2022 American Control Conference (ACC)
Atlanta, USA, June 8-10, 2022

Time-Optimal Paths for Simple Cars with Moving Obstacles in the Hamilton-Jacobi Formulation

Christian Parkinson[1] and Madeline Ceccia[2]

Abstract— **We consider the problem of time-optimal path planning for simple nonholonomic vehicles. In previous similar work, the vehicle has been simplified to a point mass and the obstacles have been stationary. Our formulation accounts for a rectangular vehicle, and involves the dynamic programming principle and a time-dependent Hamilton-Jacobi-Bellman (HJB) formulation which allows for moving obstacles. To our knowledge, this is the first HJB formulation of the problem which allows for moving obstacles. We design an upwind finite difference scheme to approximate the equation and demonstrate the efficacy of our model with a few synthetic examples.**

I. INTRODUCTION

As automated driving technology becomes more prevalent, it is ever more important to develop interpretable trajectory planning algorithms. In this manuscript, we address the problem of trajectory planning for simple self-driving cars using a method rooted in optimal control theory and dynamic programming. We consider the vehicle pictured in fig. 1. The configuration space for the car is (x, y, θ) where (x, y) denotes the coordinate for the center of mass of the car, and $\theta \in [0, 2\pi)$ denotes the angle of inclination from the horizontal. The rear axle has length $2R$ and the distance from the rear axle to the center of mass is d. Such cars are typically propelled using actuators which supply torque to either of the rear wheels [1]. The motion is subject to a nonholonomic constraint

$$\dot{y}\cos\theta - \dot{x}\sin\theta = d\dot{\theta}. \tag{1}$$

This ensures motion (approximately) tangential to the rear wheels; indeed, in the case that $d = 0$, the constraint reduces to $dy/dx = \tan\theta$. We also assume the car has a minimum turning radius, or equivalently, a maximum angular velocity so that $|\dot{\theta}| \le W$, for some $W > 0$. LaValle [2, Chap. 13] includes an extended discussion of models for this and similar vehicles.

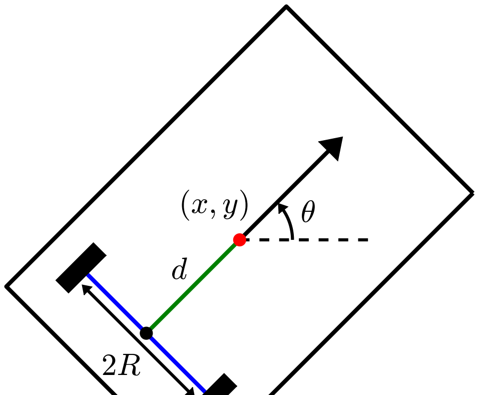

Fig. 1. A simple rectangular car.

Trajectory planning for simple cars goes back to Dubins [3] who considered that case that $d = R = 0$ (so that the car is simplified to a point mass) and only allowed unidirectional ("foward") movement. Reeds and Shepp [4] considered forward and backward motion, and proved that in the absence of obstacles, the optimal trajectories are combinations of straight lines and arcs of circles of minimum radius, and that optimal trajectories have at most two kinks where the car changes from moving forward to backward or vice versa. Later effort was devoted toward adding obstacles [5], and developing an algorithm for near-optimal trajectories which are robust to perturbation [6]. All of this work was carried out in a discrete and combinatorial fashion, breaking the paths into "turning" or "straight" segments and proving results regarding the possible combinations of these pieces.

To the authors' knowledge, this problem was first analyzed in the context of optimal control theory by Boissonnat et al. [7], [8], [9] who gave shorter proofs and extensions of results of [3] and [4]. Later, Takei and Tsai et al. [10], [11] used dynamic programming to derive a partial differential equation (PDE) which is solved by the optimal travel time function. Through all this work, the car was still simplified to a point mass. Later, the same approach was applied while considering the rectangular vehicle pictured in fig. 1 [12]. PDE-based optimal path planning algorithms have also been developed for a number of applications besides simple self-driving cars, including underwater path planning in dynamic currents [13], human navigation in a number of contexts, [14], [15], [16] and recent models for environmental crime [17], [18], [19].

Other recent work has been devoted to machine learning and variatial approaches to the problem; for example, [20], [21], [22]. Such approaches often rely on a hierarchical algorithms with global trajectory generation and local collision avoidance as in [23], [24].

[1]Christian Parkinson is a postdoctoral research associate with the Department of Mathematics, University of Arizona, 617 N. Santa Rita Ave, Tucson, AZ, 85721 chparkin@math.arizona.edu

[2]Madeline Ceccia is a student in the Department of Mathematics, California State University - Fullerton, 800 North State College Blvd. Fullerton, CA 92831 madelinececcia@csu.fullerton.edu

A. Our Contribution

We present a PDE based optimal path planning algorithm for simple self-driving vehicles. Our method is in the same spirit as [10], [11], [12]. We use dynamic programming to derive a Hamilton-Jacobi-Bellman (HJB) equation which is satisfied by the optimal travel time function. The optimal steering plan is generated using the solution to the HJB equation. To the authors' knowledge, in all previous HJB formulations of optimal trajectory planning for simple self-driving cars, the obstacles are stationary. We present a time-dependent formulation in which obstacles are allowed to move, which is a significant step in adding realism to this formulation.

In general, the time depedent HJB equation has the form

$$u_t + H(\boldsymbol{x}, \nabla u(\boldsymbol{x})) = r(x). \tag{2}$$

Because the equation is nonlinear and solutions develop kinks [25], some care is needed when solving HJB equations numerically; for example, they are not amenable to the simplest finite difference methods. Accordingly, we present an upwind finite difference scheme to solve our equation.

The HJB formulation of minimal time path planning has a number of natural advantages. Because it is rooted in optimal control theory, there are some theoretical guarantees and there are no "black box" components, so the results are interpretable. It is also a very robust modeling framework, wherein one can easily account for a number of other realistic concerns such as energy minimization. Finally, it eschews the need for hierarchical algorithms, and after a single PDE solve, this formuation can resolve optimal paths from any initial configuration to the desired ending configuration.

II. MATHEMATICAL FORMULATION

Our algorithm is based on a control theoretic formulation. Generally, to analyze control problems using dynamic programming, one derives a Hamilton-Jacobi-Bellman (HJB) equation which is satisfied by the value function. In our case, solving the HJB equation provides the optimal travel time from any given starting configuration to a fixed ending configuration, and the derivatives of the travel time function determine the optimal steering plan. For general treatment of this approach (in both theory and practice), see [26], [27].

A. Equations of Motion & Control Problem

We consider a kinematic model of a self driving car which moves about a domain $\Omega \subset \mathbb{R}^2$ in the presence of moving obstacles. Fix a horizon time $T > 0$. At any time $t \in [0, T]$, the obstacles occupy a set $\Omega_{\text{obs}}(t) \subset \Omega$, so that the free space is given by $\Omega_{\text{free}}(t) = \Omega \setminus \Omega_{\text{obs}}(t)$.

As described above, we track the current configuration of the car using variables $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) : [0, T] \to \Omega \times [0, 2\pi)$, which obey the kinematic equations

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{v} \cos\boldsymbol{\theta} - \boldsymbol{\omega} W d \sin\boldsymbol{\theta}, \\ \dot{\boldsymbol{y}} &= \boldsymbol{v} \sin\boldsymbol{\theta} + \boldsymbol{\omega} W d \cos\boldsymbol{\theta}, \\ \dot{\boldsymbol{\theta}} &= \boldsymbol{\omega} W. \end{aligned} \tag{3}$$

Here $W > 0$ is a bound on the angular velocity of the vehicle which enforces bounded curvature of the trajectory. The control variables are $\boldsymbol{v}(\cdot), \boldsymbol{\omega}(\cdot) \in [-1, 1]$, representing the tangential and angular velocity respectively. By taking velocities to be control variables, we are neglecting some of the ambient dynamics. In a more complete dynamic model, the control variables would be the torques applied to the rear wheels by the actuators. For a derivation of both this kinematic model and a dynamic model, see [28], and for generalizations of the kinematic model, see [29]. In particular, when $d = 0$, our kinematics revert to that of the Dubins car with forward and backward motion [3], [4]. We opt to include d so as to more accurately model the car's interaction with obstacles.

For configurations $(x, y, \theta) \in \Omega \times [0, 2\pi)$, define $C(x, y, \theta) \subset \Omega$ to be the space occupied by the car. In general, this could be any shape but for our purposes it will be a rectangle as pictured in fig. 1. Then given a desired ending configuration (x_f, y_f, θ_f), a trajectory $(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{\theta}(t))$ is referred to as *admissable* if each of the following is true:

(1) it obeys (3) for $t \in [0, T]$,
(2) $C(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{\theta}(t)) \cap \Omega_{\text{obs}}(t) = \varnothing$ for all $t \in [0, T]$,
(3) $(\boldsymbol{x}(T), \boldsymbol{y}(T), \boldsymbol{\theta}(T)) = (x_f, y_f, \theta_f)$.

Here (2) signifies that the car does not collide with obstacles, and (3) signifies that the trajectory ends at the desired ending configuration.

Given an initial configuration (x, y, θ), the goal is then to resolve the steering plan $\boldsymbol{v}(t), \boldsymbol{\omega}(t)$ that determines the minimal time required to traverse an admissable trajectory from (x, y, θ) to (x_f, y_f, θ_f).

B. The Dynamic Programming Approach

We resolve the optimal steering plan using dynamic programming and a Hamilton-Jacobi-Bellman (HJB) equation. To analyze the problem in the dynamic programming framework, we first define the travel-time function. For a given configuration $(x, y, \theta) \in \Omega \times [0, 2\pi)$ and time $t \in [0, T]$, we restrict ourselves to trajectories $(\boldsymbol{x}(\cdot), \boldsymbol{y}(\cdot), \boldsymbol{\theta}(\cdot))$ such that $(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{\theta}(t)) = (x, y, \theta)$. For such trajectories, if $\boldsymbol{v}(\cdot), \boldsymbol{\omega}(\cdot)$ is the corresponding steering plan, we define the first arrival time

$$t^*_{\boldsymbol{v},\boldsymbol{\omega}} = \inf\{s : (\boldsymbol{x}(s), \boldsymbol{y}(s), \boldsymbol{\theta}(s)) = (x_f, y_f, \theta_f)\}. \tag{4}$$

The cost functional for the control problem is then

$$\mathscr{T}(x, y, \theta, t, \boldsymbol{v}(\cdot), \boldsymbol{\omega}(\cdot)) = \begin{cases} t^*_{\boldsymbol{v},\boldsymbol{\omega}}, & \text{if } t^*_{\boldsymbol{v},\boldsymbol{\omega}} \le T, \\ +\infty, & \text{otherwise.} \end{cases} \tag{5}$$

The optimal travel time function is then defined

$$u(x, y, \theta, t) = \inf_{\boldsymbol{v}(\cdot), \boldsymbol{\omega}(\cdot)} \mathscr{T}(x, y, \theta, t, \boldsymbol{v}(\cdot), \boldsymbol{\omega}(\cdot)). \tag{6}$$

Intuitively, $u(x, y, \theta, t)$ is the minimal time required to steer the car to (x_f, y_f, θ_f), given that the car is at (x, y, θ) at time t. Note that if (x, y, θ) is far from (x_f, y_f, θ_f) and t is close to T, there may be no way to steer the car to the ending configuration in the allotted time. If this is the case, then $u(x, y, \theta, t) = +\infty$. However, if there are any admissable trajectories $(\boldsymbol{x}(\cdot), \boldsymbol{y}(\cdot), \boldsymbol{\theta}(\cdot))$ such that $(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{\theta}(t)) = (x, y, \theta)$, then $u(x, y, \theta, t) \leq T$.

We want to derive a partial differental equation satisfied by the optimal travel time function. The dynamic programming principle [30] for this control problem is

$$\begin{aligned} &u(x, y, \theta, t) = \\ &\qquad \delta + \inf_{\boldsymbol{v}(\cdot), \boldsymbol{\omega}(\cdot)} \{u(\boldsymbol{x}(t+\delta), \boldsymbol{y}(t+\delta), \boldsymbol{\theta}(t+\delta), t+\delta)\} \end{aligned} \tag{7}$$

where $(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{\theta}(t)) = (x, y, \theta)$ and the infimum is taken with respect to the values $\boldsymbol{v}(s), \boldsymbol{\omega}(s)$ for $s \in (t, t+\delta)$.

Supposing that $u(x, y, \theta, t)$ is smooth, we can divide by δ and send $\delta \to 0$. Doing so, the chain rule gives

$$\inf_{v, \omega} \left\{ u_t + \dot{\boldsymbol{x}} u_x + \dot{\boldsymbol{y}} u_y + \dot{\boldsymbol{\theta}} u_\theta \right\} = -1, \tag{8}$$

whereupon inserting (3) yields

$$u_t + \inf_{v, \omega} \left\{ \begin{aligned} &(u_x \cos\theta + u_y \sin\theta) v + \\ &W(-d u_x \sin\theta + d u_y \cos\theta + u_\theta)\omega \end{aligned} \right\} = -1. \tag{9}$$

Notice the minimization is linear in the variables $v, \omega \in [-1, 1]$, and thus the minimizing values can be resolved explicitly. We see that

$$\begin{aligned} v &= -\text{sign}(u_x \cos\theta + u_y \sin\theta), \\ \omega &= -\text{sign}(-d \sin\theta u_x + d \cos\theta u_y + u_\theta), \end{aligned} \tag{10}$$

where $u(x, y, \theta, t)$ solves the HJB equation

$$\begin{aligned} u_t &- |u_x \cos\theta + u_y \sin\theta| \\ &- W\,|-d u_x \sin\theta + d u_y \cos\theta + u_\theta| \end{aligned} = -1. \tag{11}$$

This derivation is only valid when $u(x, y, \theta, t)$ is smooth, which is not expected to be the case. However, under very general conditions, the travel time function is the unique viscosity solution of (11) [31]. For a fully rigorous derivation of the Hamilton-Jacobi-Bellman equation, see [32].

There are a few natural conditions appended to (11). At the terminal time T, the cost functional (5) assigns a value of either 0 or $+\infty$, depending on whether car is at the ending configuration or not. Thus, we have the terminal condition

$$u(x, y, \theta, T) = \begin{cases} 0, & (x, y, \theta) = (x_f, y_f, \theta_f), \\ +\infty, & \text{otherwise}, \end{cases} \tag{12}$$

and we want to resolve $u(x, y, \theta, t)$ for *preceding* times $t \in [0, T)$. So the equation runs "backwards" in time.

Likewise, if the trajectory has already arrived at the ending configuration, the remaining travel time is 0, so we have the boundary condition

$$u(x_f, y_f, \theta_f, t) = 0, \quad t \in [0, T]. \tag{13}$$

Lastly, to ensure the car does not collide with obstacles, we assign $u(x, y, \theta, t) = +\infty$ for any (x, y, θ, t) such that $C(x, y, \theta) \cap \Omega_{\text{obs}}(t) \neq \varnothing$.

By (10), the only possible values of the control variables are $v, \omega \in \{-1, 0, 1\}$, resulting in a bang-bang controller which has a "no bang" option. This makes intuitive sense because there is never incentive to drive or turn slower than the maximum possible speed, unless one needs to wait for an obstacle to move out of the way (whereupon $v = 0$) or one needs to drive in a straight line (whereupon $\omega = 0$). When no obstacles are present, one can eliminate the $v = 0$ option and the path will consist of straight lines and arcs of circles of minimum radius, which agrees with early analysis of the problem [3], [4].

As a final note, this derivation is very similar to that in [12]. However, when the obstacles are stationary, as in [10], [11], [12], the optimal travel time function does not depend on t, since the optimal trajectory depends only upon the current configuration, not upon the time t when the car occupies that configuration. In that case, one can eliminate the time horizon T, and opt instead for a stationary HJB equation. One can than visualize solving the stationary HJB equation by evolving a front outward from the final configuration, and recording the time as the front passes through other configurations, terminating when each point in the domain $\Omega \times [0, 2\pi)$ has been assigned a value. This is the philosophy behind level-set inspired optimal path planning [15], [16], and numerical implementations like fast sweeping [33], [34], [35] and fast marching methods [36], [37]. In theory, something similar is possible here. If one does not care to enforce a finite time horizon, then making the substitution $\tau = T - t$ and taking $T \to \infty$ will do away with it. However, in practice, we will want to discretize the HJB equation in order to solve computationally, which will require choosing a fixed time horizon. Thus we cannot do away with T, but to minimize its effect, we set it large enough that the travel time $u(x, y, \theta, 0)$ is finite for all $(x, y, \theta) \in \Omega \times [0, 2\pi)$. In this manner, any initial configuration (not overlapping the obstacles) will have admissable paths which reach (x_f, y_f, θ_f) within time T.

III. NUMERICAL METHODS

In this section, we design a numerical scheme to approximate (11). Since Hamilton-Jacobi equation admit non-smooth solutions which cannot be approximated by simple finite difference schemes, effort has been expended to develop schemes which resolve the viscosity solution. For a survey of numerical methods for Hamilton-Jacobi equations, see [25], [38].

A. An Upwind, Monotone Scheme for (11)

For simplicity, we confine ourselves to a rectangular spatial domain $\Omega = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. Choosing $I, J, K, N \in \mathbb{N}$, let $(x_i)_{i=0}^I, (y_j)_{j=0}^J, (\theta_k)_{k=0}^K, (t_n)_{n=0}^N$ be uniform discretizations of their respective domains with grid parameters $\Delta x, \Delta y, \Delta\theta, \Delta t$, and let u_{ijk}^n be our approximation to $u(x_i, y_j, \theta_k, t_n)$. For each $v, \omega \in \{-1, 0, 1\}$, define

$$\begin{aligned} A_k(v, \omega) &= v \cos\theta_k - \omega W d \sin\theta_k, \\ a_k(v, \omega) &= \text{sign}(v \cos\theta_k - \omega W d \sin\theta_k), \\ B_k(v, \omega) &= v \sin\theta_k + \omega W d \cos\theta_k, \\ b_k(v, \omega) &= \text{sign}(v \sin\theta_k + \omega W d \cos\theta_k). \end{aligned} \tag{14}$$

Then (8) can be rewritten

$$u_t + \min_{v, \omega} \{A_k(v, \omega) u_x + B_k(v, \omega) u_y + \omega W u_\theta\} = -1. \tag{15}$$

Recall, the terminal values u_{ijk}^N are supplied here, and we need to integrate this equation backwards in time. Thus at time step t_n, we need to resolve u_{ijk}^n given known values u_{ijk}^{n+1}. This suggests backward Euler time integration

$$(u_t)_{ijk}^n = \frac{u_{ijk}^{n+1} - u_{ijk}^n}{\Delta t}. \tag{16}$$

The upwind approximations to the other derivatives in (15) using u_{ijk}^{n+1} are given by

$$\begin{aligned}(A_k(v,\omega)u_x)_{ijk}^{n+1} &= |A_k(v,\omega)| \left(\frac{u_{i+a_k(v,\omega),j,k}^{n+1} - u_{ijk}^{n+1}}{\Delta x}\right),\\ (B_k(v,\omega)u_y)_{ijk}^{n+1} &= |B_k(v,\omega)| \left(\frac{u_{i,j+b_k(v,\omega),k}^{n+1} - u_{ijk}^{n+1}}{\Delta y}\right),\\ (\omega W u_\theta)_{ijk}^{n+1} &= |\omega|\, W \left(\frac{u_{i,j,k+\text{sign}(\omega)}^{n+1} - u_{ijk}^{n+1}}{\Delta\theta}\right).\end{aligned} \tag{17}$$

We insert these approximations in (15) to arrive at

$$\begin{aligned}u_{ijk}^n = u_{ijk}^{n+1} + \Delta t\Big(1 + \min_{v,w}\{&(A_k(v,\omega)u_x)_{ijk}^{n+1}\\ &+ (B_k(v,\omega)u_y)_{ijk}^{n+1} + (\omega W u_\theta)_{ijk}^{n+1}\}\Big).\end{aligned} \tag{18}$$

Since there are only finitely many pairs (v, ω), we can compute the right hand side for each pair and explicity choose the pair which suggests the minimum possible value. Using this formula and stepping through $n = N-1, N-2, \ldots, 1, 0$, we arrive at our approximation of the travel time function.

To initialize, we set $u_{ijk}^n = +\infty$ (or some very large number) for all i, j, k, n except at the node (i_f, j_f, k_f) respresenting the configuration nearest to (x_f, y_f, θ_f) where we set $u_{i_f,j_f,k_f}^n = 0$ for all n. We then only update the node u_{ijk}^n if the value suggested by (18) is smaller than the value already stored at u_{ijk}^n. This ensures that the scheme is monotone so long as the Courant-Friedrichs-Lewy condition

$$\Delta t\left(\frac{1+Wd}{\Delta x} + \frac{1+Wd}{\Delta y} + \frac{W}{\Delta\theta}\right) \le 1 \tag{19}$$

is satisfied [25], [38]. In this case, since the scheme is also consistent, the approximation converges to the viscosity solution of (11) as $\Delta x, \Delta y, \Delta\theta, \Delta t \to 0$.

We include a few implementation notes. First, to account for obstacles, at each time step n, we first need to find the illegal nodes (i.e. those which correspond to configurations wherein the car collides with an obstacle). At these nodes (i^*, j^*, k^*), we do not use (18), but rather set $u_{i^*,j^*,k^*}^n = +\infty$. In previous work, this could be done in pre-processing since the obstacles were stationary and illegal configurations only needed to be resolved once. In this work, since the obstacles move, this must be repeated at every time step. We note that a node may be illegal at one time step (hence given a large value) and free at the next. This causes no issues: when the node becomes free again, its value will be re-computed from nearby nodes as described above. Second, we use (18) for $i = 2, \ldots, I-1, j = 2, \ldots, J-1$. The values u_{ijk}^n at nodes corresponding to the spatial boundary are never updated, but should be given the value $+\infty$. This will ensure that the car never leaves the domain. Because we enforce the correct causality, the boundary nodes have no effect on interior nodes. Third, one needs to enforce periodic boundary conditions in θ by identifyting the nodes at $k = 0$ and $k = K$. Lastly, above it is stated that $v, \omega \in \{-1, 0, 1\}$. However, because it is impossible to turn a car without moving backward or forward, one should eliminate the cases $(v, \omega) = (0, \pm 1)$. So there are seven possible pairs of (v, ω) to consider in total; in short, $(v, \omega) = (\pm 1, \pm 1), (\pm 1, 0), (0, 0)$.

B. Generating Optimal Trajectories

There are a few different manners in which one can obtain optimal control values and generate optimal trajectories. It is possible to resolve control values $v_{ijk}^n, \omega_{ijk}^n$ while evaluating (18). One can define them to be the pair that achieves the minimum in (18) at any node (i, j, k, n). Alternatively, after resolving u_{ijk}^n, one can interpolate to off-grid values and use (10) to resolve the optimal steering plan at any point (x, y, θ, t). In either case, after choosing an initial point, one can insert the optimal control values into (3) and integrate the equations of motion until the trajectory reaches (x_f, y_f, θ_f). This is the approach taken by [12].

In a different approach, we opt for a semi-Lagrangian path-planner as in [11], [14]. Specifically, we first interpolate u_{ijk}^n to off grid values, so we have an approximate travel time function $u(x, y, \theta, t)$. Then, choosing an initial point $(\boldsymbol{x}_0, \boldsymbol{y}_0, \boldsymbol{\theta}_0)$ and a time step $\delta > 0$, and rewriting (3) as $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\boldsymbol{\theta}}) = F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{v}, \boldsymbol{\omega})$, we set

$$\begin{aligned}(v^*, \omega^*) &= \operatorname*{argmin}_{v,\omega} u((\boldsymbol{x}_\ell, \boldsymbol{y}_\ell, \boldsymbol{\theta}_\ell) + \delta F(\boldsymbol{x}_\ell, \boldsymbol{y}_\ell, \boldsymbol{\theta}_\ell, v, \omega), \ell\delta),\\ (\boldsymbol{x}_{\ell+1}, \boldsymbol{y}_{\ell+1}, \boldsymbol{\theta}_{\ell+1}) &= (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell, \boldsymbol{\theta}_\ell) + \delta F(\boldsymbol{x}_\ell, \boldsymbol{y}_\ell, \boldsymbol{\theta}_\ell, v^*, \omega^*),\end{aligned} \tag{20}$$

for $\ell = 0, 1, 2, \ldots$, halting when $(\boldsymbol{x}_\ell, \boldsymbol{y}_\ell, \boldsymbol{\theta}_\ell)$ is within some tolerance of (x_f, y_f, θ_f).

IV. RESULTS & EXAMPLES

We present results of our algorithm in three examples. In all cases, we use the spatial domain $\Omega = [-1, 1] \times [-1, 1]$. We take the car to be a rectangles as pictured in fig. 1 with $d = 0.07$ and $R = 0.04$ and we take the maximum angular velocity to be $W = 4$. These are dimensionaless variables used for testing purposes. In each of the following pictures the final configuration (x_f, y_f, θ_f) will be marked with a red star and the initial configurations of the various cars will be marked with green stars. We use a $101 \times 101 \times 101$ discretization of $\Omega \times [0, 2\pi)$ and then choose Δt according to (19). We choose the time horizon $T = 10$. As mentioned before, this simply needs to be chosen so that there are admissable paths from every point on the domain to the final configuration which take time less than T to traverse. In some of the examples it could likely be smaller, but $T = 10$ was sufficiently large for all of them.

In the first example, the final configuration is $(x_f, y_f, \theta_f) = (0, 0, \pi)$ meaning the cars will end at

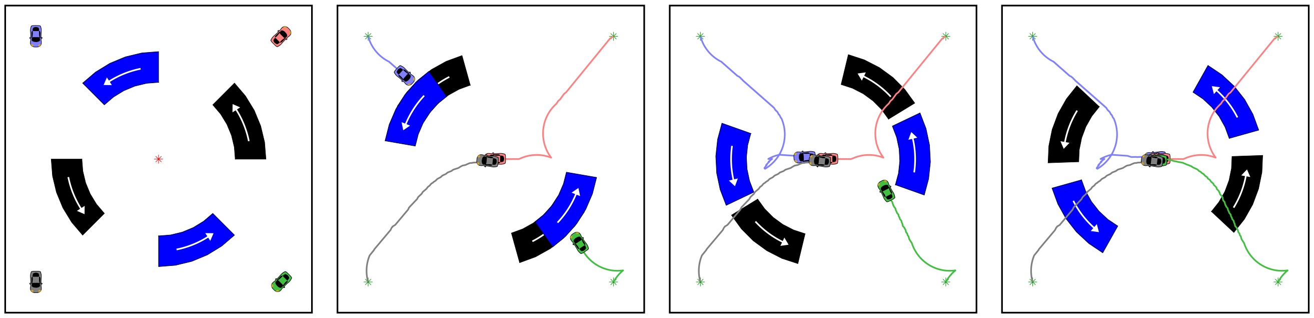

Fig. 2. In the first example, we have several cars starting in the corners and ending in the center facing due west. The obstacles (black and blue sectors) rotate counterclockwise with black obstacles rotating at three times the speed of the blue ones.

the center of the domain, facing due west. In this case, the obstacles $\Omega_{\text{obs}}(t)$ are 4 annular sectors which will rotate about the origin in the counterclockwise direction. These are represented in black and blue in fig. 2. The black obstacles rotate with 3 times the speed of the blue obstacles. Notice that in the second panel, the green and blue cars (respectively bottom right and top left corners) need to stop and wait to let the obstacles pass before completing their path. The grey and pink cars are essentially unoccluded and can travel directly to the destination. We note that these paths are generated individually, and simply plotted over each other. There is no competition between the cars.

In the second example, the final configuration is $(x_f, y_f, \theta_f) = (0.8, 0.8, \pi/4)$ so that the car needs to end near the top right corner of the domain facing northeast. The car begins in the bottom left corner of the domain as seen in fig. 3, and must navigate through three moving doorways. The black bars represent the obstacles and they oscillate as indicated by the arrows. The car is able to navigate through the domain without stopping to wait for the doors.

In the third example, we consider the more realistic scenario of a car changing lanes in between two other cars as seen in fig. 4. In this case the two blue cars are treated as obstacles and the orange car must slide in between them.

V. CONCLUSION & DISCUSSION

We present a Hamilton-Jacobi-Bellman formulation for time-optimal paths of simple vehicles in the presence of moving obstacles. This is distinguished from previous similar formulations which could only handle stationary obstacles.

There are many ways in which this work could be extended. Some simple improvements would be to account for other realistic concerns such as energy minimization or instrumentation noise, which can be added to the model in a straightforward manner, but may complicate the numerics.

Perhaps the biggest drawback of this method is that it is currently too computationally intensive for real-time applications. The simulations for each of the examples in section IV required several minutes of CPU time (on the authors' home computers). However, one may be able to apply recent methods for high-dimensional Hamilton-Jacobi equations [39], [40]. These methods are based on Hopf-Lax type formulas and trade finite differences for optimization problems. It may be difficult to account for crucial boundary conditions in our model when using such schemes, so some care would be required. However, if they could be applied to this problem, it would also provide an opportunity to extend the model to higher dimensions where finite difference methods are infeasible.

ACKNOWLEDGMENT

The authors were supported in part by NSF DMS-1937229 through the Data Driven Discovery Research Training Group at the University of Arizona.

REFERENCES

[1] K. K. Leung, C. H. Hsieh, Y. R. Huang, A. Joshi, V. Voroninski, and A. L. Bertozzi, "A second generation micro-vehicle testbed for cooperative control and sensing strategies," in *2007 American Control Conference*, pp. 1900–1907, 2007.
[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
[3] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.
[4] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards.," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, 1990.
[5] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, no. 2-4, p. 121, 1993.
[6] P. K. Agarwal and H. Wang, "Approximation algorithms for curvature-constrained shortest paths," *SIAM Journal on Computing*, vol. 30, no. 6, pp. 1739–1772, 2001.
[7] J.-D. Boissonnat, A. Cérézo, and J. Leblond, "Shortest paths of bounded curvature in the plane," *Journal of Intelligent and Robotic Systems*, vol. 11, no. 1, pp. 5–20, 1994.
[8] X.-N. Bui, J.-D. Boissonnat, P. Soueres, and J.-P. Laumond, "Shortest path synthesis for Dubins non-holonomic robot," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 2–7, IEEE, 1994.
[9] X.-N. Bui and J.-D. Boissonnat, "Accessibility region for a car that only moves forwards along optimal paths," tech. rep., 1994.
[10] R. Takei, R. Tsai, H. Shen, and Y. Landa, "A practical path-planning algorithm for a simple car: a Hamilton-Jacobi approach," in *Proceedings of the 2010 American Control Conference*, pp. 6175–6180, June 2010.
[11] R. Takei and R. Tsai, "Optimal trajectories of curvature constrained motion in the Hamilton-Jacobi formulation," *Journal of Scientific Computing*, vol. 54, pp. 622–644, Feb 2013.
[12] C. Parkinson, A. L. Bertozzi, and S. J. Osher, "A Hamilton-Jacobi formulation for time-optimal paths of rectangular nonholonomic vehicles," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 4073–4078, IEEE, 2020.
[13] T. Lolla, M. P. Ueckermann, K. Yiğit, P. J. Haley, and P. F. Lermusiaux, "Path planning in time dependent flow fields using level set methods," in *2012 IEEE International Conference on Robotics and Automation*, pp. 166–173, IEEE, 2012.

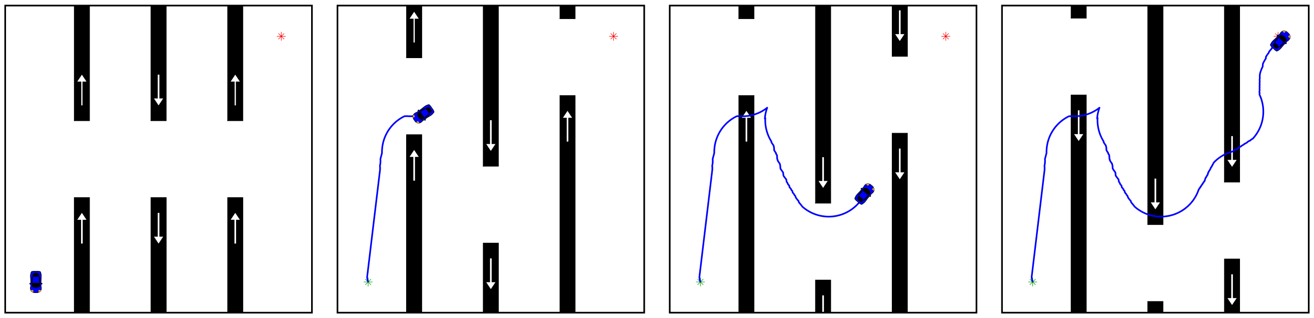

Fig. 3. In the second example, one care attempts to navigate through moving doorways. The black obstacles oscillate up and down as indicated by the arrows in each panel.

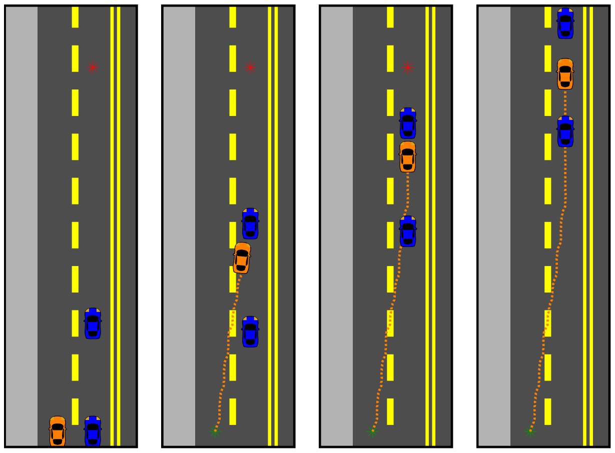

Fig. 4. A car (orange) changing lanes between two other cars (blue). Here the blue cars are the obstacles.

[14] E. Cartee, L. Lai, Q. Song, and A. Vladimirsky, "Time-dependent surveillance-evasion games," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 7128–7133, IEEE, 2019.

[15] C. Parkinson, D. Arnold, A. L. Bertozzi, Y. T. Chow, and S. Osher, "Optimal human navigation in steep terrain: a Hamilton-Jacobi-Bellman approach," *Communications in Mathematical Sciences*, vol. 17, no. 1, pp. 227–242, 2019.

[16] C. Parkinson, D. Arnold, A. Bertozzi, and S. Osher, "A model for optimal human navigation with stochastic effects," *SIAM Journal on Applied Mathematics*, vol. 80, no. 4, pp. 1862–1881, 2020.

[17] D. J. Arnold, D. Fernandez, R. Jia, C. Parkinson, D. Tonne, Y. Yaniv, A. L. Bertozzi, and S. J. Osher, "Modeling environmental crime in protected areas using the level set method," *SIAM Journal on Applied Mathematics*, vol. 79, no. 3, pp. 802–821, 2019.

[18] E. Cartee and A. Vladimirsky, "Control-theoretic models of environmental crime," *SIAM Journal on Applied Mathematics*, vol. 80, no. 3, pp. 1441–1466, 2020.

[19] B. Chen, K. Peng, C. Parkinson, A. L. Bertozzi, T. L. Slough, and J. Urpelainen, "Modeling illegal logging in Brazil," *Research in the Mathematical Sciences*, vol. 8, no. 2, pp. 1–21, 2021.

[20] A. Shukla, E. Singla, P. Wahi, and B. Dasgupta, "A direct variational method for planning monotonically optimal paths for redundant manipulators in constrained workspaces," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 209–220, 2013.

[21] R. Gao, X. Gao, P. Liang, F. Han, B. Lan, J. Li, J. Li, and S. Li, "Motion control of non-holonomic constrained mobile robot using deep reinforcement learning," in *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 348–353, IEEE, 2019.

[22] J. J. Johnson, L. Li, F. Liu, A. H. Qureshi, and M. C. Yip, "Dynamically constrained motion planning networks for non-holonomic robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6937–6943, IEEE, 2020.

[23] E. J. Rodríguez-Seda, C. Tang, M. W. Spong, and D. M. Stipanović, "Trajectory tracking with collision avoidance for nonholonomic vehicles with acceleration constraints and limited sensing," *The International Journal of Robotics Research*, vol. 33, no. 12, pp. 1569–1592, 2014.

[24] R. Mao, H. Gao, and L. Guo, "A novel collision-free navigation approach for multiple nonholonomic robots based on ORCA and linear MPC," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[25] M. Falcone and R. Ferretti, "Numerical methods for Hamilton–Jacobi type equations," in *Handbook of Numerical Methods for Hyperbolic Problems* (R. Abgrall and C.-W. Shu, eds.), vol. 17 of *Handbook of Numerical Analysis*, pp. 603 – 626, Elsevier, 2016.

[26] W. H. Fleming and R. W. Rishel, *Deterministic and stochastic optimal control*, vol. 1. Springer Science & Business Media, 2012.

[27] D. P. Bertsekas, "Dynamic programming and optimal control 3rd edition, volume ii," *Belmont, MA: Athena Scientific*, 2011.

[28] E. N. Moret, *Dynamic modeling and control of a car-like robot*. PhD thesis, Virginia Tech, 2003.

[29] B. Triggs, "Motion planning for nonholonomic vehicles: An introduction," 1993.

[30] R. Bellman, *Dynamic Programming*. RAND Corporation research study, Princeton University Press, 1957.

[31] M. G. Crandall and P.-L. Lions, "Viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 277, no. 1, pp. 1–42, 1983.

[32] M. Bardi and I. Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Modern Birkhäuser Classics, Birkhäuser Boston, 2008.

[33] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao, "Fast sweeping algorithms for a class of Hamilton–Jacobi equations," *SIAM journal on numerical analysis*, vol. 41, no. 2, pp. 673–694, 2003.

[34] C. Y. Kao, S. Osher, and J. Qian, "Lax–Friedrichs sweeping scheme for static Hamilton–Jacobi equations," *Journal of Computational Physics*, vol. 196, no. 1, pp. 367–391, 2004.

[35] C. Parkinson, "A rotating-grid upwind fast sweeping scheme for a class of Hamilton-Jacobi equations," *Journal of Scientific Computing*, vol. 88, no. 1, pp. 1–36, 2021.

[36] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control*, vol. 40, pp. 1528–1538, Sep 1995.

[37] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[38] S. Osher and R. P. Fedkiw, *Level set methods and dynamic implicit surfaces*, vol. 153 of *Applied Mathematical Sciences*. Springer–Verlag, 2003.

[39] J. Darbon and S. Osher, "Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere," *Research in the Mathematical Sciences*, vol. 3, no. 1, pp. 1–26, 2016.

[40] A. T. Lin, Y. T. Chow, and S. J. Osher, "A splitting method for overcoming the curse of dimensionality in Hamilton–Jacobi equations arising from nonlinear optimal control and differential games with applications to trajectory generation," *Communications in Mathematical Sciences*, vol. 16, 1 2018.