Improving Path Accuracy for Autonomous Parking Systems: An Optimal Control Approach

Hansen, Emma; Wang, Yebin

TR2020-099 July 03, 2020

Abstract

Kinodynamic planning explores the collision-free configuration space by constructing a tree on-the-fly. The process terminates when the tree expands into a specified neighborhood of the goal configuration. Often, the resultant path does not reach the goal accurately enough, which raises the question: how does one make an accurate, kinematically feasible connection between the tree and the goal. This is the non-trivial steering problem. Aiming to balance computational efficiency and position accuracy, this work solves an approximate steering problem through applying Pontryagin's Maximum Principle (PMP). The main contributions of this work are: establishment of an exhaustive set of possible structures of optimal control solutions; and development of a custom solver based on the these structures. Simulations demonstrate the PMP-based custom solver achieves better accuracy than a PID feedback controlbased approach, and is more computationally efficient than a gradient descent-based numerical optimization approach.

American Control Conference (ACC) 2020

Improving Path Accuracy for Autonomous Parking Systems: An Optimal Control Approach

Emma Hansen[1] and Yebin Wang[2]

Abstract— **Kinodynamic planning explores the collision-free configuration space by constructing a tree on-the-fly. The process terminates when the tree expands into a specified neighborhood of the goal configuration. Often, the resultant path does not reach the goal accurately enough, which raises the question: how does one make an accurate, kinematically feasible connection between the tree and the goal. This is the non-trivial steering problem. Aiming to balance computational efficiency and position accuracy, this work solves an approximate steering problem through applying Pontryagin's Maximum Principle (PMP). The main contributions of this work are: establishment of an exhaustive set of possible structures of optimal control solutions; and development of a custom solver based on the these structures. Simulations demonstrate the PMP-based custom solver achieves better accuracy than a PID feedback control-based approach, and is more computationally efficient than a gradient descent-based numerical optimization approach.**

I. Introduction

Path planning arises in numerous applications, such as autonomous vehicles [1] and robotics [2]. Established results include graph-based A* [3]–[6] and D* [2], [7]; navigation functions and potential fields [8]; sampling-based algorithms such as probabilistic roadmaps (PRM) [9], expansive-space trees [10], and rapidly-exploring random trees (RRT) [11] and their variants RRT* and PRM* [12], particle RRT [13], and anytime RRT [14], [15].

Sampling-based approaches are attractive to high-dimensional applications, since they relieve the curse of dimensionality. A sampling-based algorithm typically involves sampling either a configuration space [11] or control input space [10]. By sampling the configuration space, the planning algorithm achieves probabilistic completeness [9], whereas making a connection between a tree and the new sample requires solving a steering problem [12], [16]. The steering problem is hard and computationally expensive to solve for most of robotics, where dynamics are nonlinear, nonholonomic, or under-actuated. By sampling the control input space, the planning algorithm circumvents the steering problem, and is at most resolution-complete [17], [18], which means the resultant path is ϵ-distance away from the goal. Another key limitation of the sampling-based approach is its random nature, i.e., every run of the algorithm likely takes a different time period and gives a different path.

Hybrid A [19], Anytime D [20], [21], and A-search guided tree [6] address the randomness of sampling-based approaches by exploiting the idea of A-search algorithm [3], where the node selection is guided according to deterministic criteria. Given a selected node, its expansion for child nodes typically follows pre-defined rules. These algorithms possess deterministic characteristics as well as resolution-completeness, i.e., a resultant path is ϵ-distance away from the goal. Ideally, one would like to have ϵ as small as possible for accuracy purposes,but a smaller ϵ dramatically increases the scale of the tree and thus the computational burden. How to choose the ϵ which strikes a balance between computational efficiency and position accuracy is tricky. Other ideas such as using motion primitives with adaptive lengths have been proposed to balance efficiency and accuracy [21], which increases the tree complexity in the proximity of the goal.

The contributions of this paper address the balance of computational efficiency and position accuracy for autonomous parking systems by solving an approximate steering problem. Our work differentiates from the state-of-the-art in the following aspects. By avoiding solving the exact steering problem, the computational burden is greatly reduced, at the expense of a slight loss in accuracy. By solving approximate steering problem in the optimal control framework, we can improve positioning accuracy and land a smoother path over various feedback control-based path generation approaches.

In terms of technical contributions, we first formulate the approximate steering problem as a constrained optimal control problem, and employ Pontryagin's Maximum Principle (PMP) to obtain necessary optimality conditions. Rigorous analysis of these conditions is carried out to establish an exhaustive set of possible structures for optimal control solutions of the approximate steering problem. Based on the complete set of possible structures, we further develop a custom solver to save computation time for solving the approximate steering problem. Finally, we compare the results of this custom optimal control-based solver with results from a gradient descent (GD) solver and a PID feedback control-based solver. It is shown in terms of accuracy of the final solution, the custom solver is more accurate than the PID solver, but the GD solver is more accurate than the custom solver. In terms of computational efficiency, the custom solver is faster than the GD solver as the number of time steps increases above 50.

II. Problem Formulation

Consider a robot with the following dynamics

$$\dot{X} = f(X) + g(X)u, \tag{1}$$

where $X \in \mathcal{X} \subset \mathbb{R}^{n_x}$ is state, $u \in \mathcal{U} \subset \mathbb{R}^m$ the control, f a smooth vector field (the drift), and $g = [g_1^\top, \cdots, g_m^\top]^\top$ with g_i a smooth vector field. A *configuration* of (1) is a complete specification of the position of every point in that

[1]Emma Hansen is a Master's student at the University of Washington Dept. of Aeronautics and Astronautics, Seattle, WA 98195. This work was done while she was an intern with Mitsubishi Electric Research Laboratories. `email: emmah0@uw.edu`

[2]Yebin Wang is with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139. `email: yebinwang@ieee.org`

system. The *configuration space* $\mathcal{C} \subset \mathbb{R}^{n_c}$ is a compact set representing all possible configurations of the system. A collision-free configuration space $\mathcal{C}_{free}$ is the set of configurations at which the robot has no intersection with obstacles in the environment, the complement of this is the collision configuration space, $\mathcal{C}_{obs} = \mathcal{C} \backslash \mathcal{C}_{free}$. An *admissible trajectory* $\mathcal{X}_t$ is a solution of (1) with given initial and final conditions and $u \in \mathcal{U}$. An *admissible path* $\mathcal{P}_t$ is the image of an admissible trajectory on $\mathcal{C}$. For brevity, an admissible path, if additionally collision-free, is termed a *feasible path*. Whenever (1) represents its kinematics, $n_x = n_c$ and $\mathcal{C} = \mathcal{X}$, which is assumed in this work.

Now, consider a front wheel drive vehicle. Its kinematics are modeled as follows [22]:

$$\dot{x} = v\cos(\theta),\ \dot{y} = v\sin(\theta),\ \dot{\theta} = vs/R, \tag{2}$$

where (x, y) are the coordinates of the midpoint of the rear wheels, θ is the vehicle orientation, v is the longitudinal velocity, s is a normalized steering control, and R is the minimum turning radius. The system state space $X = (x, y, \theta)^\top$ coincides with the configuration space, i.e., $\mathcal{C} = \mathcal{X} \subset \mathbb{R}^3$.

A. Path Planning Problem

A typical path planning problem is given as follows.

Problem 2.1: Given an initial configuration $X_0 \in \mathcal{C}_{free}$, a goal configuration $X_f \in \mathcal{C}_{free}$, and system (2), find a feasible path $\mathcal{P}_t : [0, 1] \to \mathcal{C}_{free}$ which:

(I) starts at X_0 and ends at X_f, while satisfying (2); and
(II) lies in the collision-free configuration space $\mathcal{C}_{free}$.

Let $J(\cdot)$ be a cost function assigning to each non-trivial path a non-negative cost. Optimal path planning is to find a feasible path $\mathcal{P}_t^* : [0, 1] \to \mathcal{C}_{free}$ that minimizes $J(\cdot)$.

Most path planning algorithms do not solve Problem 2.1 exactly: a path ends at a configuration $\tilde{X}_f \neq X_f$, and thus violates condition (I). However, path planning algorithms ensure $\tilde{X}_f$ lies in an ϵ-neighborhood of X_f:

$$\mathcal{B}_\epsilon(X_f) \triangleq \{X | d(X, X_f) \leq \epsilon, \forall X \in \mathcal{X}\},$$

where $d(\cdot, \cdot)$ is a distance metric. Designing smooth motion of the vehicle from $\tilde{X}_f$ to X_f is the focus of this paper.

B. Exact and Approximate Steering Problems

Denote $p(t) = [x(t)\ \ y(t)\ \ \theta(t)]$, and let $u(t) = [s(t)\ \ v(t)]$ be the control input. The exact steering problem can be formulated as the following optimal control problem.

Problem 2.2: Given p_0 the initial configuration, p_f the goal configuration, t_f the fixed final time, and a cost function $L(p, u)$, find control $s(t) \in [-b_s, b_s], \forall t \in [0, t_f]$, and $v(t) \in [b_{v,l}, b_{v,u}], \forall t$, satisfying:

$$\begin{aligned} \min_{s(t),v(t)} \quad & \int_0^{t_f} L(p, u)\mathrm{d}t \\ \text{subject to} \quad & \dot{p} = \begin{bmatrix} v(t)\cos(\theta(t)) \\ v(t)\sin(\theta(t)) \\ v(t)s(t)/R \end{bmatrix}, \\ & p(0) = p_0,\ p(t_f) = p_f, \\ & -b_s \leq s(t) \leq b_s,\ b_{v,l} \leq v(t) \leq b_{v,u}, \end{aligned}$$

where and $b_s,\ b_{v,u}$ and $-b_s,\ b_{v,l}$ are the upper and lower bounds of the steering and speed, respectively.

Remark 2.3: If $L(p, u) = 1$, Problem 2.2 is reduced to the Reeds-Shepp problem, and its analytical solution can be obtained as a Reeds-Shepp path [22]. A drawback of Reeds-Shepp path is the discontinuity of steering control. Aiming to produce a smooth movement of the vehicle, we take $L(p, u) = s(t)^2$ to ensure that the resultant path is continuous. A similar problem has been studied in [23], where the optimal solution could be challenging to find.

In practice, solving the exact steering problem with cost function $L(p, u) = s(t)^2$ is unnecessary as it is impossible for a vehicle to reach the goal. Considering the path generated by a path planner is ϵ-distance away from the goal, what we need is how to move the vehicle toward the goal with a simple manoeuvre. It is realistic to consider an approximate steering problem, where the feasible control set is restricted. Specifically, to enforce the constraint $v > 0$ or $v < 0$ to reflect the desire that the movement from $\tilde{X}_f$ to p_f does not involve the change of gear, because $\tilde{X}_f \in \mathcal{B}_\epsilon(X_f)$. Enforcing this directional constraint on longitudinal velocity reduces (2) to a Dubins car [24].

With the additional constraint on the direction of longitudinal velocity v, one can reparameterize the kinematic model (2) in terms of path length, l. Without loss of generality, assume v is fixed and always positive. The speed along the path is exactly the longitudinal velocity v, i.e., $\mathrm{d}l(t)/\mathrm{d}t = v$. As a result, (2) can be rewritten:

$$\frac{\mathrm{d}p}{\mathrm{d}l} = f(p, u, t) \triangleq \begin{bmatrix} \cos\theta(l) \\ \sin\theta(l) \\ s(l)/R \end{bmatrix}. \tag{3}$$

The approximate steering problem is given as follows.

Problem 2.4: Given the initial configuration p_0, the goal configuration p_f, and a cost function $L(p, u) = s(l)^2$, find a finite path length l_f and continuous control $s(l) \in [-b_s, b_s],\ l \in [0, l_f]$ satisfying:

$$\begin{aligned} \min_{s(l),l_f} \quad & \int_0^{l_f} s(l)^2\mathrm{d}l \\ \text{subject to} \quad & (3), \\ & p(0) = p_0,\ p(l_f) = p_f, \\ & -b_s \leq s(l) \leq b_s. \end{aligned} \tag{4}$$

Remark 2.5: Problem 2.4 is a free final "time" (path length) problem. From [24], it admits at least one feasible solution: Dubins path. On the other hand, similar to the Reeds-Shepp path, a Dubins path between two arbitrary configurations comprises of circular and straight arcs, and thus suffers from discontinuous steering control.

Instead of imposing the boundary condition $p(l_f) = p_f$, one can augment the cost function $\int_0^{l_f} s^2\mathrm{d}l$ with a terminal cost $|p(l_f) - p_f|_Q^2$ with Q being a weighted matrix. It is apparent that the optimal solution to an approximate steering problem with the new cost function always exists. Playing with boundary conditions or the cost function does not affect main results established in this work. We focus on Problem 2.4 for illustrative purpose.

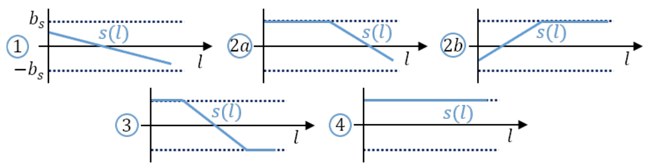

Fig. 1. Solution forms from simulation results. Only upper bound solutions for steering control are shown, lower bound solutions are also viable.

C. *Numerical Optimization Approach*

Problems 2.2 - 2.4 are continuous-time constrained optimal control problems. A prevailing approach to solve these problems is numerical optimization, where the control is parameterized over time or functional space, and state, cost function, and system dynamics are discretized over time or arc length to cast the continuous-time problem into a numerical optimization problem. Interested readers are referred to [25]–[28] and references therein for details.

Following a numerical optimization approach, Problem 2.2 was solved directly using Matlab's `fmincon`, which proved to be computationally expensive as the number of time steps were increased. To reduce the computation time to less than 1s, the maximum number of iterations was restricted to 50; this resulted in a significant drop in accuracy. To reduce computation time while maintaining accuracy, a GD numerical optimization approach was developed. In this solver, the problem was re-written as an unconstrained optimization problem using penalty terms with cost function:

$$\kappa_1||p(l_f) - p_f||_Q^2 + \int_0^{l_f} s(l)^2 + \kappa_2 \left(\max(s(l) - b_s, 0)^2 + \max(-b_s - s(l), 0)^2\right) \mathrm{d}l,$$

where κ_1, κ_2 are weighting terms.

Simulation of the GD solver revealed an optimal steering control trajectory admits 10 possible structures. They can be characterized by the piecewise equation:

$$s(l) = \begin{cases} \pm b_s & l \in [0, l_1) \\ \rho(l, \theta(l)) & l \in [l_1, l_2) \\ \mp b_s & l \in [l_2, l_f] \end{cases}, \tag{5}$$

where $\rho(l)$ is a function defining the motion of $\theta(l)$, and l_1, l_2 are where $s(l)$ switches from constrained to unconstrained, or vice versa. As an example, Figure 1 summarizes five cases where the steering control hits the upper bound. The goal is to show rigorously the steering control can only take on forms given by (5), under certain assumptions on p_f. This can be done by showing the steering control can only pass through zero ($s(l) = 0$) once, and cannot consecutively hit and leave the same boundary.

D. *Assumptions and Notation*

A path planner [6] is cascaded with our approximate steering solver. With goal X_f, the path planner will stop at $\tilde{X}_f$ if it satisfies the following stop criteria:

$$|\tilde{X}_f - X_f|_Q \leq 1, \; |\mathcal{P}_{rs}(\tilde{X}_f, X_f)| \leq 1,$$

where $Q = \mathrm{diag}(1, 1, R^2)$ and $|\mathcal{P}_{rs}(\tilde{X}_f, X_f)|$ represents the length of the Reeds-Shepp path between $\tilde{X}_f$ and X_f. Solving the approximate steering problem with initial $p_0 = \tilde{X}_f$ and goal $p_f = X_f$ is equivalent to solving the same problem with $p_0 = 0$ and $p_f = X_f - \tilde{X}_f$. The first stop criteria avoids the presence of a large orientation error, represented by θ_f in p_f. Particularly, $-1/R < \theta_f < 1/R$. For parking tasks, the orientation error is necessarily small due to the tightness of the parking space and the fact that a large orientation error likely leads to a collision. Additionally, the second stop criteria keeps the lateral error, y_f, small. This reflects the fact that a small lateral error requires a long Reeds-Shepp path to connect. Without loss of generality, we assume p_f is located in the first quadrant ($x_f > 0, y_f > 0$) and $0 \leq \theta_f < 1/R$. Additionally, we assume $x_f < R$ for a small adjustment.

A dot over a symbol, $\dot{z}$, indicates a derivative with respect to t or l, whichever is consistent with the expression.

III. Main Results

In this section we apply PMP to Problem 2.4 and analyze necessary optimality conditions to establish that the steering control will only ever cross zero ($s(l) = 0$) once. This guarantees the solutions will be of the form in (5).

A. *Pontryagin's Maximum Principle Set-up*

It can be verified that Problem 2.4 satisfies Assumptions 3.1-3.3 of Chapter V-3 in [29], and it satisfies the set of feasible controls is fixed. Thus, we can use the point-wise maximum principle in [29]. Hamiltonian for Problem 2.4 is:

$$H(l, p, s, \lambda_0, \lambda) = \lambda_0 s(l)^2 + \lambda_1(l)\cos\theta(l) + \lambda_2(l)\sin\theta(l) + \lambda_3(l)\frac{s(l)}{R}, \tag{6}$$

where $\lambda_0 = \{0, -1\}$. Necessary optimality conditions are:

1) Vector $(\lambda_0, \lambda(l))$ satisfies $(\lambda_0, \lambda(l)) \neq 0$ for $l \in [0, l_f]$.
2) For almost every $l \in [0, l_f]$, the costate dynamics are:

$$\frac{\mathrm{d}\lambda}{\mathrm{d}l} = -\left(\frac{\partial H}{\partial p}\right)^\top = \begin{bmatrix} 0 \\ 0 \\ \lambda_1(l)\sin\theta(l) - \lambda_2(l)\cos\theta(l) \end{bmatrix},$$

which admits the following analytical solution:

$$\lambda_1 = c_1, \quad \lambda_2 = c_2, \quad \dot{\lambda}_3 = c_1 \sin\theta(l) - c_2\cos\theta(l).$$

3) For any feasible control $\bar{s}(l)$,

$$H(l, p, s, \lambda_0, \lambda) \leq H(l, p, \bar{s}, \lambda_0, \lambda),$$

where s is the optimal solution.
4) If the cost function and state dynamics are continuous at $l = 0$ and $l = l_f$, the transversality condition says the vector $(H(0), -\lambda(0), H(l_f), \lambda(l_f))$ is orthogonal to the set of boundary conditions given by $\mathcal{B} = \{(l_0, p_0, l_f, p_f)\}$. Since l_f is the only free variable in $\mathcal{B}$ then it must be that $H(l_f) = 0$, which means $H(l) = 0$ for all l, because the Hamiltonian is not directly dependent on l [29].

For notational simplicity we drop the arguments p, s, λ_0, λ and write the Hamiltonian as $H(l)$.

B. Analysis of Necessary Optimality Conditions

To show the steering control can only take the forms in (5), we prove $s(l)$ can only ever cross 0 once.

Let $c_1 = k\cos(\alpha)$ and $c_2 = k\sin(\alpha)$, then $k = \sqrt{c_1^2 + c_2^2} \geq 0$ and $\alpha = \arctan(\frac{c_2}{c_1}) + m\pi \in (-\frac{\pi}{2} + m\pi, \frac{\pi}{2} + m\pi)$, $m \in \mathbb{Z}$. Write the Hamiltonian and the costate dynamics as:

$$H(l) = \lambda_0 s(l)^2 + k\cos(\theta(l) - \alpha) + \lambda_3(l)\frac{s(l)}{R},$$
$$\dot{\lambda}_3(l) = k\sin(\theta(l) - \alpha).$$

Proposition 1: The costate $\lambda_3(l)$ from Problem 2.4 can only cross through $\lambda_3(l) = 0$ once.

Proof: Begin by examining the unconstrained steering control solution. When the control $s(l)$ is unconstrained, it should solve the equation $\frac{\partial H}{\partial s} = 0$,

$$\frac{\partial H}{\partial s} = 0 = 2\lambda_0 s(l) + \frac{1}{R}\lambda_3(l). \tag{7}$$

If we were to have $\lambda_0 = 0$, then (7) would give $\lambda_3(l) = 0$. If $\lambda_3(l) = 0$ for any interval in l then it must be $k = 0 \Rightarrow c_1 = c_2 = 0$, which results in $\lambda(l) = 0$, but we know $(\lambda_0, \lambda(l)) \neq 0$, thus if there exists an unconstrained portion of the control, we must have $\lambda_0 = -1$. With this, we can now rearrange (7) for $s(l)$:

$$s(l) = \frac{1}{2R}\lambda_3(l). \tag{8}$$

Substituting this into the Hamiltonian gives:

$$H(l) = -\left(\frac{1}{2R}\lambda_3(l)\right)^2 + k\cos(\theta(l) - \alpha) + \lambda_3(l)\left(\frac{1}{2R^2}\lambda_3(l)\right) = 0,$$

which can be solved for $\lambda_3(l)$ as:

$$\lambda_3(l) = \pm\sqrt{-4kR^2\cos(\theta(l) - \alpha)}. \tag{9}$$

To check this equation for $\lambda_3(l)$ satisfies the costate dynamics, take the derivative with respect to l:

$$\begin{aligned}\frac{d\lambda_3}{dl} &= \pm\frac{d}{dl}\sqrt{-4kR^2\cos(\theta(l) - \alpha)} \\ &= \frac{\pm 1}{2}\frac{-4kR^2(-\sin(\theta(l) - \alpha))\frac{d\theta}{dl}}{\sqrt{-4kR^2\cos(\theta(l) - \alpha)}} \\ &= \frac{\pm 2kR\ s(l)\sin(\theta(l) - \alpha)}{\sqrt{-4kR^2\cos(\theta(l) - \alpha)}}.\end{aligned}$$

We know $\frac{d\lambda_3}{dl} = k\sin(\theta(l) - \alpha)$, substituting this into the above equation and rearranging for $s(l)$ gives:

$$s(l) = \frac{\pm 1}{2R}\sqrt{-4kR^2\cos(\theta(l) - \alpha)}. \tag{10}$$

Now, note $\lambda_3(l) = \pm\sqrt{-4kR^2\cos(\theta(l) - \alpha)}$. Thus:

$$s(l) = \frac{\lambda_3(l)}{2R},$$

which is exactly what we get from differentiating the Hamiltonian! Thus, (9) satisfies the costate dynamics.

Now, consider $\lambda_3(\theta(l)) = 0$. This requires $0 = \cos(\theta(l) - \alpha)$, which is satisfied when:

$$\theta(l) - \alpha = n\pi - \frac{\pi}{2}, n \in \mathbb{Z}.$$

Thus, $\sin(\theta(l) - \alpha)$ does not change sign when $\lambda_3(l) = 0$.

In order to have a real-valued solution we must have $\cos(\theta(l) - \alpha) < 0$. This gives $2\pi n + \frac{\pi}{2} \leq \theta(l) - \alpha \leq 2\pi n + \frac{3\pi}{2}$,

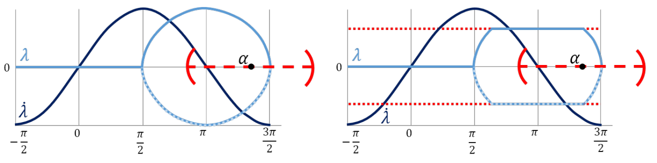

Fig. 2. Left: $\lambda_3(l)$ (light blue), and $\dot{\lambda}_3(l)$ (dark blue), as functions of $\theta(l) - \alpha$. The dashed portion of $\lambda_3(l)$ shows where $\lambda_3(l)$ is negative. The dashed red line shows the range where $\theta(l)$ can exist based on α. Right: A visualization of what $\lambda_3(l)$ might look like for a steering control solution when the steering control hits the boundary, given by the dotted red line.

which needs to hold for all $\theta(l)$. In particular, consider when $\theta(l) = 0$, then $2\pi n + \frac{\pi}{2} \leq -\alpha \leq 2\pi n + \frac{3\pi}{2}$ which, for $n = -1$, gives: $\frac{\pi}{2} \leq \alpha \leq \frac{3\pi}{2}$. This has non-zero intersection with the range of α given before when $m = 1$, therefore $\alpha \in (\frac{\pi}{2}, \frac{3\pi}{2})$.

Now, show $\lambda_3(l)$ can only cross 0 once. To do so, we analyse the behaviour of $\lambda_3(l)$ immediately before and after it touches 0. Consider Figure 2, there are two cases to examine: $\lambda_3(l) \to 0$ from the positive side, and $\lambda_3(l) \to 0$ from the negative side.

1) If $\lambda_3(l) > 0$ approaches 0, then $\theta(l)$ is increasing and we must have $\frac{d\lambda_3}{dl} < 0$. Since $\frac{d\lambda_3}{dl}$ does not change sign when $\lambda_3(l)$ does, then $\lambda_3(l)$ will continue to decrease after touching 0. Since $\lambda_3(l) < 0$ now, $\theta(l)$ will decrease, which shows $\lambda_3(l)$ will follow clockwise the right half of the loop in Figure 2: Left.
2) If $\lambda_3(l) < 0$ approaches 0, then $\theta(l)$ is decreasing and we must have $\frac{d\lambda_3}{dl} > 0$. By the same argument as above, $\lambda_3(l)$ will continue to increase after touching 0, and $\theta(l)$ will begin to increase. Thus, $\lambda_3(l)$ will follow clockwise the left half of the loop in Figure 2: Left.

This shows $\lambda_3(l)$ will follow the loop in the left plot of Figure 2 clockwise, and without getting stuck at 0.

The relationship between $\lambda_3(l)$ and $\theta(l)$ determines the only way $\lambda_3(l)$ can pass through 0 more than once is if it passes through first at $\theta(l) - \alpha = \frac{3\pi}{2}$ and then at $\theta(l) - \alpha = \frac{\pi}{2}$, or in the opposite order; $\lambda_3(l)$ cannot pass through 0 consecutively at the same value of $\theta(l) - \alpha$.

Now, knowing $\alpha \in (\frac{\pi}{2}, \frac{3\pi}{2})$, consider the range around α where $\theta(l)$ can exist - this is shown by the red dashed line in Figure 2: Left. If $\alpha > \pi$ or $\alpha < \pi$, $\theta(l)$ has no chance of passing through 0 twice. If $\alpha = \pi$, $\lambda_3(l)$ still cannot pass through 0 twice because $\theta(l) \in (-\frac{\pi}{2}, \frac{\pi}{2})$ the open interval, since it is not physically possible for $|\theta(l)| \geq \pi/2$ if $x_f < R$. Thus, we have shown $\lambda_3(l)$ cannot pass through 0 twice. ■

Using Proposition 1, we can state the following result about the steering control.

Proposition 2: The steering control solution, $s(l)$, for Problem 2.4 cannot pass through $s(l) = 0$ more than once, nor can it consecutively hit and leave the same boundary.

Proof: From Proposition 1 and the costate dynamics, we can conclude if $s(l)$ starts on a boundary, see the right plot in Figure 2 for the corresponding $\lambda_3(l)$ visualization, that $s(l)$ cannot go back to the same boundary. Consider the case of consecutively hitting the upper boundary: to do so, $\lambda_3(l)$ would need to decrease off the boundary and then increase back toward the boundary, all the while remaining positive because $\lambda_3(l)$ cannot pass through zero more than once. This is not possible given the $\lambda_3(l)$ dynamics. A similar argument can be made for the lower boundary. Thus, $s(l)$ cannot consecutively hit the same boundary.

Additionally, from (8) and Proposition 1 we have $s(l)$ can only pass through zero once. Therefore, we have shown $s(l)$ it will only take on the form claimed in (5). ∎

C. *The Custom Solver*

Now that we have the form of $\lambda_3(l)$ and how it is related to $s(l)$, via (10), the general form of $s(l)$ is:

$$s(l) = \begin{cases} \pm b_s & l \in [0, l_1) \\ \pm\sqrt{-k\cos(\theta(l)-\alpha)} & l \in [l_1, l_2) \\ \mp b_s & l \in [l_2, l_f] \end{cases} .$$

This can be substituted into the $\theta(l)$ dynamics and integrated to solve for $\theta(l)$ as:

$$\theta(l) = \begin{cases} \pm\frac{b_s}{R} l & l \in [0, l_1) \\ \alpha + \pi - 2\mathrm{am}[\frac{\sqrt{k}}{2R}(Rc_1 \mp l)|2] & l \in [l_1, l_2) \\ \mp\frac{b_s}{R} l & l \in [l_2, l_f] \end{cases} ,$$

where $\mathtt{am}$ is the Jacobi amplitude function $\phi = \mathtt{am}(z|m)$ with parameter $m = \bar{k}^2$, where $\bar{k}$ is the modulus, and c_1 is an integration constant. But, due to the possibility that the steering control may need to switch sign during the unconstrained portion, using the differential equation system with costate dynamics was chosen over using the exact solution for the unconstrained segment.

The general forms of the systems of equations the custom solver solves are:

$$\left.\begin{aligned} \theta(l) &= \pm\tfrac{b_s}{R} l \\ x(l) &= \mp\tfrac{R}{b_s}\sin(\pm\tfrac{b_s}{R} l) \\ y(l) &= \pm\tfrac{R}{b_s}\cos(\pm\tfrac{b_s}{R} l) \end{aligned}\right\} \text{ for } l \in [0, l_f] \text{ s.t } s(l) = \pm b_s,$$

and

$$\left.\begin{aligned} \dot{\lambda}_3(l) &= k\sin(\theta(l)-\alpha) \\ \dot{\theta}(l) &= \tfrac{\lambda_3(l)}{2R^2} \\ \dot{x}(l) &= \cos(\theta(l)) \\ \dot{y}(l) &= \sin(\theta(l)) \end{aligned}\right\} \text{ for } l \in [0, l_f] \text{ s.t } |s(l)| < b_s.$$

Where the order in which the systems are solved and the boundary conditions are dependent on the shape of the control solution, which can be any of the shapes found in Figure 1, or with the lower bounds.

IV. Simulation

This section conducts two comparisons. The first compares the accuracy of the final state achieved by applying three different methods to solve Problem 2.4: the custom solver using the structure of (5), a GD-based solver, and a PID feedback control-based method; the second compares computational efficiency of the custom solver and the GD method. For all simulations, the values used are: $R = 4.132$m, $b_{v,l} = 0$m/s, $b_{v,u} = 2$m/s, $b_s = 1$, and $t_f = 0.5$s. Except for the PID feedback control-based method, the number of iterations of all other solvers was capped at $N = 50$. This was to enforce a shorter computation time, with the goal of eventually having the algorithm run in real-time. Unless otherwise stated, all

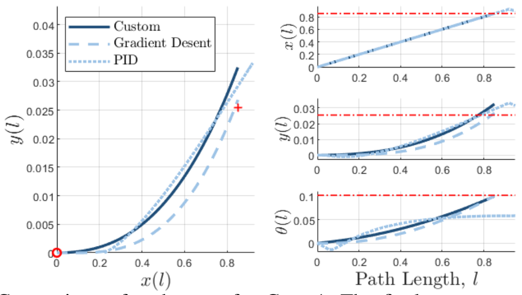

Fig. 3. Comparison of path error for Case 1. The final state error for the custom, GD, and PID methods is 0.0073, 0.0017, and 0.0593, respectively. The red dash-dot line in each plot on the left is the goal state.

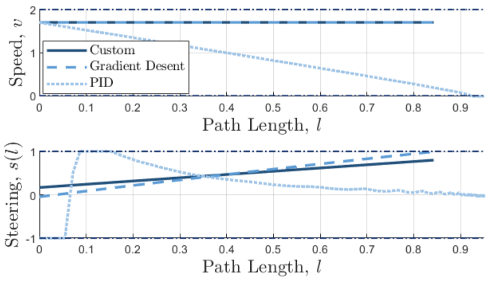

Fig. 4. Control solutions for Case 1 for each of the three methods. The dash-dot lines indicate the control bounds.

data which involves statistics, elapsed time, and final state error was gathered over 100 simulations, so that any trends would be statistically significant.

The PID feedback control-based method generates a path by solving an output regulation problem. It is illustrated by Fig. 5, where the PID block adopts two PID controllers to regulate longitudinal and lateral errors toward zero, respectively. The longitudinal and lateral errors are defined in the body frame attached to the measured path $p(t)$. We take PID control as a baseline for simplicity, which is a key requirement in practice.

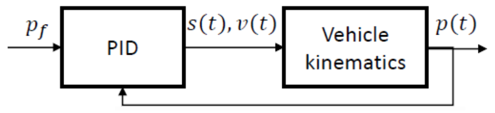

Fig. 5. Diagram for PID feedback control-based method.

In the first comparison, the final positions, p_f, for three cases are given as follows:

Case 1: $p_{f1} = (0.8503, 0.0255, 0.1012)$

Case 2: $p_{f2} = (0.7238, 0.0498, 0.1919)$

Case 3: $p_{f3} = (0.8067, 0.0561, 0.0693)$.

Case 1 was picked to demonstrate a fully unconstrained control solution; Case 2 was picked to demonstrate a fully constrained control solution; and Case 3 was picked to demonstrate an initially constrained, then unconstrained solution. The path generation results for three methods are shown in Figures 3-9. Both the custom and GD solvers were run with a final time of 0.5s. Due to the slower convergence time of the PID feedback approach, if it were run for only 0.5s the final state error was ≈ 0.27, many orders or magnitude larger than the error of the custom and GD solvers. For this reason, the PID feedback control solver was run for 5s.

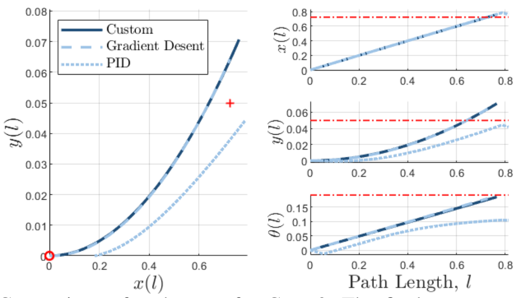

Fig. 6. Comparison of path error for Case 2. The final state error for the custom, GD, and PID methods is 0.0431, 0.0188, and 0.0960, respectively. The red dash-dot line in each plot on the left is the goal state.

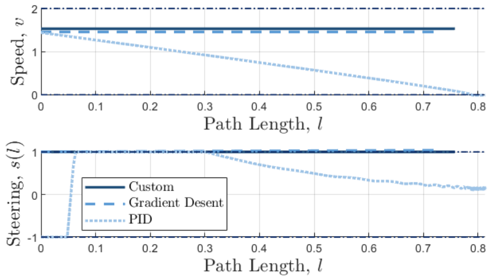

Fig. 7. Control inputs for Case 2 for each of the three methods. The dash-dot lines indicate the control bounds.

From the figures and reported errors, it is clear the custom method outperforms the PID method in terms of accuracy. It should also be noted the gradient method outperforms both the custom and PID methods in terms of accuracy in Cases 1 and 2. In Figures 4 and 7 the steering solution of the PID method starts off negative, which is an unnecessary manoeuvre (based on the GD solution), but can be explained since the PID solution does not take path quality into account. It was observed, in general, that the GD method performs better in terms of accuracy than the custom solver. This is counter-intuitive, and we leave it as future work to investigate why. But, as demonstrated below, the custom solver is much faster than the GD method.

For the comparison of computational efficiency, the goal configuration, p_f, is chosen randomly according to

$$\begin{aligned} x_f &= 0.7 + 0.2\chi, \quad \chi \sim \mathcal{U}(0,1), \\ y_f &= 0.1\gamma, \quad \gamma \sim \mathcal{U}(0,1), \\ \theta_f &= 0.2\tau, \quad \tau \sim \mathcal{U}(0,1), \end{aligned}$$

where χ, γ, τ are random variables having a uniform distribution $\mathcal{U}(0,1)$. For the results of the computational efficiency comparison between the custom and GD solvers see Table I.

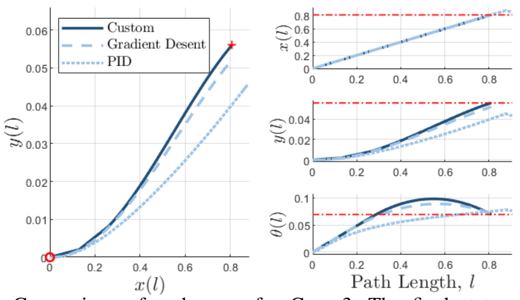

Fig. 8. Comparison of path error for Case 3. The final state error for the custom, GD, and PID methods is 5.305e-05, 0.0041, and 0.0421, respectively. The red dash-dot in each plot on the left is the goal state.

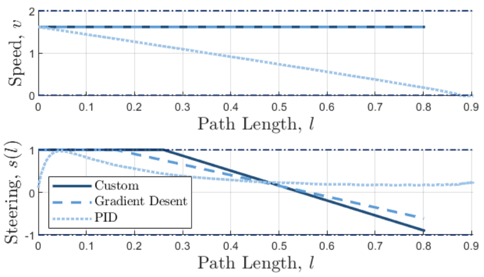

Fig. 9. Control inputs for Case 3 for each of the three methods. The dash-dot lines indicate the control bounds.

TABLE I

Comparison of average elapsed time of the custom, $T_{el,C}$, and GD, $T_{el,G}$, solvers and the percent of simulations where custom was faster than GD.

N	$T_{el,C}$ [s]	$T_{el,G}$ [s]	% Custom faster
200	0.5034	1.2355	98%
150	0.5020	1.0235	97%
125	0.4020	0.7875	93%
100	0.4363	0.6544	81%
50	0.4116	0.4045	62%
25	0.3645	0.2846	38%

These results show as N increases, the computation time of the gradient method increases significantly, whereas the computation time of the custom method changes very little. Thus, as the number of time steps increases, the custom method becomes more computationally efficient than the gradient method. A detailed complexity analysis of both solvers is left for future extension.

When a time budget of 0.5s is enforced it was found the custom solver can work with up to $N \approx 175$ time steps, whereas the GD solver can only work with up to $N \approx 75$ time steps. With the custom solver running $N = 175$ and the GD solver running $N = 75$ time steps, it was still found the GD solver was more accurate 86% of the time.

In addition, it was noticed in 19% of the simulations for the custom solver, and 14.75% of the simulations for the gradient solver, feasible solutions were produced. We characterize a feasible solution as one where the final state error was less than or equal to 0.001. Figure 10 offers a comparison of the y_f and θ_f coordinates of the end points corresponding to feasible and infeasible end points, for the custom solver (left) and the gradient solver (right). From this figure, it is seen that the end points which correspond to infeasible solutions are those related to more "extreme" parking manoeuvres, i.e., ones involving a large y_f with a small θ_f, or ones involving a small y_f with a large θ_f. It is understandable that extreme manoeuvres would be more difficult to find feasible solutions for, but the rate of finding feasible solutions is still undesirable. In future work, increasing the maximum number of iterations to examine the effect on solution feasibility should be examined along with improving computation time.

Table II contains the average elapsed run time, T_{el}, and error in the final state, $\delta p(t_f)$, of the GD solver (G) for the approximate problem and `fmincon` for the exact problem (E). The exact problem results show the average computation time does not increase significantly as the number of times steps increases, but it should be noted for $N > 25$,

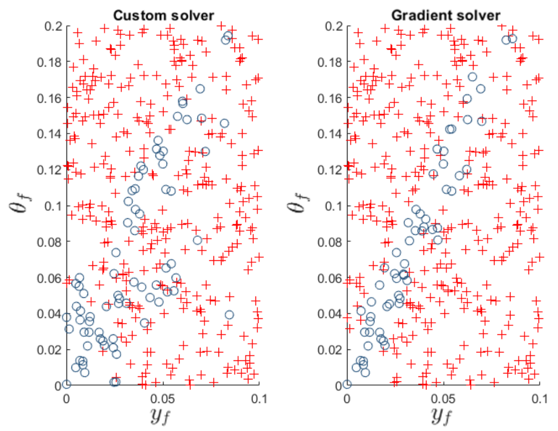

Fig. 10. Comparison of y_f and θ_f feasible end points (blue $\circ$) and the infeasible endpoints (red $+$). Data was gathered over 400 simulations.

TABLE II

Comparison of run time and accuracy of GD solver for Problem 2.4 and `fmincon` for Problem 2.2. Iterations were capped at 50 for both solvers.

N	$\delta p_G(t_f)$	$\delta p_E(t_f)$	$T_{el,G}$ [s]	$T_{el,E}$ [s]
200	0.0130	0.2834	1.3737	0.7120
150	0.0160	0.2761	1.0190	0.2729
125	0.0111	0.2803	0.8699	0.1521
100	0.0132	0.2740	0.6314	0.1142
50	0.0135	0.2805	0.3857	0.0510
25	0.0135	0.1910	0.2722	0.0481

`fmincon` was unable to converge and produced a constant output for both steering and speed. This is largely due to capping the number of iterations to 50. With such a practical restriction on iterations, the nonlinear optimization solver is unlikely to converge and produce meaningful output.

V. Conclusions and Future Work

This paper showed, using Pontryagin's maximum principle, that under realistic assumptions on the goal configuration, the only form the steering control can take for an approximate steering problem is one of 10 forms where it can only cross through zero once. This guarantee on the steering solution form was used to develop a custom solver for the problem, which was then compared against the solution from a GD method, and from a PID feedback control method. Simulations showed the custom solver was more accurate than the PID feedback method, but less accurate than the GD. In terms of computational efficiency, though, the custom solver outperformed the GD method. Future work should improve accuracy and reduce the computation time of the custom solver so it can be implemented in real time.

References

[1] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot Res.*, vol. 29, no. 5, pp. 485–501, 2010.

[2] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," Tech. Rep. CMU-RI-TR-93-20, Robotics Institute, Carnegie Mellon University, 1993.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[4] M. Likhachev, G. Gordon, and S. Thrun, *ARA*: Anytime A* with probable bounds on sub-optimality*, ch. Advances in Neural Information Processing Systems. MIT Press, 2003.

[5] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[6] Y. Wang, "Improved A-search guided tree construction for kinodynamic planning," in *Proc. 2019 ICRA*, pp. 5530–5536, 2019.

[7] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An anytime, replanning algorithm," in *Proc. 2005 ICAPS*, pp. 262–271, 2005.

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot Res.*, vol. 5, no. 1, pp. 417–431, 1986.

[9] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 566–580, Aug. 1996.

[10] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot Res.*, vol. 21, no. 3, pp. 233–255, 2002.

[11] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[13] N. A. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *Proc. 2007 ICRA*, pp. 1617–1624, 2007.

[14] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proc. 2006 IROS*, pp. 5369–5375, 2006.

[15] J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proc. 2006 ICRA*, pp. 2366–2371, 2006.

[16] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Trans. Automat. Control*, vol. 38, pp. 700–716, May 1993.

[17] B. R. Donald and P. Xavier, "Provably good approximation algorithms for optimal kinodynamic planning: robots with decoupled dynamics bounds," *Algorithmica*, vol. 14, pp. 443–479, 1995.

[18] H. M. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Cambridge, MA: MIT Press, 2005.

[19] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[20] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, B. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[21] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Int. J. Robot Res.*, vol. 28, no. 8, pp. 933–945, 2009.

[22] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[23] U. Halder and U. Kalabic, "Time-optimal solution for a unicycle path on SE(2) with a penalty on curvature," in *IFAC Proc.*, vol. 50, no. 1, pp. 6320–6325, 2017.

[24] 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*, pp. 497–516, 1957.

[25] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Automat. Control*, vol. 54, pp. 2318–2327, Oct. 2009.

[26] Q. Gong, W. Kang, and I. M. Ross, "A pseudospectral method for the optimal control of constrained feedback linearizable systems," *IEEE Trans. Automat. Control*, vol. 51, pp. 1115–1129, Jul. 2006.

[27] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guid. Control Dynam.*, vol. 21, pp. 193–207, Mar.-Apr. 1998.

[28] Y. Zhao, Y. Wang, M.-C. Zhou, and J. Wu, "Energy-optimal collision-free motion planning for multiaxis motion systems: An alternating quadratic programming approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, pp. 327–338, Jan. 2019.

[29] L. Berkovitz, *Optimal Control Theory*. Applied Mathematical Sciences, New York: Springer, 1974.

[30] Y. Kanayama, Y. Kimura, F. Miyasaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *Proc. 1990 ICRA*, (Cincinnati, OH), pp. 384–389, May. 1990.

[31] Z.-P. Jiang and H. Nijmeijer, "Tracking control of mobile robots: a case study in backstepping," *Automatica*, vol. 33, pp. 1393–1399, Jul. 1997.

[32] Y. Zhu and U. Özgüner, "Constrained model predictive control for nonholonomic vehicle regulation problem," in *IFAC Proc.*, no. 2, pp. 9552–9557, 2008.