Fig. 5 illustrates the SRC scheme (without the reversibility option) for a final target location $x = (45.5, 98)$. Notice how the vehicle is able to stay in the vicinity of the the optimal path even when it strays off. Fig. 6 shows the optimal path under the same parameters with the reversibility option. Fig. 7 shows a sequence of intermediate steps illustrating how the SRC scheme with reversibility plans its next motion. For this example, the reversibility option generates a path shorter by about 20%. Finally, Fig. 8 shows the case when an additional final orientation constraint $\theta_f = \pi/4$ is added.

![Fig. 5. The path taken by the car steered by the SRC scheme. The blue curve marks $P_0$, the optimal path from the initial pose. The target is shown by a red ‘+’. Length of piecewise linear car path is 348 cm.](image)

![Fig. 6. The path taken by the car steered by the SRC scheme with the reversibility option. The target is shown by a red ‘+’. Length of piecewise-linear car path is 280 cm.](image)

![Fig. 7. Intermediate steps of the SRC scheme with the reversibility option. The blue curve marks $P_m$, the optimal path emanating from the last recorded car location, and the red ‘+’ shows the target $x_f$. Note how the scheme initially generates the Dubins’ car path, then reverses its direction when the car is able to steer through the small opening above the circular obstacle.](image)

IV. Conclusions

In this article, we first formulated a Hamilton-Jacobi (HJ) equation to solve the problem of computing an optimal path, given an initial pose and a target pose, under a curvature constraint. This models a reversible simple car with a turning radius bounded from below. We provided a numerical method to solve this problem.

In the second part we applied our HJ formulation to a general scheme to steer an reversible simple car. This scheme itself can be worked in tandem with other optimal path planning algorithms. The performance of this scheme was experimentally validated. We refer the reader to a future publication for a rigorous analysis of the proposed
HJ formulation and its numerical discretization.

V. ACKNOWLEDGEMENTS

We would like to thank professor Andrea Bertozzi for her suggestions and for providing us with the robotic testbed to conduct our experiments in the UCLA Applied Mathematics Laboratory funded by an ARO MURI grant 50363-MA-MUR. We would also like to thank professors Stanley Osher and Alexander Vladimirsky for their valuable advice.

Landa’s, Takei’s, and Tsai’s research is supported by a MURI subcontract from U. of South Carolina and an NSF Algorithm for Threat Detection Grant DMS-0914465. Shen’s research is supported by an ARO MURI grant 50363-MA-MUR.

REFERENCES


