A Multi-Agent Formulation of the Optimal Path Planning Problem for Autonomous Vehicles

Edward Huynh, Christian Parkinson, Misha Chertkov

Abstract

We considered the problem of navigating a car from an initial point to its final destination, constrained to both dynamics and physical constraints. We cast this problem (along with stationary obstacles, moving obstacles, and other cars) in the optimal control framework subsequently derived the HJB equation. Using an algorithm suggested in [1], we generate optimal trajectories based on the HJB equation on the order of seconds. The novelty of our work is in proposing a new multi-agent model to simulate a network of vehicles that seek to navigate to their destinations while avoiding collisions.

1 Introduction

The increasing relevance of autonomous vehicles has led to developments in technology to maneuver them to their respective destinations. This has applications in ground [2], marine [3], aerial [4], and space [5] robotics motion planning. Several considerations need to be made in doing so such as energy consumption, collision costs, and the ability to navigate to a destination in shortest time. The basic mathematical formulation of the problem includes an agent whose movement is constrained by an equation of motion, and who desires to optimally travel from one point to another while obeying this equation. Optimality is defined in this case as minimizing a particular cost functional. In some contexts, it is of great interest to navigate multiple vehicles at the same time while addressing the above considerations [6]. In this paper, we consider this problem in the context of optimal path planning.

Dubins [7] and Reeds-Shepps [8] cars are simple models for modeling vehicle motion, and their paths are subject to curvature constraints. The former models cars with forward moving motion while the latter deals with vehicles that may move both forward and backwards. The first work to applying the optimal path planning approach to determine optimal paths for these models goes back to the work of Takei et. al. [9] and later extended in [10] . In this work, they used dynamic programming to derive the Hamilton-Jacobi formulation for minimal time paths with curvature constraints and solved the equation using level-set methods. Their work also discussed curvature-constrained motion on manifolds and the presence of obstacles. Since then, this work was expanded upon by several authors to include both stationary and moving obstacles [11, 12, 13]. Aside from cars, [10] also considers the implementation of Dubins planes and submarines.

Among the approaches to solve Hamilton-Jacobi equations, important numerical methods include the level-set method, fast-sweeping, and fast-marching schemes [14, 15, 16]. Fast-marching schemes in particular were extended to solve eikonal equations and several types of Hamilton-Jacobi equations [17, 18]. These methods rely on discretizing the domain

that the PDE is defined on and implementing finite difference schemes to estimate derivatives. However, the cost in approximating the solution in these ways scale exponentially in time and spatial dimensions. This problem is further compounded when we consider possible multi-car models. Thus, solutions obtained for these are typically near-optimal and may lead to many solutions. In recent years, due to recent developments in applications involving decision-making and game theory, computing the solution to the equation at certain points while breaking the curse of dimensionality has attracted immense interest. In [19], a numerical solution to the equation was obtained using the Hopf-Lax formula. A (conjectured) extension of the Hopf-Lax formula is proposed in [20] and several methods [20, 1] are proposed to solve more general Hamilton-Jacobi equations. In the former, they use coordinate descent along with finite differences to resolve the solution. In contrast, the latter is based off of the PDHG (Primal-Dual-Hybrid-Gradient, also known as Chambolle-Pock) algorithm [21].

In this paper, we propose a new formulation for the multi-agent scenario using Hamilton-Jacobi-Bellman equations. Our approach is based on differential game and optimal control theory by devising solution through the equations. Optimal trajectories can be found efficiently and quickly using a splitting method algorithm found in [1]. Moreover, to avoid intersections of the trajectories in real-time, we impose a collision avoidance constraint. By solving the optimal control problem using the HJB equation, we avoid the curse of dimensionality and can compute optimal trajectories for multiple vehicles efficiently. This paper is organized as follows. In section 2, we derive the model along with the mathematical preliminaries. In section 3, we discuss the numerical approach and propose an algorithm which we used to solve the equations. In section 4, we present and discuss the results of our simulations.

2 Mathematical Formulation

2.1 Preliminaries

We begin by phrasing our problem in the framework of optimal control. Suppose an agent has an initial position given by $\boldsymbol{x}_0 \in \mathbb{R}^n$ and desires to navigate to $\boldsymbol{x}_f \in \mathbb{R}^n$. Let $T > 0$ denote a finite time horizon, $f : \mathbb{R}^n \times [0,T] \times \mathbb{R}^k \to \mathbb{R}^n$ be a function, and $\boldsymbol{x} : [0,T] \to \mathbb{R}^n$ denote the trajectory of the agent which satisfies

$$\begin{aligned} \dot{\boldsymbol{x}} &= f(\boldsymbol{x}, t, \boldsymbol{\alpha}), \quad 0 \le t \le T, \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0. \end{aligned}$$

We let $\boldsymbol{\alpha} \in \mathcal{A}$ denote the control, and $\mathcal{A}$ is the set of admissible control maps. Let $\mathcal{C}([0,T], \mathbb{R}^n)$ denote the set of continuous maps from the time interval $[0,T]$ to $\mathbb{R}^n$. Suppose the vehicle has a cost function $C : \mathcal{C}([0,T], \mathbb{R}^n) \times \mathcal{A} \to \mathbb{R}$

$$C(\boldsymbol{x}, \boldsymbol{\alpha}) = g(\boldsymbol{x}(T)) + \int_0^T L(\boldsymbol{x}(t), t, \boldsymbol{\alpha}(t))\, dt$$

where $L : \mathbb{R}^n \times [0,T] \times \mathbb{R}^k \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$ denote the running and terminal costs respectively. Then the goal is to solve the optimization problem

$$\inf_{\boldsymbol{\alpha} \in \mathcal{A}} C(\boldsymbol{x}, \boldsymbol{\alpha}).$$

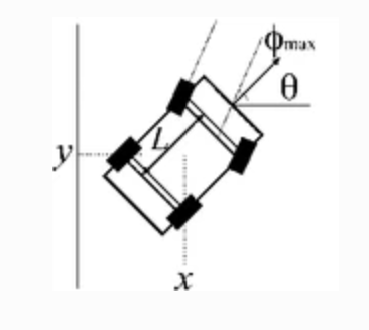


Figure 1: Dubins Car

2.2 Reed-Shepps Vehicle Dynamics

Suppose a vehicle C has initial position $(x_0, y_0) \in \mathbb{R}^2$ with initial angle θ_0 measured counterclockwise from the positive x-axis desires to navigate to a final position (x_f, y_f) with ending orientation θ_f. We say $\boldsymbol{x}_0 = (x_0, y_0, \theta_0)$ and $\boldsymbol{x}_f = (x_f, y_f, \theta_f)$ denote the initial and final poses of the vehicle respectively. Let $\Omega_{free} \subset \mathbb{R}^2$ denote the free space that the vehicle may navigate and $\Omega_{obs} \subset \mathbb{R}^2$ denote the obstacle space. Then the domain for the problem is $\Omega = \Omega_{free} \cup \Omega_{obs}$. In the case of a Reeds-Shepps car, we have that the vehicle satisfies

$$\begin{aligned} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \phi. \end{aligned}$$

Here the $\boldsymbol{\alpha} = (v, \phi)$ are the velocity and the angular velocity of the vehicle. We also have $v \in [1, -1]$ and the nonholonomic constraint $|\dot{\theta}| = |\phi| \leq \frac{1}{\rho_{min}} = W$, where ρ_{min} denotes the minimal turning radius. In addition, since we assume that the car is a point mass then we have the additional (nonholonomic) constraint $\dot{y}\cos\theta - \dot{x}\sin\theta = 0$.

Since the vehicle desires to reach its destination in minimal time, then we seek to minimize the following:

$$C(\boldsymbol{x}, \boldsymbol{\alpha}) = g(T) + \int_0^T L(\boldsymbol{x}, t, \boldsymbol{\alpha})\ dt = t^*,$$

where t^* denotes the first time where the vehicle reaches $\boldsymbol{x}_f$, and the optimization is taken over all curves $\boldsymbol{x}(\tau)$, $\tau \in (0, T]$ with $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, t, \boldsymbol{\alpha})$ and $\boldsymbol{\alpha} = (v, \phi) \in \mathcal{A}$. Here, letting $A = [-1, 1] \times [-W, W]$ denote the control values, then $\mathcal{A} = \{\boldsymbol{\alpha} : \boldsymbol{\alpha} : [0, T] \to A\}$.

2.3 Derivation of the Hamilton-Jacobi-Bellman Equation

We next formally derive the HJB equation under this model. We define the value function:

$$u(\boldsymbol{x}, t) = \inf_{\boldsymbol{\alpha}(\tau),\ 0 \leq \tau \leq t} t^*$$

We apply the dynamic programming principle [22] :

$$u(\boldsymbol{x}(t), t) = \delta + \inf_{\substack{\boldsymbol{\alpha}(\tau) \in A \\ \tau \in [t, t+\delta)}} u(\boldsymbol{x}(t+\delta), t+\delta).$$

We subtract u from each side and divide by δ to obtain

$$\inf_{\substack{\boldsymbol{\alpha}(\tau)\in A \\ \tau\in[t,t+\delta)}} \frac{u(\boldsymbol{x}(t+\delta),t+\delta)-u(\boldsymbol{x}(t),t)}{\delta}+1=0.$$

Assuming that the value function is differentiable ,then letting $\delta\to 0$ yields

$$\inf_{\boldsymbol{\alpha}(t)\in A} u_t+\langle\nabla u,\dot{\boldsymbol{x}}\rangle=-1.$$

Substituting in the Reeds-Shepps dynamics yields

$$u_t+\inf_{\boldsymbol{\alpha}(t)\in A}\left(u_x v\cos(\theta)+u_y v\sin(\theta)+u_\theta\phi\right)=-1.$$

Then the optimal control is given by $v=-\text{sign}(u_x\cos(\theta)+u_y\sin(\theta))$ and $\phi=-\text{sign}(u_\theta)W$ so that

$$u_t-|u_x\cos(\theta)+u_y\sin(\theta)|-W|u_\theta|=-1.$$

and u satisfies the terminal condition

$$u(\boldsymbol{x},T)=0.$$

It has been shown that if a control is constructed such that it is the minimum value found in the HJB equation, then it is optimal and solves the optimization problem. This follows from the verification theorem [23]. Thus, by solving the HJB equation and using the formulas for (v,ϕ), we are thereby able to find $\boldsymbol{\alpha}$.

We also need to assume additional boundary conditions which arise from physical constraints in the problem. We assume that T is a large enough time horizon such that the vehicle may navigate to $\boldsymbol{x}_f$ by time T. If the vehicle can reach $\boldsymbol{x}_f$ at time $T^*<T$, then we let $u(\boldsymbol{x}_f,t)=0$ for all $T^*<t\leq T$. On the other hand, if $\boldsymbol{x}$ is a position such that the car cannot reach $\boldsymbol{x}_f$ from $\boldsymbol{x}$ on this interval, then $u(\boldsymbol{x},t)=+\infty$. If the vehicle is already at its terminal destination at $t=0$, then we set $u(\boldsymbol{x}_f,t)=0$ for all $t\geq 0$, since the minimal time to reach the terminal destination is 0. We also note that since we would like the vehicles to avoid choosing paths that pass through obstacles, we let $u(\boldsymbol{x},t)=+\infty$ for $\boldsymbol{x}\in\Omega_{obs}$. This is the same thing as assigning infinite cost to any trajectory that hits an obstacle.

2.4 Reformulation

To solve this problem more efficiently, we desire to remove the necessity on auxiliary boundary conditions as described in the previous section, so this leads us to redefine

$$u(x,t)=\inf_{\boldsymbol{\alpha}}\left\{\text{distance from } (x(T),y(T),\theta(T)) \text{ to } (x_f,y_f,\theta_f) \text{ given that the car is at } (x,y,\theta) \text{ at time } t\right\}.$$

That is, we seek to minimize

$$\mathcal{C}(\boldsymbol{x},\boldsymbol{\alpha})=|\boldsymbol{x}(T)-\boldsymbol{x}_f|.$$

Define the value function

$$u(x,t)=\inf_{\boldsymbol{\alpha}\in\mathcal{A}}\mathcal{C}(\boldsymbol{x},\boldsymbol{\alpha})$$

Then by dynamic programming since the cost function consists of only a terminal cost, then we have

$$u(\boldsymbol{x}(t), t) = \inf_{\substack{\boldsymbol{\alpha}(\tau)\in A \\ \tau\in[t,t+\delta)}} u(\boldsymbol{x}(t+\delta), t+\delta).$$

A similar derivation to the one found in the previous section shows that u satisfies

$$\inf_{\boldsymbol{\alpha}(t)\in\mathcal{A}} (u_t + \langle \nabla u, \dot{\boldsymbol{x}} \rangle) = 0.$$

Substituting the dynamics $\dot{\boldsymbol{x}}$ yields

$$u_t + \inf_{\boldsymbol{\alpha}(t)\in A} (v u_x \cos(\theta) + v u_y \sin(\theta) + \phi u_\theta) = 0.$$

This has the terminal condition $u(\boldsymbol{x}, T) = |\boldsymbol{x}(T) - \boldsymbol{x}_f|$.

To use our proposed numerical method, we will convert this problem into an initial-valued problem. We let $t \mapsto T - t$ (and abusing notation by letting $T - t := t$) so that we obtain

$$u_t - \inf_{\boldsymbol{\alpha}(t)\in\mathcal{A}} \{v u_x \cos(\theta) + v u_y \sin(\theta) + \phi u_\theta\} = 0$$

$$u(\boldsymbol{x}, 0) = |\boldsymbol{x} - \boldsymbol{x}_f|.$$

We find the optimal controls: $v = -\text{sign}(u_x \cos(\theta) + u_y \sin(\theta))$ and $\phi = -W \text{sign}(u_\theta)$. This yields the equation

$$u_t + |u_x \cos(\theta) + u_y \sin(\theta)| + W|u_\theta| = 0$$

with initial condition $u(\boldsymbol{x}, 0) = |\boldsymbol{x}(0) - \boldsymbol{x}_f|$.

We note that this has the form of the Hamilton-Jacobi equation:

$$u_t + H(\boldsymbol{x}, \nabla u) = 0,$$

where $H(x, p) = |p_1 \cos(\theta) + p_2 \sin(\theta)| + W|p_3|$ with initial condition $u(x, 0) = g(x)$. Firstly, in terms of modeling, the reformulation considers minimizing the distance of the car to the final point as opposed to the travel time. Therefore, optimal paths from a point x are chosen such that the distance between the terminal point on this trajectory and the desired destination is minimal. This formulation leads to a solution that is equivalent to the "time-optimal" one in the following sense: if T (the time horizon) is the minimal time that the car may reach $\boldsymbol{x}_f$, then the controls obtained from solving the above equation is exactly the same as the one for the time-optimal formulation. This formulation also avoids prescribing boundary conditions: if $\boldsymbol{x} \in \Omega_{obs}$, we set $v == 0$ so that the vehicle will never move inside an obstacle. In this way, an optimal trajectory will never pass through an obstacle (although we allow being tangent to $\partial\Omega_{obs}$, the boundary of an obstacle).

2.5 Obstacle Formulation

In order to take care of obstacles, we instead consider multiplying the Hamiltonian by an indicator function [13], i.e. we solve

$$u_t + \mathbb{1}_{\Omega_{free}}(x, y) H(\boldsymbol{x}, \nabla u) = 0.$$

If $\boldsymbol{x} \in \Omega_{obs}$, then we set $v == 0$ so that the car does not move in an obstacle. In the equation, we consequently obtain $u_t = 0$, which implies the function is constant in an obstacle. Therefore, a car will have not lessen their value within an obstacle and will be discouraged from moving into one.

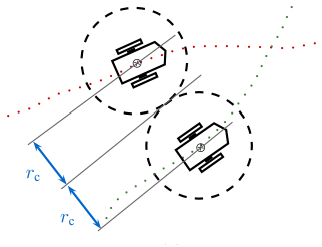


Figure 2: Security Clearance

2.6 The Multi-Agent Formulation

We will now describe the multi-vehicle formulation. Suppose we have K vehicles, each with respective initial and final destinations $\boldsymbol{x}_0^k$ and $\boldsymbol{x}_f^k$, $k = 1, \ldots, K$. We suppose each vehicle follows Reeds-Shepps dynamics and that each vehicle desires to navigate to their destinations within some finite time horizon T (which is the same for all). Each trajectory induces a cost:

$$C^k(\boldsymbol{x}, \boldsymbol{\alpha}) = |\boldsymbol{x}^k(T) - \boldsymbol{x}_f^k|.$$

Then each vehicle seeks to minimize their cost function, where the minimum must satisfy the boundary value problem

$$\begin{gathered} u_t^i + |u_x^i \cos(\theta) + u_y^i \sin(\theta)| + W|u_\theta^i| = 0, \quad 0 \le t \le T, \\ u^i(\boldsymbol{x}^i, 0) = |\boldsymbol{x}^i(T) - \boldsymbol{x}_f^i|. \end{gathered}$$

At the same time, we also wish to make sure each vehicle reaches their target without "colliding" with another vehicle. So we impose a constraint:

$$||\boldsymbol{x}^i(t) - \boldsymbol{x}^j(t)|| \ge 2r_c, \quad t \in [0, T], \quad i, j = 1, \ldots, K, \quad i \ne j,$$

where r_c is the radius that defines a security clearance for each vehicle. Through this constraint, the respective equations for each vehicle are coupled.

Thus, we must solve the problem

$$\begin{gathered} u_t^i + |u_x^i \cos(\theta) + u_y^i \sin(\theta)| + W|u_\theta^i| = 0, \quad 0 \le t \le T, \quad i = 1, \ldots, K, \\ u^i(\boldsymbol{x}^i, 0) = |\boldsymbol{x}^i(T) - \boldsymbol{x}_f^i|, \\ ||\boldsymbol{x}^i(t) - \boldsymbol{x}^j(t)|| \ge 2r_c, \quad t \in [0, T], \quad i, j = 1, \ldots, K, \quad i \ne j. \end{gathered}$$

We note that a solution to this problem can be described in several ways, but we use the concept of Nash equilibrium from game theory. A set of optimal controls (or strategies) $(\boldsymbol{\alpha}_1^*, \ldots, \boldsymbol{\alpha}_K^*)$ is called a Nash equilibrium if

$$C^i(\boldsymbol{\alpha}_i^*, \boldsymbol{\alpha}_{-i}^*) \le C^i(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_{-i}^*), \quad \forall \boldsymbol{\alpha}_i \in \mathbb{R}^k, \ i = 1, \ldots, K,$$

where $\boldsymbol{\alpha}_{-i}$ denotes the $(K-1)$-tuple $(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{i-1}, \boldsymbol{\alpha}_{i+1}, \ldots, \boldsymbol{\alpha}_K)$. We believe that the controls must satisfy these relations in the multi-agent setting. In our simulation of the two-agent case, we obtain a solution for both cars, but it is unclear (at the present) whether the optimal paths generated arose from controls that satisfy Nash equilibrium.

In contrast to [6], we enforce the inequality constraint in the solution of the system by implementing the moving obstacles framework given in [13]. If we let $\Omega_{free}(t)$ denote the free space at time t, then by solving

$$u_t + \mathbb{1}_{\Omega_{free}(t)}(x, y, t)H(\boldsymbol{x}, \nabla u) = 0,$$

we obtain the optimal trajectories with moving obstacles. We note that this is slightly different from the previous framework on stationary obstacles by allowing the indicator function to vary with time.

3 Numerical Methods

Prior to [13], the standard approach to solving these equations has been to use grid-based methods such as level-set, fast-marching, and fast-sweeping schemes. However, these methods scale poorly with dimension and thus suffer the curse of dimensionality. Moreover, these methods solve the equation over the entire domain and require mesh sizes that satisfy the CFL (Courant-Friedrichs-Lewy) condition. For spatial dimensions ≥ 3, the computation is generally infeasible. In contrast, in lower dimensions obtaining a solution may take on the order of minutes, which is impractical for real-time applications.

Often in real-time applications, we are not interested in solving the equation over the whole space, but instead finding the optimal trajectory starting from a particular initial point. Therefore, we will return and solve the optimal control problem, but our method will use some aspects of the HJB equation For this, we will use a novel splitting method introduced by [1] to efficiently compute the value function on these set of paths. This method is based on the conjectured Hofp-Lax formula [20]. For the equation

$$u_t + H(\boldsymbol{x}, t, \nabla u) = 0, \quad (\boldsymbol{x}, t) \in \mathbb{R}^n \times (0, \infty)$$

with boundary condition $u(x, T) = 0$, then the generalized Hofp-Lax formula is given by

$$u(x,t) = \inf_{\alpha \in \mathbb{R}} \left\{ g(x(0)) + \int_0^t \left[\langle p(s), \nabla_p H(x(s), p(s)) \rangle - H(x(s), p(s)) \right] ds \right\}$$

such that $x(t) = x$, $p(t) = y$, $\dot{x}(s) = \nabla_p H(x(s), p(s))$, and $\dot{p}(s) = -\nabla_x H(x(s), p(s))$.

We first start by describing the numerical approach to solving for the path trajectories for an individual HJB equation in the moving obstacles regime. This approach follows from [13] and is based on the work of [1].

We start by discretizing the time domain: $t_0 = t_1 < t_2 < \ldots < t_N = t$. Next we discretize the trajectories using Backward-Euler:

$$x(t_j) \approx x(t_{j-1}) + \Delta(t_{j-1}) + \Delta t \nabla_p H(x(t_{j-1}), p(t_{j-1})).$$

We summarize the splitting method algorithm used by [13] to solve for path trajectories and which was adapted from [1] in Algorithm 1

Algorithm 1 Splitting Method

Given a point $(x,t) \in \mathbb{R}^d \times (0,t)$, a Hamiltonian H, an initial data function g, a time-discretization count N, a max iteration count K, an error tolerance TOL, and relaxation parameters $\sigma,\ \tau,\ \kappa > 0$, we solve the optimal control problem as follows:
Set $x_N^1 = x$, $p_0^1 = 0$, and $\delta = t/N$. Initialize $x_0^1, x_1^1, \ldots, x_{N-1}^1, p_1^1, \ldots, p_N^1$ randomly, and set $z_j^1 = x_j^1$ for all $j = 0, 1, \ldots, N$.

for k = 1 to K **do**
  Set $p_0^{k+1} = 0$
  for j = 1 to N **do**
    $p_j^{k+1} = arg\ min_{\tilde{p}}\{\delta\sigma H(x_j^k, \tilde{p}) + \frac{1}{2}||\tilde{p} - (p_j^k + \sigma(z_j^k - z_{j-1}^k))||_2^2\}$
  end for
  $x_0^{k+1} = arg\ min_{\tilde{x}}\{\tau g(\tilde{x}) + \frac{1}{2}||\tilde{x} - (x_0^k + \tau p_1^{k+1})||_2^2$ (note: $p_0^{k+1} = 0$)
  for j = 1 to $N-1$ **do**
    $x_j^{k+1} = arg\ min_{\tilde{x}}\{-\delta\tau H(\tilde{x}, p_j^{k+1}) + \frac{1}{2}||\tilde{x} - (x_j^k - \tau(p_j^{k+1} - p_{j+1}^{k+1}))||_2^2\}$
  end for
  Set $x_N^{k+1} = x$
  for j = 0 to N **do**
    $z_j^{k+1} = x_j^{k+1} + \kappa(x_j^{k+1} - x_j^k)$
  end for
  $change = \max\{||x^{k+1} - x^k||_2^2,\ ||p^{k+1} - p^k||_2^2\}$
  if $change < TOL$ **then**
    break
  end if
end for
$u = g(x_0) + \sum_{j=1}^N \langle p, x_j - x_{j-1}\rangle - \delta H(x_j, p_j)$
return u; the value of the solution at the point (x,t)

Next, to solve the multi-car problem, we implement the Greedy Optimal Path Finding Algorithm, summarized as Algorithm 2. For this, we implement the splitting method to solve each of the trajectories of the cars. While doing this, for each vehicle trajectory we treat the other vehicle's path as a moving obstacle and compute the optimal trajectory under this assumption. We keep iterating this until the trajectories converge within some threshold. Here, we use L^2-norm, but any other norm could be used.

Algorithm 2 Greedy Optimal Path Finding

Input: Given set of HJ equations with initial and final poses $(\boldsymbol{x}_{10}, \theta_{10})$, $(\boldsymbol{x}_{20}, \theta_{20})$, $(\boldsymbol{x}_{1f}, \theta_{1f})$, $(\boldsymbol{x}_{2f}, \theta_{2f})$

Compute optimal trajectories $x^{1,0}$ $x^{2,0}$ of C_1 and C_2 respectively in the absence of either using time-independent HJ equation via splitting method

for i = 1,2,... **do**

Compute optimal trajectory $x^{1,i}$ of C_1 while under assumption C_2 travels on $x^{2,i-1}$ with splitting method

Compute optimal trajectory $x^{2,i}$ of C_2 while under assumption C_1 travels on $x^{1,i-1}$ with splitting method

if $||x^{1,i} - x^{1,i-1}|| < \varepsilon$ and $||x^{2,i} - x^{2,i-1}|| < \varepsilon$ **then**

Break loop

end if

end for

Output: Optimal trajectories $(\boldsymbol{x}_1(t), \theta_1(t))$ and $(\boldsymbol{x}_2(t), \theta_1(t))$

4 Results

We simulate different scenarios for a car in the presence of (a) no obstacles, (b) a single stationary obstacle, (c) a single moving obstacle, (d) multiple stationary obstacles, and (e) a two car scenario. The simulations were performed in MATLAB on a Windows Surface Pro 7 laptop with Intel® Core™ i5-1035G4 CPU, and 8.00 GB total RAM.

We begin by solving the problem with no obstacles. We initialize a car at $(-1, -1)$ with orientation $\theta_0 = 0$ and set the terminal point $(1, 1)$ with orientation $\theta_f = 0$. We run one trial and find that the algorithm finishes in less than 1 second seconds The trajectory is given in Figure 3.

Next, we simulate a circular obstacle by considering a circle of radius 3/4 with center $(0, 0)$. We have the same initial and terminal destination of the vehicle with the same corresponding orientations. The algorithm finishes in less than 10 seconds. The trajectory is given in Figure 4.

We then simulate a case with a single moving obstacle, which is a circle of radius 3/4 with dynamics $\dot{x}_c = 0$ and $\dot{y}_c = 0.5\sin(t)$. The algorithm finishes in about 15 seconds. The trajectory is given in Figure 5.

We also solve the problem with multiple stationary obstacles. In our simulation, we had three circular obstacles with centers $\left(\frac{1}{2}, \frac{1}{2}\right)$, $\left(-\frac{1}{4}, -\frac{1}{4}\right)$, and $\left(\frac{3}{5}, -\frac{1}{2}\right)$ with radii $r_1 = \frac{1}{2}$, $r_2 = \frac{1}{2}$, and $r_3 = 0.35$. The algorithm takes about 1 minute and 30 seconds to finish. Notably, the max number of iterations was reached. The trajectory is shown in the Figure 6.

Finally, we implement the 2 vehicle case with initial positions $(x_0^1, y_0^1) = (-1, -1)$ and $(x_0^2, y_0^2) = (1, 1)$ with each of their terminal positions given by $(x_f^1, y_f^2) = (1, 1)$ and $(x_f^2, y_f^2) = (-1, -1)$. In other words, We set each of their terminal positions to be exactly where the other car is initialized and starting and ending up in the same orientations (facing each other). We also set the max number of between-car-iterations to be 200. We find that the algorithm doesn't converge and required all the iterations between the cars. The trajectories are shown in Figure 7.

5 Conclusion and Future Work

We considered the problem of navigating a car from an initial point to its final destination, constrained to both dynamics and physical constraints. We formalized this problem in the framework of optimal control and subsequently derived the HJB equation. We then further generalized this problem by considering both stationary and moving obstacles. We then made a final generalization to include multiple cars navigating to their destination. In doing so, we introduced a collision constraint to ensure the cars do not collide. To generate the optimal trajectories in each of these cases, we applied a novel splitting method. Our results for everything but the multi-agent case suggested that computation of the optimal trajectories take on the order of seconds, which showcases the advantage of this method versus traditional ones.

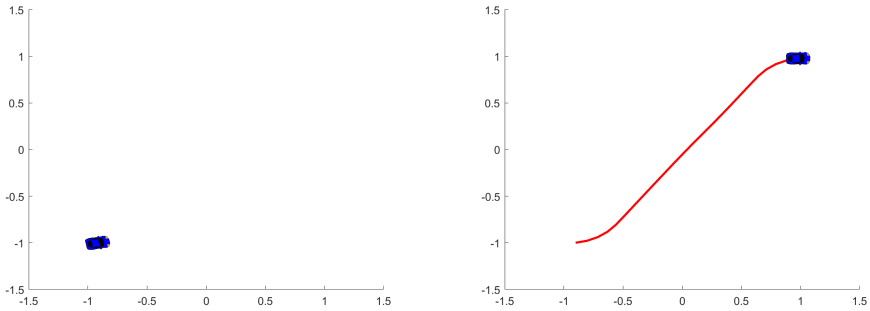
We note an interesting phenomenon in the solution of the two car case. The obtained paths share a certain symmetry; in fact, another solution to the problem may be generated by reflection. Moreover, given the trajectories of the two vehicles, we can see that by fixing the trajectory of one car, we expect that there should not be a more optimal path. In that sense, we strongly believe that the solution obtained is a Nash equilibrium, although it remains to be proven that this is the case.

One drawback of the method comes from the modeling of the problem. Since it is assumed that the minimum time T for a car to reach its destination is known *a priori*, we are able to generate the optimal trajectories of the vehicle in each case. However, if T is set to be larger than this minimum time, one may obtain non-optimal and non-realistic trajectories. Thus, determining the time horizon T must be done ad-hoc. One way to extend this work would be to modify the cost function in the alternative formulation or suggest a new one that yields an equivalent result to the time-optimal case. Secondly, another obvious pathway to extending this work is to improve on the algorithm in the multi-agent case to obtain convergence in a shorter time. We may also anticipate that uncertainty may appear in the system, e.g. in the environment or in the Lagrangian, so such a problem would be posed as a stochastic optimal control problem. Data driven approaches, such as one offered by reinforcement learning, have arisen as promising methods to solve these problems. It would be interesting to see how to adapt this approach together with the one found in this manuscript. Thirdly, it will be important to determine the efficacy of our multi-agent model and method of solution by considering more than 2 cars (say 3, 4, or 5). Finally, it would be interesting to explore how different solutions to the multi-agent problem can be obtained, either through initialization of the parameters in the algorithm, or through considering different regimes of the initial and terminal points of each car.

References

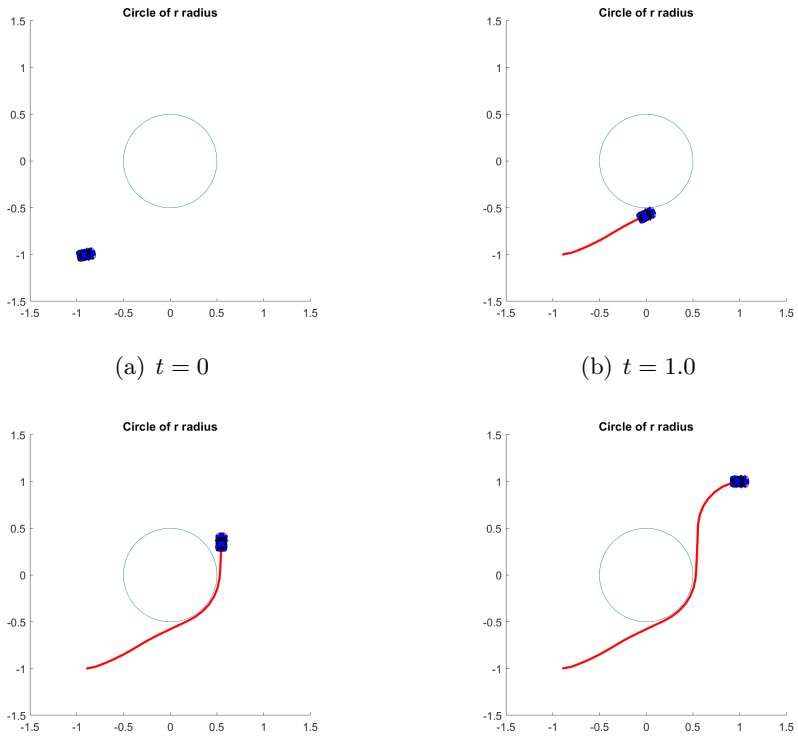
[1] Alex Tong Lin, Yat Tin Chow, and Stanley Osher. A splitting method for overcoming the curse of dimensionality in Hamilton-Jacobi equations arising from nonlinear optimal control and differential games with applications to trajectory generation. *arXiv preprint arXiv:1803.01215*, 2018.

[2] Thomas M. Howard and Alonzo Kelly. Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots. *The International Journal of Robotics Research*, 26(2):141–166, 2007.



(a) t=0 (b) t = 2.9499

Figure 3: This showcases the optimal path found no obstacles from $(x_0, y_0, \theta_0) = (-1, -1, 0)$ to $(x_f, y_f, \theta_f) = (1, 1, 0)$.



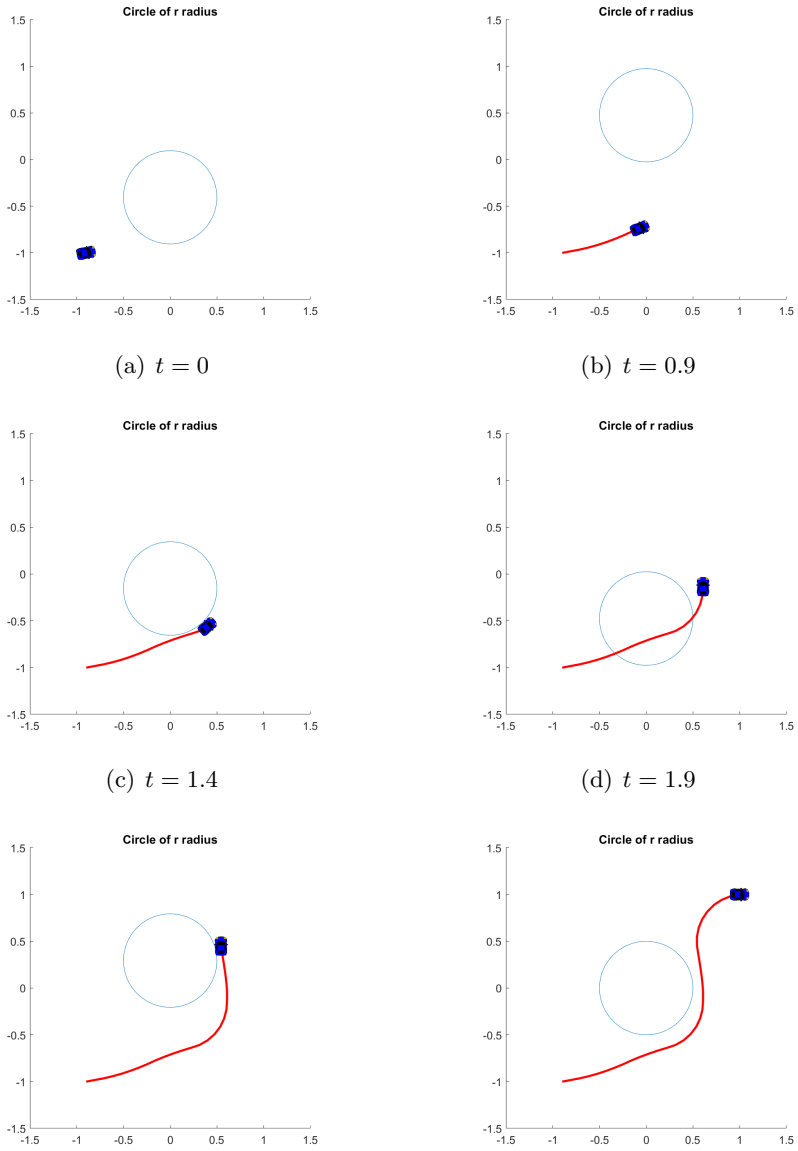
(a) $t = 0$ (b) $t = 1.0$

(c) $t = 2.2$ (d) $t = 3.2$

Figure 4: This is the optimal path for one stationary obstacle

[3] Edward Fiorelli, Naomi Ehrich Leonard, Pradeep Bhatta, Derek A. Paley, Ralf Bachmayer, and David M. Fratantoni. Multi-AUV Control and Adaptive Sampling in Monterey Bay. *IEEE Journal of Oceanic Engineering*, 31(4):935–948, 2006.

[4] Isaac Kaminer, Oleg Yakimenko, Vladimir Dobrokhodov, Antonio Pascoal, Naira Hov-



(a) $t = 0$ (b) $t = 0.9$

(c) $t = 1.4$ (d) $t = 1.9$

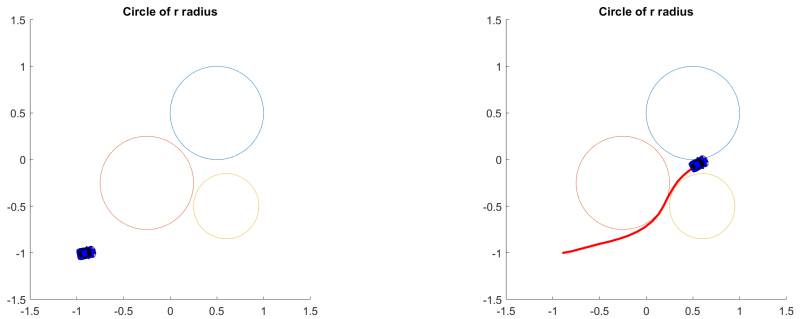
(e) $t = 2.5$ (f) $t = 3.4$

Figure 5: This showcases the optimal path found with one moving obstacle.

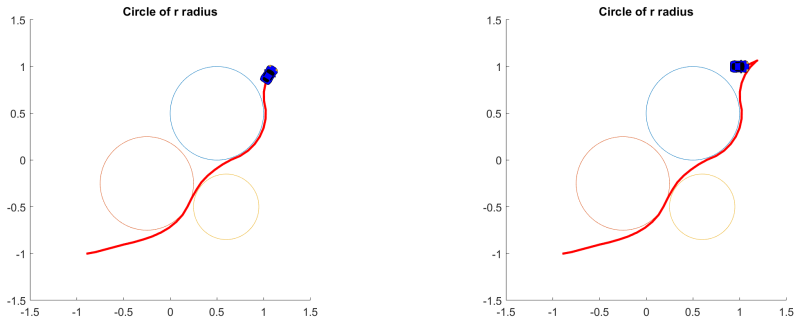
akimyan, Vijay Patel, Chengyu Cao, and Amanda Young. *Coordinated Path Following for Time-Critical Missions of Multiple UAVs via L1 Adaptive Output Feedback Controllers*.

[5] I. Garcia and J.P. How. Trajectory optimization for satellite reconfiguration maneuvers with position and attitude constraints. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 889–894 vol. 2, 2005.

[6] Andreas J. Häusler, Alessandro Saccon, António Pedro Aguiar, John Hauser, and António M. Pascoal. Energy-Optimal Motion Planning for Multiple Robotic Vehi-



(a) $t = 0$ (b) $t = 1.1$



(c) $t = 2.4$ (d) $t = 3.65$

Figure 6: This showcases the optimal path found with multiple obstacles.

cles With Collision Avoidance. *IEEE Transactions on Control Systems Technology*, 24(3):867–883, 2016.

[7] 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*, 79(3):497–516, 1957.

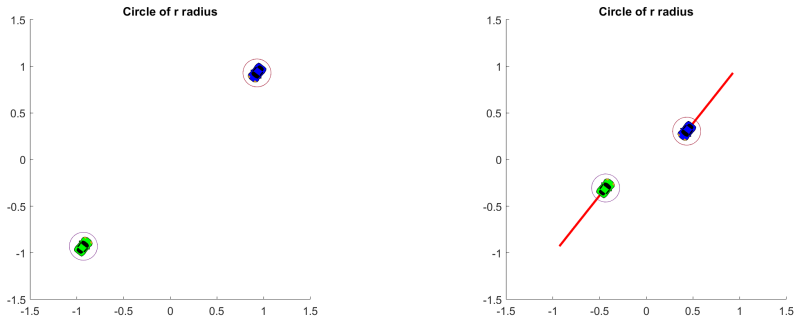
[8] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367 – 393, 1990.

[9] Ryo Takei, Richard Tsai, Haochong Shen, and Yanina Landa. A practical path-planning algorithm for a simple car: a Hamilton-Jacobi approach. In *Proceedings of the 2010 American Control Conference*, pages 6175–6180, 2010.

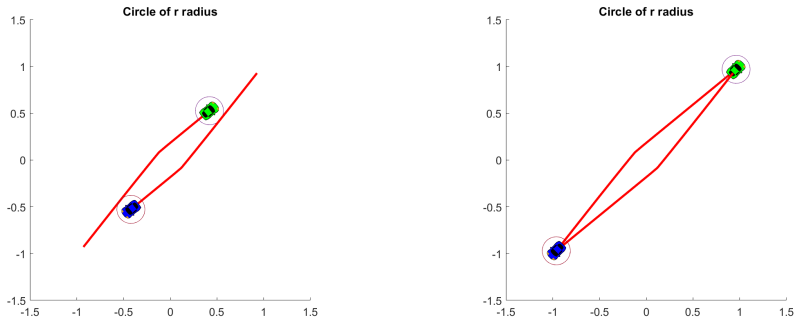
[10] Ryo Takei and Richard Tsai. Optimal Trajectories of Curvature Constrained Motion in the Hamilton–Jacobi Formulation. *Journal of Scientific Computing*, 54:622–644, 2013.

[11] Han-Jung Chou, Jing-Sin Liu, and Wen-Chieh Tung. Numerical simulation of collision-free near-shortest path generation for Dubins vehicle via Hamilton–Jacobi–Bellman equation: A case study. *Cogent Engineering*, 7(1):1782710, 2020.

[12] Christian Parkinson and Madeline Ceccia. Time-Optimal Paths for Simple Cars with Moving Obstacles in the Hamilton-Jacobi Formulation, 2021.



(a) $t = 0$ (b) $t = 0.8$



(c) $t = 2.2$ (d) $t = 2.75$

Figure 7: This showcases the (near) optimal path found with multiple cars.

[13] Christian Parkinson and Isabelle Boyle. Efficient and Scalable Path-Planning Algorithms for Curvature Constrained Motion in the Hamilton-Jacobi Formulation, 2023.

[14] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[15] Chiu-Yen Kao, Stanley Osher, and Yen-Hsi Tsai. Fast Sweeping Methods for Static Hamilton-Jacobi Equations. *SIAM Journal on Numerical Analysis*, 42(6):2612–2632, 2005.

[16] J.N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[17] J. A. Sethian and A. Vladimirsky. Fast methods for the Eikonal and related Hamilton–Jacobi equations on unstructured meshes. *Proceedings of the National Academy of Sciences*, 97(11):5699–5703, 2000.

[18] Yen-Hsi Richard Tsai, Li-Tien Cheng, Stanley Osher, and Hong-Kai Zhao. Fast sweeping algorithms for a class of hamilton-jacobi equations. *SIAM Journal on Numerical Analysis*, 41(2):673–694, 2004.

[19] Jérôme Darbon and Stanley Osher. Algorithms for Overcoming the Curse of Dimensionality for Certain Hamilton-Jacobi Equations Arising in Control Theory and Elsewhere, 2016.

[20] Yat Tin Chow, Jérôme Darbon, Stanley Osher, and Wotao Yin. Algorithm for overcoming the curse of dimensionality for state-dependent Hamilton-Jacobi equations. *Journal of Computational Physics*, 387:376–409, 2019.

[21] A. Chambolle and Thomas Pock. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, 40:120–145, 2011.

[22] Richard Bellman. On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952.

[23] Lawrence Evans. An Introduction to Mathematical Optimal Control Theory Version 0.2. 02 2013.