Tunable Trajectory Planner Using G^3 Curves

Alexander Botros and Stephen L. Smith, *Senior Member, IEEE*

***Abstract*—Trajectory planning is commonly used as part of a local planner in autonomous driving. This paper considers the problem of planning a continuous-curvature-rate trajectory between fixed start and goal states that minimizes a tunable trade-off between passenger comfort and travel time. The problem is an instance of infinite dimensional optimization over two continuous functions: a path and a velocity profile. We propose a simplification of this problem that facilitates the discretization of both functions. This paper also proposes a method to quickly generate minimal-length paths between start and goal states based on a single tuning parameter: the second derivative of curvature. Further, we discretize the set of velocity profiles along a given path into a selection of longitudinal jerk way-points along the path. Finally, we repeatedly solve the path and velocity profiles in an iterative fashion. Numerical examples are provided to illustrate the benefits of the proposed methods.**

***Index Terms*—Motion planning, path planning, trajectory optimization.**

I. Introduction

In MOTION planning for autonomous vehicles, a trajectory generation technique is commonly used as part of a local planner [1]. The local planner produces many local trajectories satisfying constraints that reflect the vehicles dynamics, and discards those trajectories that collide with static or predicted dynamic obstacles. The best trajectory is selected according to a metric that captures the efficiency/length of the trajectory, and its smoothness or comfort [2]. The trajectory is used as the reference for a tracking controller for a fixed amount of time, before the local planner produces a new reference.

This work focuses on the first stage of the process outlined above: we address the problem of computing a trajectory between start and goal states that can be used by a local planner to produce a reference trajectory. This is typically done using a simple path model instead of more complex vehicle dynamics [3]. Once a reference trajectory is computed, a tracking controller with a high-fidelity vehicle model is employed to track the reference. Such controllers and local planners are beyond the scope of this work.

Manuscript received January 28, 2021; revised July 16, 2021 and November 2, 2021; accepted January 2, 2022. Date of publication January 11, 2022; date of current version July 12, 2022. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). *(Corresponding author: Alexander Botros.)*

The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: alexander.botros@uwaterloo.ca; stephen.smith@uwaterloo.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIV.2022.3141881.

Digital Object Identifier 10.1109/TIV.2022.3141881

We focus on states of the form $(x, y, \theta, \kappa, v)$ where $(x, y) \in \mathbb{R}^2$ represents the planar location of the vehicle, θ the heading, κ the curvature, and v the velocity. In particular, we seek to compute trajectories that optimize a trade off between comfort and travel time [4], [5], though the methods developed here can be used to consider other features as well.

Comfort of a trajectory is related to the acceleration, rate of change of yaw, and jerk (i.e., the derivative of the acceleration) experienced by a particle (vehicle) sliding along it [5]. A common metric for comfort is the integral of the square of the jerk (IS jerk) over the trajectory [4]. When a vehicle is moving at constant speed, jerk is experienced lateral to the trajectory and is proportional to the derivative of curvature—called the *sharpness*—of the vehicle trajectory. Though the smoothness and comfort of the resulting vehicle motion depends on the reference trajectory *and* the tracking controller, references with low jerk require less effort from the controller and ensure comfort [6], particularly at high speeds.

Often, trajectory planning is treated as a two part spatio-temporal problem [7]. The first sub-problem deals with computing a *path* from start to goal, while the second addresses how this path should be converted into a motion by computing a *velocity profile*. Optimizing a travel time/comfort trade off over a set of paths and velocity profiles ensures the resulting trajectory's adherence to desirable properties like short travel time, and comfort.

This problem is an instance of infinite dimensional optimization in which a non-convex cost (discussed later) is minimized over two continuous functions: path and velocity profile. On one extreme, one could attempt to compute both functions simultaneously. Because of the nature of the problem, conventional techniques like convex optimization or sequential quadratic programming, cannot be used without first modifying the problem. On the other extreme, one could consider a simplified version of the problem that completely decouples path and velocity sub-problems. This approach is widely used (e.g. [3]), but does not consider the influence of the choice of path on the velocity profile (or vice-versa).

We propose a hybrid of these two approaches: we decouple the problem but repeatedly solve the path and velocity profiles in an iterative fashion. In particular, we consider a simplification of the original optimization problem that limits the set of admissible paths to one in which individual elements can easily be distinguished from each other via a single parameter $\bar{\rho} \in \mathbb{R}_{>0}$. We also propose a modification of the techniques employed by [8] to discretize the set of admissible velocity profiles into $N + 1 \geq 5$ way-points for longitudinal jerk. Thus, the two continuous functions, path and velocity profile, over which the our cost is minimized are replaced with $N+2$ constants. We then repeatedly select paths by selecting values for $\bar{\rho}$, and compute the $N+1$ remaining parameters that minimize cost given the selected path, iteratively refining both path and velocity profile.

A. Contributions

This paper contributes to the work of trajectory planning in the following ways:

- Given a set of weights representing a trade off between comfort and travel time, we propose a method of simplifying the resulting infinite-dimensional non-convex optimization problem to one of finite dimension, allowing us to iteratively refine both the path and velocity via coordinate descent.
- To use the technique proposed here, we develop a method to compute paths between start-goal configurations whose second derivative of curvature is piece-wise constant taking values only in $\{0, \pm\bar{\rho}\}$ given a value $\bar{\rho} \in \mathbb{R}_{>0}$.
- Finally, we present a modification of the technique from [8] that allows us to compute a cost-minimizing velocity profile given a path.

B. Related Work

1) Path Planning: In differential geometry a path $(x(u), y(u))$ for a parametrization u is called C^k continuous if it is continuous up to the k^{th} derivative with respect to u. On the other hand, the path is called G^k continuous, or *geometrically* continuous, if there exists a parametrization u for which $(x(u), y(u))$ is C^k continuous [9]. Geometric continuity is a measure of smoothness of a path independent of its parametrization but is equivalent to the familiar notion of C^k continuity when the parameter selected is arc-length [9]. Every path computation technique available will produce a path that is at least G^0 continuous. Typically, a path is said to be G^k if it is G^k continuous, but not G^{k+1} continuous.

In [10], the authors use a cubic polynomial curvature representation. The resulting path does not possess sufficient degrees of freedom to account for bounds on the curvature, resulting in potentially infeasible paths.

Clothoid paths [11] address feasibility issues by constraining curvature and curvature rate. A *clothoid* is a G^2 curve whose instantaneous curvature, is a linear function of the curve's arc-length. In [12], paths are generated as a sequence of clothoid and straight line segments. Because curvature is a linear function of arc-length, the sharpness of these paths (and thus the lateral jerk) is constant. Hence, clothoid paths minimize the maximum squared jerk rather than the IS jerk, a drawback of all G^2 paths.

Requiring that the curvature be twice differentiable, and placing bounds on curvature, curvature rate, and second derivative of curvature results in G^3 paths whose IS jerk (and not simply maximum squared jerk) can be minimized. In [13], G^3 paths are computed by concatenating cubic spline curves and straight lines. The infinitely differentiable spline segments of the path result in longer than necessary arc-lengths. We will show that G^3 paths such that every sub-path is also G^3 (and not G^k for $k > 3$) are the shortest paths for which IS jerk can be minimized.

In [14], the authors introduce continuous curvature rate (CCR), and hybrid curvature rate (HCR) curves. These curves are G^3 curves in which the derivative of curvature with respect to arc-length, σ, is a piece-wise linear, continuous function of the arc-length s, and the second derivative of curvature with respect to arc-length, ρ, is piece-wise constant. Thus every sub-path of these paths is also G^3 (not G^k for $k > 3$). The authors of [14] compute paths between start and goal states by combining CCR/HCR curves and straight lines.

By placing bounds on σ and ρ, the paths developed in [14] have the added benefit of bounding not only lateral/longitudinal jerk, but angular jerk (the rate of change of angular acceleration), as well as lateral/longitudinal snap (the time derivative of jerk). These benefits make HCR/CCR curves particularly attractive for use in the path planning phase of trajectory planning. The techniques developed in [14] will be extended in this paper in four ways. First, the authors of [14] focus on paths whose start and goal states have either 0 or maximum curvature. Though the symmetry resulting from this assumption greatly simplifies path planning, it limits the usefulness of the approach in producing a tracking trajectory. We relax this assumption. Second, the authors of [14] use search techniques to join HCR/CCR curves with straight lines to produce a path. In this paper, we present an algorithm to quickly determine such paths by reducing the path planning problem to a Dubin's-like sub-problem. Third, it is assumed in [14], that the maximum second derivative of curvature with respect to arc-length, $\rho_{\max}$, along a path is known. However, in HCR/CCR curves, the peice-wise constant function $\rho(s)$ dictates the slope of the curvature rate. Therefore, $\rho_{\max}$ should be functions of speed as, for example, a driver may turn the steering wheel very quickly in a parking lot, but not on a highway. In this paper, we compute $\rho_{\max}$ by including it as a parameter denoted $\bar{\rho}$ in an optimization problem that seeks to minimize a trade off between travel time and passenger comfort. Finally, it is assumed in [14], that the velocity of the vehicle is a positive or negative constant. While the work presented herein is limited to positive speeds, we develop a method by which a velocity profile can be computed.

In [3], [15], the authors use pre-computed reference trajectories as part of their local planners. Because trajectories are *pre*-computed, the time required to develop these trajectories is far less important than the space required to store them. Rather than storing the path as a large set of samples (as would be required using a numerical solver), there is an efficient way to store G^3 paths in closed form.

2) Velocity Planning: Once a candidate path has been developed, a velocity profile to describe the motion of the vehicle along that path must be computed. This velocity profile should take into account a tradeoff between comfort and duration of travel. A common technique involves minimizing a cost function that is a weighted sum of undesirable features [4]. In [5], [16], the authors integrate the weighted sum of the squared offset of the trajectory from the center of the road, the squared error in velocity from a desired profile, the squared acceleration, the squared jerk, and the squared yaw rate. A similar technique is employed in [17] with the addition of a penalization on final arc-length. In the context of local trajectory planning, obstacles such as road boundaries, pedestrians, etc. are not considered. Therefore, in this work, we focus on developing a velocity profile and bound $\bar{\rho}$ that minimize a cost function similar to [5], [16], penalizing arc-length, acceleration, jerk, and yaw rate.

A common approach to velocity planning for a path is to adopt a piece-wise quadratic velocity model with potentially discontinuous, piece-wise constant, bounded, longitudinal jerk [18], [19]. The velocity profile we compute is piece-wise cubic in arc-length with continuous, bounded linear longitudinal jerk. We employ this model for three reasons. First, continuous longitudinal jerk allows us to minimize the effect of longitudinal jerk on discomfort as opposed to simply bounding it. Second, selecting thrice continuously differentiable velocity profiles for G^3 paths ensures C^3 continuous trajectories. Finally, quadratic polynomials are a special case of cubic polynomials, thus piece-wise quadratic velocity model can be obtained via the methods presented here if that is desired.

C. Combined Path & Velocity Planning

In [20], the authors propose a trajectory generation technique and MPC-based tracking algorithm. However, the reference paths generated are G^1, and the techniques are limited to lane changes on a highway. In [21], comfortable trajectories are developed for use in highway driving by computing a finite set of s-shaped swerve trajectories. The shape of the trajectories and the assumption that a center-line is known limits this techniques generalizability. More recently, in [22], the authors combine numerical optimization techniques with lattice-based motion planning to compute trajectories for autonomous vehicles in unstructured environments. These results rely on a user-specified set of maneuvers to generate their motion primitives. Similar to our approach, [23] uses coordinate-descent to compute trajectories for quadrotors that minimize energy.

II. Problem Statement

A note on notation: we use $(\cdot)'$ to denote differentiation with respect to arc-length s along a path, while $\dot{(\cdot)}$ represents differentiation with respect to time t. The set of G^3 curves in 2D space is the solution set of the differential equations [14]:

$$\begin{bmatrix} x' \\ y' \\ \theta' \\ \kappa' \\ \sigma' \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \kappa \\ \sigma \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \rho \end{bmatrix}, \tag{1}$$

where $(x, y) \in \mathbb{R}^2, \theta \in [0, 2\pi), \kappa \in \mathbb{R}, \sigma \in \mathbb{R}$ denote the instantaneous position, heading, curvature, and curvature rate along the curve, respectively. Note, σ is the derivative of curvature with respect to arc-length (called the *sharpness*, or *curvature rate*), while ρ represents the second derivative of curvature with respect to arc-length which may be discontinuous. By (1), any function ρ uniquely defines a path P_ρ of path states from a fixed start. Given a function ρ, let s_f denote the final arc-length of the path P_ρ. As in [11], [14], we assume boundary conditions for the path states:

$$\begin{aligned} &x(0) = x_s,\ y(0) \ = y_s,\ \theta(0) \ = \theta_s,\ \kappa(0), = \kappa_s, \\ &x(s_f) = x_g,\ y(s_f) = y_g,\ \theta(s_f) = \theta_g,\ \kappa(s_f) = \kappa_g, \\ &\sigma(0) = \sigma(s_f) = 0, \end{aligned} \tag{2}$$

and physical constraints

$$\begin{aligned} &\kappa(s) \in [-\kappa_{\max}, \kappa_{\max}],\ \forall s \in [0, s_f], \\ &\sigma(s) \in [-\sigma_{\max}, \sigma_{\max}],\ \forall s \in [0, s_f], \\ &\rho(s) \in [-\rho_{\max}, \rho_{\max}],\ \forall s \in [0, s_f]. \end{aligned} \tag{3}$$

For the purposes of this paper, we assume known initial and final velocities v_s, v_g, (resp.) and zero initial and final acceleration. This ensures smooth concatenation of trajectories, and is common in trajectory generation [19]. Under these assumption, a continuous velocity profile is uniquely determined by v'', denoted β.

The functions ρ and β will serve as the variables of an Optimization Problem (OP) that balances travel time and discomfort. For any candidate pair (ρ, β), the resulting path and velocity profile is given by (P_ρ, v_β) where v_β is the velocity profile associated with β. If ρ and β are parameterized by arc-length, we write $(P_\rho(s), v_\beta(s))$, whereas we write $(P_\rho(t), v_\beta(t))$ if ρ and β are parameterized by time. Computing a velocity profile v_β for a G^3 curve P_ρ induces a re-parametrization of the curve. For the resulting trajectory to be thrice continuously differentiable (w.r.t. t), we require that β be a continuous functions of t.

Using a technique similar to [4], [5], [16], [17], we define a cost penalizing a trade off between travel time and comfort:

$$\begin{aligned} C(P_\rho(t), v_\beta(t)) &= \int_0^{t_f} (w_a C_a + w_{\mathcal{J}} C_{\mathcal{J}} + w_y C_y + w_t) dt, \\ C_a &= |a_N(t)|^2 + |a_T(t)|^2, C_y = \left| \frac{d\theta(t)}{dt} \right|^2, \\ C_{\mathcal{J}} &= \mathcal{J}_N(t)^2 + \mathcal{J}_T(t)^2. \end{aligned} \tag{4}$$

Here, $C_a, C_{\mathcal{J}}, C_y$ represent the squared magnitude of the acceleration $\boldsymbol{a}$ (expressed using normal and tangential components a_N, a_T), the squared magnitude of jerk $\boldsymbol{\mathcal{J}}$ (expressed using normal and tangential components $\mathcal{J}_N, \mathcal{J}_T$), and the squared magnitude of yaw rate for the trajectory $(P_\rho(t), v_\beta(t))$. These costs are weighted with constants $w_a, w_{\mathcal{J}}, w_y$ representing the relative importance of each feature to a user. We refer to the terms $\int_0^{t_f} w_m C_m, m \in \{a, \mathcal{J}, y, t\}$ as the integral squared (IS) acceleration, IS jerk, IS yaw, and time cost, respectively.

The method we employ to compute a velocity profile requires that the limits of integration of the cost be fixed for a fixed path P_ρ. Therefore, we re-parameterize the cost (4) in terms of arc-length s. Letting $\boldsymbol{n}, \boldsymbol{\tau}$ represent the unit normal and unit tangent vectors, respectively, we observe:

$$\begin{aligned} \boldsymbol{a} &= a_N \boldsymbol{n} + a_T \boldsymbol{\tau} = |\kappa| v^2 \boldsymbol{n} + a_T \boldsymbol{\tau} \\ \boldsymbol{\mathcal{J}} &= \dot{\boldsymbol{a}} = (3 v a_T |\kappa| + v^3 \bar{\sigma}) \boldsymbol{n} + (\dot{a}_T - \kappa^2 v^3) \boldsymbol{\tau}, \end{aligned} \tag{5}$$

where $\bar{\sigma} = (|\kappa|)'$. The expression for $\boldsymbol{\mathcal{J}}$ was obtained by differentiating $\boldsymbol{a}$ with respect to time (using the Frenet-Serret formula to integrate the normal and tangent vectors, see [24]) and observing $\dot{m} = m'(ds/dt) = m'v$ for any path state m. Finally, letting $\alpha = v,' \beta = \alpha,' b = \dot{a}_T$, we observe

$$a_T = \alpha v,\ \dot{a}_T = b = v(\beta v + \alpha^2). \tag{6}$$

Combining (4), (5), and (6), and integrating with respect to s instead of t yields:

$$\begin{aligned} C(P_\rho(s), v_\beta(s)) &= \int_0^{s_f} w_a\tilde{C}_a + w_{\mathcal{J}}\tilde{C}_{\mathcal{J}} + w_y\tilde{C}_y + w_t\tilde{C}_t ds, \\ \tilde{C}_a &= v^3|\kappa|^2 + \alpha^2 v,\ \tilde{C}_y = |\kappa|^2 v,\ \tilde{C}_t = v^{-1} \\ \tilde{C}_{\mathcal{J}} &= v^3(3\alpha|\kappa| + \bar{\sigma}v)^2 + v(\beta v + \alpha^2 - \kappa^2 v^2)^2. \end{aligned} \tag{7}$$

We now use this cost together with the constraints developed earlier to state the new OP that is the focus of this paper:

$$\begin{aligned} \min_{\rho(s),\beta(s)} \quad & C(P_\rho(s), v_\beta(s)) \\ s.t. \quad & \text{constraints } (1), (2), (3), (6) \\ & \begin{bmatrix} v'_\beta(s) & \alpha'(s) \end{bmatrix}^T = \begin{bmatrix} \alpha(s) & \beta(s) \end{bmatrix}^T \\ & v_\beta(0) = v_s,\ v_\beta(s_f) = v_f,\ \alpha(0) = \alpha(s_f) = 0, \\ & v_\beta(s) \in (0, v_{\max}], a(s) \in [-a_{\max}, a_{\max}], \\ & b(s) \in [-b_{\max}, b_{\max}],\ \forall s \in [0, s_f]. \end{aligned} \tag{8}$$

Let $R \times B$ be the set of all $(\rho(s), \beta(s))$ whose associated trajectories are feasible solutions to (8). Though this work focuses on optimizing a trade-off between travel time and comfort, the techniques developed herein can be extended to account for other trajectory features as well by simply adding and/or removing features in the cost function (8). The techniques developed here require only that the integrand of the cost be polynomial in $v(s), \alpha(s), \beta(s)$ and not an explicit function of ρ. If $w_a = w_{\mathcal{J}} = w_y = 0, w_t = 1$ and $v(s) = 1, \forall s \in [0, s_f]$, then (8) reduces to the OP from [14]:

$$\begin{aligned} &\min_\rho\ s_f \\ &s.t. \text{ constraints } (1), (2), (3) \end{aligned} \tag{9}$$

III. Approach

The optimization problem in (8) is an instance of infinite dimensional, non-convex optimization. The non-convexity of (8) is due to the cost $\tilde{C}_{\mathcal{J}}$. Further, the non-holonomic constraints (1) require the evaluation cubic Fresnel integrals for which there is no closed form solution. For these reasons, we propose an approach that simplifies (8). In this section we present the high-level idea behind our technique, beginning with a motivating Theorem.

Theorem III.1 (Optimal G^3 paths): If (ρ^*, β^*) is a solution to the OP (8), then ρ^* is piece-wise constant.

The result of this Theorem follows directly from the observation that ρ does not appear explicitly in the cost of (8), and appears linearly in the constraints. Thus the Hamiltonian is linear in ρ, and the result follows from Pontryagin's Minimum Principle [25, Chapter 12].

By Theorem III.1, we conclude that G^3 paths such that every sub-path is also G^3 (not G^k for $k > 3$) are the shortest paths for which IS jerk can be minimized without simply bounding the maximum squared jerk. However, there are many such paths connecting start and goal path states. The first part of our technique is to simplify the OP (8) by considering only those G^3 curves $P_{\hat{\rho}}$ such that there exists a constant $\bar{\rho} \leq \rho_{\max}$ for which $P_{\hat{\rho}}$ solves (9) when $\rho_{\max}$ is replaced with $\bar{\rho}$. That is, we approximate a solution to (8) by solving

$$\begin{aligned} &\min_{\bar{\rho} \leq \rho_{\max}} \left(\min_{\beta(s) \in B} C(P_{\hat{\rho}}(s), v_\beta(s)) \right), \\ &s.t.,\ \hat{\rho}(s) \in R,\ \text{solves (9) for } \rho_{\max} = \bar{\rho}. \end{aligned} \tag{10}$$

In words, (10) approximates the OP in (8) by replacing the continuous function $\rho(s)$ with a constant $\bar{\rho}$. The path associated with $\bar{\rho}$ is a shortest G^3 path $P_{\hat{\rho}}(s)$ such that $\hat{\rho}(s)$ is bounded in magnitude by $\bar{\rho}$. From the results in [14], we observe that $\hat{\rho}$ is a function taking values only in $\{\pm\bar{\rho}, 0\}$. This approach has two major advantages: first, we have replaced the continuous function $\rho(s)$ in (8) with constant $\bar{\rho}$ while still maintaining the optimal form of the G^3 path (piece-wise constant in $\hat{\rho}$). Second, by tuning $\bar{\rho}$, we can still produce G^3 curves with both sharp (typically favored by users who value short travel times) and gradual curvature functions (favored by users who value comfortable trajectories).

The second part of our technique involves simplifying (10) yet further by replacing the continuous function $\beta(s)$ with a decision vector. Similar to [8], we discretize the arc-length along the path $P_{\hat{\rho}}$ into $N + 1 \in \mathbb{N}_{\geq 5}$ fixed points $\{s_0, \ldots, s_N\}$. We then replace the continuous function $\beta(s)$ with the decision vector $\boldsymbol{B} = [b_i, i = 0, \ldots, N]$ where b_i is the longitudinal jerk at arc-length s_i. These way-points b_i can then be used to produce a continuous, piece-wise linear function $\beta(s)$ by connecting sequential values b_i, b_{i+1} with straight lines. In our implementation we use $N = 10$, which works well in practice.

Thus far, our approach has simplified (8) by replacing the two continuous functions ρ, β with $N + 2$ constants $\bar{\rho}, b_i, i = 0, \ldots, N$. The final stage of our technique is to compute the cost-minimizing values of these constants via coordinate descent with non-convex cost: we iteratively select values of $\bar{\rho}$, compute the associated path $P_{\hat{\rho}}(s)$ for each selection, and then compute the cost minimizing vector $\boldsymbol{B}$ given the fixed path. For each $\bar{\rho}$, the decision vector $\boldsymbol{B}$ can be efficiently computed since the integrand of the cost C in (10) is a polynomial expression of v, α, β for a fixed path, facilitating the use of second order optimization techniques (Section V).

Unfortunately, the same techniques are less effective when computing $\bar{\rho}$. Even infinitesimal changes to the value $\bar{\rho}$ can result in large changes to the path $P_{\hat{\rho}}$ that solves (9) given $\rho_{\max} = \bar{\rho}$, the velocity profile v_β that is optimal given the path $P_{\hat{\rho}}$, and therefore to the cost $C(P_{\hat{\rho}}, v_\beta)$. It is therefore difficult to derive analytical expressions for the first and second derivatives of cost with respect to $\bar{\rho}$. Further, it has been found that estimating

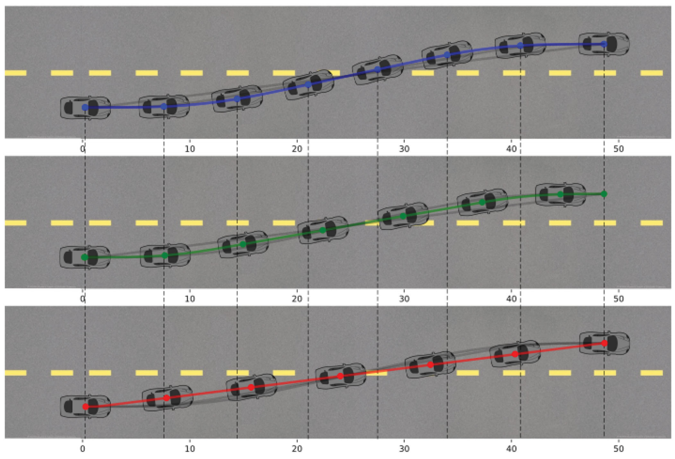

Fig. 1. Example reference trajectories for a lane change maneuver given three minimization objectives: discomfort (top), time (bottom), and a mix between time and discomfort (mid). Initial and final speeds are fixed at 10 m/s, cars are drawn ever 0.75 seconds.

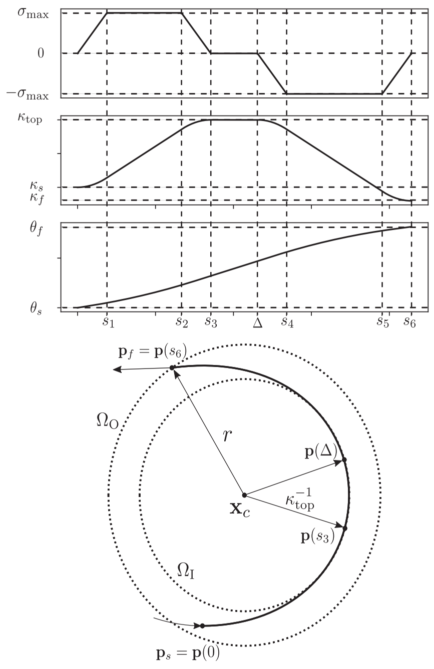

Fig. 2. Basic G^3 curve. **Top**: The functions $\sigma(s)$ (top), $\kappa(s)$ (mid), and $\theta(s)$ (bottom). **Bottom**: The resulting curve in the x, y plane from start configuration $\boldsymbol{p}_s$ to final configuration $\boldsymbol{p}_f$. Image also appears in [14].

these values numerically can be time consuming. For these reasons, we appeal to derivative-free optimization techniques like Py-BOBYQA [26]. In order to use the methods described above, we require two sub-techniques: The first solves (9) for any given values $\sigma_{\max}, \kappa_{\max}$, and $\rho_{\max} = \bar{\rho}$ (Section IV). The second computes the optimal way-points $\alpha(s_i)$ for any fixed path (Section V).

IV. Computing G^3 Paths

In this section, we describe how to compute G^3 paths that solves (9) for known values of $\rho_{\max}$. The solution $\rho(s)$ to (9) is piece-wise constant, taking values only in $\{0, \pm\rho_{\max}\}$ [14]. We begin with an investigation of single G^3 curves, and then present a technique to connect G^3 curves to form G^3 paths.

A. *Single G^3 Curves*

In this section, we describe a single G^3 curve. The results follow closely those in [14], but have been re-derived to include initial and final curvatures that are not constrained to $\{0, \pm\kappa_{\max}\}$. For simplicity, we use the same notation as [14] to outline the general form of a G^3 curve.

Following Fig. 2, a G^3 curve performing a left-hand turn begins at an arc-length $s = 0$ from an initial state $\boldsymbol{p}_s = \boldsymbol{p}(0)$ where $\rho(0) = \rho_{\max}$. At $s = s_1$, the curvature rate reaches its maximum allowable value $\sigma_{\max}$ and remains constant until $s = s_2$ at which point $\rho(s_2) = -\rho_{\max}$ and $\sigma(s)$ decreases linearly to 0. At $s = s_3$, $\sigma(s_3) = 0$ and the curvature reaches its *top* value $\kappa(s_3) = \kappa_{\text{top}}$ where it remains until $s = \Delta$. Here, $\rho(\Delta) = -\rho_{\max}$, and $\sigma(s)$ decreases linearly until $\sigma(s_4) = -\sigma_{\max}$. At $s = s_5$, $\rho(s_5) = \rho_{\max}$, and $\sigma(s)$ increases to 0 at $s = s_6 = s_f$. Here, $\kappa(s_6) = \kappa_f$. The circle Ω_{I} is the circle centered at $\boldsymbol{x}_c$ with radius κ_{top}^{-1}. A right-hand turn may be achieved by taking the mirror image of $\sigma(s)$ about the horizontal axis.

Given $\rho_{\max}, \sigma_{\max}, \kappa_{\text{top}}, \Delta$, the values of the *switching arc-lengths* s_i, $i = 1, \ldots, 6$ such that $\kappa(0) = \kappa_s, \kappa(s_3) = \kappa_{\text{top}}, \kappa(s_6) = \kappa_f$, are given by:

$$
\begin{aligned}
s_1 &= \begin{cases} \frac{\sigma_{\max}}{\rho_{\max}}, & \text{if } |\kappa_{\text{top}} - \kappa_s| > \frac{(\sigma_{\max})^2}{\rho_{\max}} \\ \sqrt{\frac{|\kappa_{\text{top}} - \kappa_s|}{\rho_{\max}}}, & \text{otherwise} \end{cases} \\
s_2 &= \begin{cases} \frac{|\kappa_{\text{top}} - \kappa_s|}{\sigma_{\max}}, & \text{if } |\kappa_{\text{top}} - \kappa_s| > \frac{(\sigma_{\max})^2}{\rho_{\max}} \\ s_1, & \text{otherwise} \end{cases} \\
s_3 &= s_1 + s_2, \\
s_4 &= \Delta + \begin{cases} \frac{\sigma_{\max}}{\rho_{\max}}, & \text{if } |\kappa_{\text{top}} - \kappa_f| > \frac{(\sigma_{\max})^2}{\rho_{\max}} \\ \sqrt{\frac{|\kappa_{\text{top}} - \kappa_f|}{\rho_{\max}}}, & \text{otherwise} \end{cases} \\
s_5 &= \Delta + \begin{cases} \frac{|\kappa_{\text{top}} - \kappa_f|}{\sigma_{\max}}, & \text{if } |\kappa_{\text{top}} - \kappa_f| > \frac{(\sigma_{\max})^2}{\rho_{\max}} \\ s_4, & \text{otherwise} \end{cases} \\
s_6 &= s_4 + s_5 - \Delta.
\end{aligned} \tag{11}
$$

Further,

$$
\kappa(s) = \begin{cases} \kappa_s \pm 0.5\rho_{\max} s^2, \ s \in [0, s_1] \\ \kappa(s_1) \pm \rho_{\max} s_1 (s - s_1), \ s \in [s_1, s_2] \\ \kappa(s_2) \pm 0.5\rho_{\max}(s - s_2)(s_2 + 2s_1 - s), \ s \in [s_2, s_3] \\ \kappa_{\text{top}}, \ s \in [s_3, \Delta] \\ \kappa_{\text{top}} \mp 0.5\eta\rho_{\max}(s - \Delta)^2, \ s \in [\Delta, s_4] \\ \kappa(s_4) \mp \eta\rho_{\max}(s_4 - \Delta)(s - s_4), \ s \in [s_4, s_5] \\ \kappa(s_5) \mp 0.5\eta\rho_{\max}(s - s_5)(s_5 - 2\Delta + 2s_4 - s). \end{cases} \tag{12}
$$

The top sign of each "$\pm, \mp$" is used if $\kappa_{\text{top}} \geq \kappa_s$, while the bottom sign is used otherwise and $\eta = 1$ if $\sigma(s_1)\sigma(s_4) < 0$ and

$\eta = -1$ otherwise. A value $\eta = -1$ is used when the curvature function is either monotonically increasing or decreasing over the entire curve. By (1), the path state vector $\boldsymbol{p}$ along a G^3 curve whose curvature is given by (12) is:

$$\boldsymbol{p}(s) = \begin{bmatrix} x(s) \\ y(s) \\ \theta(s) \\ \kappa(s) \end{bmatrix} = \begin{bmatrix} x_s + \int_0^s \cos(\theta(\tau))d\tau \\ y_s + \int_0^s \sin(\theta(\tau))d\tau \\ z_0 + z_1 s + z_2\, s^2 + z_3\, s^3 \\ \kappa(s) \end{bmatrix}, \tag{13}$$

where z_0, z_1, z_2, z_3 are obtained by integrating $\kappa(s)$ in (12). As Fig. 2 implies, the point $\boldsymbol{x}_c$, and radius r, are given by

$$\boldsymbol{x}_c = \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} x(s_3) - \kappa_{\text{top}}^{-1} \sin(\theta(s_3)) \\ y(s_3) + \kappa_{\text{top}}^{-1} \cos(\theta(s_3)) \end{bmatrix}, \tag{14}$$

$$r = ||(x(s_6), y(s_6)) - (x_c, y_c)||. \tag{15}$$

If $\rho_{\max}, \sigma_{\max}$ are known, the values $\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta$ uniquely define a G^3 curve denoted $G(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta)$ which is the set of path states $\boldsymbol{p}(s)$ given by (13) from $s = 0$ to $s = s_6$. A G^3 path P_ρ, solving (9) is a concatenation of such curves and straight lines. Let $G_i^{s_j}(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta)$, be the value of path state i at arc-length s_j. In the next section, we present our technique for solving (9) by concatenating G^3 curves and straight lines.

B. Connecting G^3 Curves With a Straight Line

This section proposes a technique to connect G^3 curves using a straight line via a reduction of the G^3 curve path planning problem (9) to a Dubin's-like path planning problem with two different minimal turning radii. This can be solved quickly using common tangent lines. For fixed values $\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f = 0$, consider the set of curves

$$\mathcal{G} = \{G(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f = 0, \Delta) \;:\; \Delta \geq s_3\}. \tag{16}$$

Members of $\mathcal{G}$ are G^3 curves with final curvature 0 enabling us to connect G^3 curves with straight lines while preserving curvature continuity. The assumption that $\sigma(s_6) = 0$ ensures that G^3 curves can be connected with straight lines while preserving continuous curvature rate.

For each pair $(\boldsymbol{p}_s, \kappa_{\text{top}})$ the set $\mathcal{G}$ in (16), contains as an element, the curve $G(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f = 0, \Delta = s_3)$ whose curvature immediately decreases upon reaching κ_{top} ($\Delta = s_3$). Further, for fixed values $\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f = 0$, the value of s_3 given in (11) is independent of Δ, implying that every curve in $\mathcal{G}$ shares the same switching arc-length s_3. The same holds for the center $\boldsymbol{x}_c$ given in (14). Therefore, the values $\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f = 0$, which are shared by all curves in $\mathcal{G}$, are sufficient to determine s_3 and $\boldsymbol{x}_c$. Finally, note that each curve in $\mathcal{G}$ is coincident with every other curve in $\mathcal{G}$ for all $s \leq s_3$. The following Lemma illustrates the relationship between Δ and $G_\theta^{s_6}(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta)$.

Lemma IV.1 (Final Heading): For each set of curves $\mathcal{G}$ in (16), there is a unique solution Δ to the equation $G_\theta^{s_6}(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta) = \theta_f$ such that s_6 is minimized:

$$\Delta = s_3 + (\theta_g - G_\theta^{s_6}(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta = s_3))\kappa_{\text{top}}^{-1}. \tag{17}$$

where s_3, s_6 are given by (11) for the curve $G(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta)$.

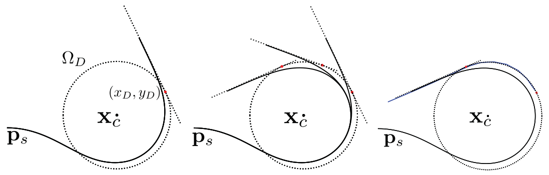

Fig. 3. (Left) Representative circle Ω_D, and point (x_D, y_D). (Mid) Illustration of Theorem IV.2. (Right) Illustration of Step 5, with partial Dubin's path $\mathcal{D}$ (blue) and corresponding G^3 curve $\hat{G}_i$ (black).

The proof of Lemma IV.1 can be found in the Appendix. Lemma IV.1 implies that the final headings of elements in $\mathcal{G}$ varies linearly with their values of Δ. This Lemma motivates the following key Theorem that will be heavily leveraged later.

Theorem IV.2 (Reducing Theorem): Given a set of curves $\mathcal{G}$ in (16), let x_f, y_f, θ_f denote the final x, y, and θ values of any curve $G(\boldsymbol{p}_s, \kappa_{\text{top}}, 0, \Delta) \in \mathcal{G}$. Let m denote the slope of a line induced by θ_f: $m = \tan\theta_f$. The shortest distance from the point $\boldsymbol{x}_c$ defined in (14) to the line passing through (x_f, y_f) whose slope is m is a constant with respect to Δ.

The proof of Theorem IV.2 can be found in the Appendix but the result is shown in Fig. 3 (Mid). Here, red dots correspond to the points on the lines containing the final path states of each curve in $\mathcal{G}$ that are closest to $\boldsymbol{x}_c$. Observe that they lie on a circle centered at $\boldsymbol{x}_c$.

Definition IV.3 (Representative Circle): Given a set of curves $\mathcal{G}$ in (16), and using the same notation as Theorem IV.2, the *representative circle* Ω_D of a curve in $\mathcal{G}$, is the circle centered at $\boldsymbol{x_c} = (x_c, y_c)$ containing the point (x_D, y_D) where

$$x_D = \begin{cases} \frac{(mx_f - y_f + m^{-1}x_c + y_c)}{m + m^{-1}}, & \text{if } \theta_f \neq \pi/2 \\ x_f, & \text{otherwise} \end{cases}$$

$$y_D = \begin{cases} -m^{-1}(x_D - x_c) + y_c, & \text{if } \theta_f \neq \pi/2 \\ y_c & \text{otherwise.} \end{cases} \tag{18}$$

That is, for the straight line containing (x_f, y_f) whose slope is $\tan\theta_f$, the representative circle is the circle centered at $\boldsymbol{x}_c$ containing the point on the line closest to $\boldsymbol{x}_c$. Observe that Theorem IV.2 implies that all curves in a set $\mathcal{G}$ have coincident representative circles. An example representative circle, and corresponding points (x_D, y_D) is show in Fig. 3 (Left). In this figure, the black curve is $G(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f = 0, \Delta = s_3)$. Theorem IV.2 implies that every G^3 curve in a given set $\mathcal{G}$ in (16) has the same representative circle.

We now present our algorithm for computing G^3 paths that solve (9) given a known value of $\rho_{\max}$. The high-level idea is to begin by constructing two G^3 curves originating from both start and goal (with reverse orientation), and then to connect these curves using either a straight line or another curve. For simplicity, let $\boldsymbol{p}_1 = \boldsymbol{p_s}, \boldsymbol{p}_2 = \boldsymbol{p}_g$, and let $\boldsymbol{p}_3$ denote $\boldsymbol{p}_g$ with reverse orientation. That is, $\boldsymbol{p}_3 = (x_g, y_g, \theta_g + \pi, -\kappa_g)$.

Step 1: For each of the possible start and goal curve orientations: (R,L), (R,R), (L,R), (L, L) where R is a right-hand turn and L a left, perform the following steps, and then discard all but the choice with minimum final arc-length.

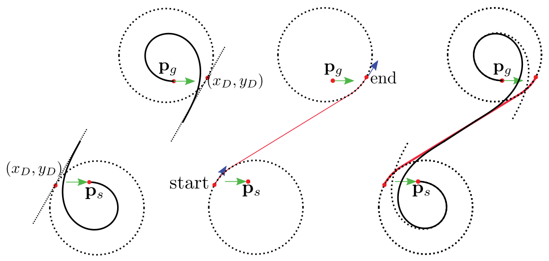

Fig. 4. (Left) Representative circles around $\boldsymbol{p}_s$ and $\boldsymbol{p}_g$ in reverse. (Mid) Dubins'-like solution between start and goal states. (Right) Solution path.

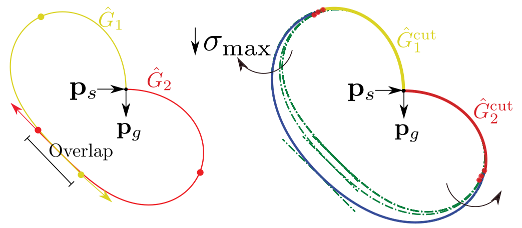

Fig. 5. Step 7. (Left) Illustration of overlap: curve $P2$ terminates behind $P1$. (Right) Stretching of curves $P1, P2$ by decreasing $\sigma_{\max}$ until $P1, P2$ can be connected by third curve.

Step 2: Let $G_i = G(\boldsymbol{p}_i, |\kappa_{\text{top}_i}| = \kappa_{\max}, \kappa_{f_i} = 0, \Delta_i = s_{3_i})$, $i = 1,3$ be two curves originating from $\boldsymbol{p}_1, \boldsymbol{p}_3$, respectively, whose top curvature reaches its maximum allowable magnitude $\kappa_{\max}$ (with sign given by the chosen orientation), and where s_{3_i} is given by (11). This step is illustrated by the black curves in Fig. 4 (Left) for orientation (R,L). Let $\mathcal{G}_i$ denote the two sets of curves from (16) with $G_i \in \mathcal{G}_i$.

Step 3: Compute the points $(x_{D_i}, y_{D_i}), i = 1,3$ for curves G_i on the representative circles of each curve. This step is illustrated in Fig. 4 (Left). Here, the dotted circles represent the representative circles of each curve, and the red dots on these circles represent the points $(x_{D_i}, y_{D_i}), i = 1,3$.

Step 4: Compute the Dubin's-type path $\mathcal{D}$ that consists of two curves and a straight line connecting $(x_{D_i}, y_{D_i}), i = 1,3$ with minimum turning radius on each curve $r_{D_i} = ||(x_{D_i}, y_{D_i}) - (x_{c_i}, y_{c_i})||$. This step is illustrated in Fig. 4 (mid). Note that curves $G_i, i = 1,3$ may have different values of r_{D_i}, resulting in a Dubin's problem with different minimum turning radii. This may not have a solution of the form described here, which will be addressed this in the next section.

Step 5: Compute the angle θ_f inscribed by the straight portion of $\mathcal{D}$ and the horizontal axis. Compute the value of $\hat{\Delta}_i$ from (17) such that the curves $\hat{G}_i = G(\boldsymbol{p}_i, \kappa_{\text{top}_i}, \kappa_{f_i}, \hat{\Delta}_i) \in \mathcal{G}_i$ $i = 1,3$ have final headings $\theta_f, \theta_f + \pi$, respectively. By Theorem IV.2, curves $G, \hat{G}_i$ have coincident representative circles which, by construction, have radii r_{D_i} the minimum turning radii of $\mathcal{D}$. Therefore, the final states $(x_{f_i}, y_{f_i}, \theta_{f_i}, \kappa_i = 0, \sigma_{f_i} = 0)$ of the curves $\hat{G}_i$ are on the path $\mathcal{D}$. This is illustrated in Fig. 3 (Right). Connect the curves $\hat{G}_i = G(\boldsymbol{p}_i, \kappa_{\text{top}_i}, \kappa_{f_i} = 0, \Delta_i)$ with a straight line. This step is illustrated in Fig. 4 (Right). Solid black, and dotted black lines represent $\hat{G}_i, G_i$, respectively.

Step 6 (Looping): If $\sigma_{\max}, \rho_{\max}$ are too small compared to $\kappa_{\max}$, then one or both of the curves G_i in Step 2 will *loop* [14]. A curve $G(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta)$ with initial and final headings θ_s, θ_f (resp.), will loop if $|\theta_f - \theta_s| < G_\theta^{s_6}(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta = s_3)$. If such looping occurs on curve $i = 1,3$, we decrease the magnitude of the top curvature κ_{top_i}, and return to Step 2. Minimum arc-length is guaranteed by computing the smallest magnitude values of κ_{top_i} that ensure a connection in Step 5.

C. Connecting G^3 Curves With a G^3 Curve

Two problems may arise in Steps 1–6 above. The first, we call *overlap*. Overlap arises when the endpoints of the two curves $\hat{G}_i, i = 1,3$ computed in Step 5 cannot be connected with a straight line that preserves orientation. The phenomenon is illustrated in Fig. 5 (Left), and can occur if the centers $\boldsymbol{x}_{c_i}$ of the representative circles of the curves G_i in Step 2 are too close together ($< r_1 + r_3$ where r_i is the radius (15) for the curve $\hat{G}_i$). Observe that overlap is a result of $\sigma_{\max}, \rho_{\max}$ being small relative to κ_{top_i}: by the time the curvature of curve $\hat{G}_i$ changes from κ_{top_i} to 0, the final configuration has already passed that of the second curve and cannot be connected with a straight line that preserves orientation. Therefore, if overlap occurs we attempt to find a curvature other than 0 that can be used to connect $\hat{G}_i$. This is summarized here:

Step 7 (Overlap): If overlap occurs, we cut the curves $\hat{G}_i, i = 1,3$ at $s = \Delta_i$. That is, let $\hat{G}_1^{\text{cut}} = G(\boldsymbol{p}_i, \kappa_{\text{top}_i}, \kappa_{f_i} = \kappa_{\text{top}_i}, \Delta_i)$, and we attempt to connect $\hat{G}_1^{\text{cut}}, i = 1,3$ with a third G_3 curve. If no such third curve exists, we slowly decrease the value of $\sigma_{\max}$, and go back to Step 1. Fig. 5 (right) illustrates an example where $\sigma_{\max}$ is decreased until $\hat{G}_i^{\text{cut}}, i = 1,3$ can be connected with a third curve. Minimum arc-length is preserved by finding the largest feasible value of $\sigma_{\max}$ that allows for such a connection.

The second problem arises if no Dubins' solution can be found in Step 4. Similar to overlap, this arises if the centers $\boldsymbol{x}_{c_i}$ are too close together. For this reason, we again decrease the value of $\sigma_{\max}$ and return to Step 1. At the end of Steps 1–7, the process described above will have computed a G^3 curve $P_{\rho_{\max}}(s)$ that solves (9).

Remark IV.4: Critically, to store a solution path between start and goal states it is sufficient to store only the values $\rho_{\max}$, the top curvature κ_{top_i} and the value Δ_i for each of the potentially three curves, and the final curvatures of the first and last curves (i.e., 0 if $\hat{G}_1, \hat{G}_2$ can be connected with a straight line, and κ_{top_i} otherwise). Indeed, from (1), (11), (12) the resulting path can easily be obtained from these 9 values.

V. Computing Velocity Profiles

In this section we describe how to compute a longitudinal jerk profile $\beta(s)$ that minimizes the cost function in (10) over a fixed G^3 path $P_\rho(s)$. Similar to the procedure outlined in [8], we approximate the continuous optimization problem by a discrete problem that asymptotically approaches the continuous version.

Given a fixed path $P_\rho(s)$ with final arc-length s_f, the high-level idea is to select $N+1$ arc-lengths s_i from $s_0 = 0$ to $s_N = s_f$ where $N \geq 4$, and then to select $N+1$ values $\beta(s_i)$. The value of $\beta(s)$ between sequentially sampled arc-lengths s_i, s_{i+1} is the linear function passing through $\beta(s_i), \beta(s_{i+1})$.

Let $\boldsymbol{s} = [s_0, \ldots, s_N]$ denote $N+1$ arc-lengths with $s_0 = 0, s_N = s_f$. Further, let $\boldsymbol{B} = [b_0, b_1, \ldots, b_N]$ denote a vector of $N+1$ real numbers. This vector will serve as values of $\beta(s)$ at fixed arc-lengths $s_0, s_1, \ldots, s_N$. Given any vector $\boldsymbol{B}$, we may compute a velocity profile, $v^{\boldsymbol{B}}(s)$, longitudinal acceleration profile $\alpha^{\boldsymbol{B}}(s)$, and longitudinal jerk profile $\beta^{\boldsymbol{B}}(s)$ recursively as follows:

$$\begin{aligned} v^{\boldsymbol{B}}(s) &= v^{\boldsymbol{B}}(s_i) + \alpha^{\boldsymbol{B}}(s_i)(s - s_i) \\ &\quad + \frac{b_i(s-s_i)^2}{2} + \left(\frac{b_{i+1} - b_i}{s_{i+1} - s_i}\right)\frac{(s-s_i)^3}{6}, \\ \alpha^{\boldsymbol{B}}(s) &= \alpha^{\boldsymbol{B}}(s_i) + b_i(s - s_i) + \left(\frac{b_{i+1} - b_i}{s_{i+1} - s_i}\right)\frac{(s-s_i)^2}{2}, \\ \beta^{\boldsymbol{B}}(s) &= b_i + \left(\frac{b_{i+1} - b_i}{s_{i+1} - s_i}\right)(s - s_i), \end{aligned} \tag{19}$$

for all $s \in [s_i, s_{i+1}]$ and for all $i = 0, \ldots, N-1$. Observe that no bounds are placed on $\frac{d\beta^{\boldsymbol{B}}(s)}{ds}$. Therefore, for sufficiently small values of $s_{i+1} - s_i, i = 0, \ldots, N-1$ and large values of $\frac{d\beta^{\boldsymbol{B}}(s)}{ds}$, the equations in (19) can be used to model continuous piece-wise quadratic velocity profiles with piece-wise constant $\beta(s)$. Using the notation

$$\begin{aligned} S_1(i) &= \frac{1}{6}(s_{i-1} - s_i)^2, \\ S_2(i) &= \frac{1}{6}(s_{i-2} - s_i)(s_{i-2} + s_{i-1} - 2s_i), \\ S_3(j,i) &= \frac{1}{6}(s_{j-1} - s_{j+1})(s_j + s_{j-1} - 3s_i + s_{j+1}), \\ S_4(i) &= \frac{1}{2}(s_i - s_{i-1}), \\ S_5(i) &= \frac{1}{2}(s_{i+1} - s_{i-1}), \end{aligned}$$

we compute $v^{\boldsymbol{B}}, \alpha^{\boldsymbol{B}}, \beta^{\boldsymbol{B}}$ at each value s_i as linear combinations of the vector $\boldsymbol{B}$:

$$\begin{aligned} v^{\boldsymbol{B}}(s_i) &= v_s + b_i S_1(i) + b_{i-1} S_2(i) + \sum_{j=1}^{i-2} b_j S_3(j,i), \\ \alpha^{\boldsymbol{B}}(s_i) &= b_i S_4(i) + \sum_{j=1}^{i-1} b_j S_5(j),\ \beta^{\boldsymbol{B}}(s_i) = b_i. \end{aligned} \tag{20}$$

We set $b_0 = 0, b_N = 0$. This is to ensure that when two trajectories are concatenated by a local planner, the resulting β-profile is continuous. Finally, we compute b_{N-2}, b_{N-1} in terms of $b_i, i = 1, \ldots, N-3$ to ensure that $\alpha(s_f) = 0, v(s_f) = v_f$. This is done by solving the linear system of equations $v^{\boldsymbol{B}}(s_N) = v_f, \alpha^{\boldsymbol{B}}(s_N) = 0$, for b_{N-1}, b_{N-2} where $v^{\boldsymbol{B}}(s_N), \alpha^{\boldsymbol{B}}(s_N)$ are given by (20). Equation (20) also facilitates the development of linear inequalities for the vector $\boldsymbol{B}$ to encode the constraints $0 \le v^{\boldsymbol{B}}(s_i) \le v_{\max}, |\alpha^{\boldsymbol{B}}(s_i)| \le \alpha_{\max}, |\beta^{\boldsymbol{B}}(s_i)| \le \beta_{\max}$.

Let $\boldsymbol{B}' = [b_1, \ldots, b_{N-3}]$ denote a decision vector of waypoint values of longitudinal jerk. From $\boldsymbol{B}'$, we compute $\boldsymbol{B} = [b_0, b_1, \ldots, b_N]$ by setting $b_0 = b_N = 0$, and computing b_{N-1}, b_{N-2} by solving a system of linear equations to ensure the boundary constraints of the velocity profile. To select a decision vector, we solve the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{B}' \in [-\beta_{\max}, \beta_{\max}]^{N-3}} &\ C(P_\rho(s), v_{\beta^{\boldsymbol{B}}}(s)) \\ s.t.,\ &\forall s_i \in \boldsymbol{s}, \\ &0 \le v^{\boldsymbol{B}}(s_i) \le v_{\max}, \\ &|\alpha^{\boldsymbol{B}}(s_i)| \le \alpha_{\max},\ |\beta^{\boldsymbol{B}}(s_i)| \le \beta_{\max}. \end{aligned} \tag{21}$$

For a fixed path $P_\rho(s)$, the integrand of $C(P_\rho(s), v_{\beta^{\boldsymbol{B}}}(s))$ in (7) reduces to a non-convex polynomial expression of $v^{\boldsymbol{B}}(s), \alpha^{\boldsymbol{B}}(s), \beta^{\boldsymbol{B}}(s)$. Using $v^{\boldsymbol{B}}(s), \alpha^{\boldsymbol{B}}(s), \beta^{\boldsymbol{B}}(s)$ from (19), (20), first and second order derivatives of the cost with respect to $\boldsymbol{B}'$ can be easily computed (though these expressions are omitted for brevity). The optimization problem in (21) is an instance of non-convex, non-linear optimization with linear constraints, and can be solved quickly in practice using solvers like IPOPT [27] or SNOPT [28].

Remark V.1: Paths computed using our methods require only 9 constants to describe (see Remark IV.4). Moreover, the velocity profiles here require only $N+1 \ge 5$ additional constants. We typically use $N = 10$ (which works well in practice) for a total of 19 constants to describe a trajectory. This is typically far less than what is required to describe numerically derived paths in the absence of analytic expressions. For example, a 10 m trajectory with (x, y, θ, v) samples spaced every 0.1 meters would require 20 times more space to store than the proposed method.

VI. Results

We now demonstrate the benefits of our approach. We begin by illustrating how the techniques developed here can be used to anticipate the driving styles of several archetypal users: a comfort-favoring (comfort) user, a moderate user who favors a mix of speed and comfort, and a speed-favoring (speed) user. Next, we compare the theoretical cost of paths generated using the proposed method to those from [14]. This comparison was chosen based on Theorem III.1 which implies that any path that solves (8) is a G^3 path. Finally, we measure tracking error and final cost of our proposed trajectories using Matlab's seven degree-of-freedom simulator with default road friction and input noise, and built-in Stanley controller [29]. Our methods were encoded entirely in Python 3.7 (Spyder), and IPOPT was used in Python to solve the OP in (21) to calculate velocity. The results were obtained using a desktop equipped with an AMD Ryzen 3 2200 G processor and 8 GB of RAM running Windows 10 OS.

A. Setup

Unit-Less Weighting: The features in the cost function (7) represent different physical quantities. For the cost function to represent a meaningful trade-off between these features given weights, a scaling factor is included [30]. We let $\hat{C}_m, m \in \{a, t, y, \mathcal{J}\}$ denote the cost of the trajectory that solves (8) given weight $w_m = 1$, $w_n = 0$, $\forall n \in \{a, t, y, \mathcal{J}\}$, $n \ne m$. We then scale the weights as

$$\hat{w}_m = \frac{w_m}{\hat{C}_m} \sum_{m\in\{a,\mathcal{J},y,t\}} \hat{C}_m, \quad m \in \{a, \mathcal{J}, y, t\}.$$

This scaling process[1] has the following benefits: if all features are weighted equally, then the scaled costs of each feature $\hat{w}_m \hat{C}_m$ are equal. Also, for some features m, the feature cost C_m is very sensitive to changes in the trajectory in a neighborhood of the optimal trajectory. For these features, if the scaling factor used any but the optimal value $\hat{C}_m$, it would risk of over-reducing the weight on this feature.

Parameter Bounds: For all experiments, the curvature parameter limits are given by [14]:

$$\kappa_{\max} = 0.1982\, m^{-1}, \ \sigma_{\max} = 0.1868\, m^{-2}$$
$$\rho_{\max} = 0.3905\, m^{-3}. \tag{22}$$

while the limits on velocity, acceleration, and instantaneous jerk are given, respectively, by [31]

$$v_{\max} = 100\, km/hr, \ a_{\max} = 0.9\, m/s^2, \ b_{\max} = 0.6\, m/s^3. \tag{23}$$

B. Evaluation

Qualitative Analysis: To qualitatively evaluate our methods, we analyse trajectories for three archetypal users: a *speed* user, a *comfort* user, and a *moderate*. To simplify the comparisons, we combine the scaled features IS acceleration, IS jerk, and IS yaw into a *discomfort cost*, and compare this to *time cost* for each user:

$$\text{discomfort cost} = \sum_{m\in\{a,\mathcal{J},y\}} \hat{w}_m \hat{C}_m, \ \text{time cost} = \hat{w}_m \hat{C}_t.$$

The expected favored trajectories of the three users is apparent: the speed user wishes to minimize travel time at the expense of comfort – favor trajectories featuring short paths and high velocities. The comfort user, prefers comfortable trajectories over short travel times. Finally, the intermediate driver should favor a balance of the two other users. In this experiment, the start-goal configurations for each user are:

$$p_s = [0, 0, 0, 0], \ p_f = [30, 105, 0, 0.1695],$$

with $v_s = 12\, m/s, v_g = 13\, m/s$. The weights for each user are:

$$\text{speed User } w_t = 0.9, w_y = w_{\mathcal{J}} = w_a = 0.033,$$
$$\text{Intermediate User } w_t = w_y = w_{\mathcal{J}} = w_a = 0.25$$
$$\text{comfort User } w_t = 0.01, w_y = w_{\mathcal{J}} = w_a = 0.33. \tag{24}$$

Thus, the speed user places a high weight on time, the comfort user places a high weight on discomfort, and the intermediate user places an equal weight on all features. The results are illustrated in Fig. 6. The left image illustrates the velocity profile (top), discomfort cost (mid), and time cost (bottom) for the speed (red), intermediate (blue), and comfort (green) drivers.

[1]NOTE: The scaling process described here is not included in the timing of our algorithm reported in Table III

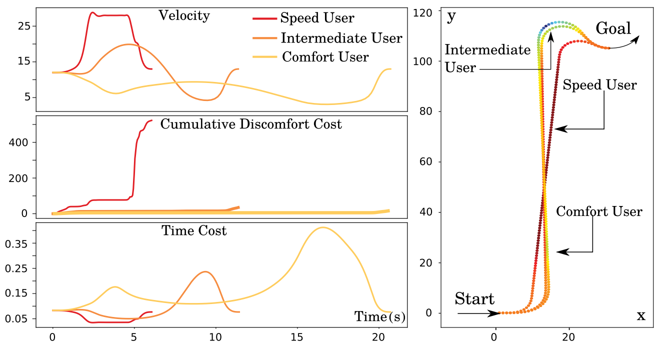

Fig. 6. Comparison of three users. Left: velocity (m/s) (top), time cost (mid) and discomfort cost (bottom) for three users. Right: trajectories.

The right image illustrates the resulting trajectories for each user with color representing velocity. The shortest trajectory is that of the speed user. Observe that her path is short and velocities high, reflecting her preference towards short travel times. The longest trajectory is that of the comfort user, and we note that her velocity decreases at moments of high curvature. Longer path lengths allow the comfort user to change velocity over a longer time period allowing her to reach lower velocities with lower acceleration and jerk. The middle trajectory is that of the intermediate user, who decreases her velocity on the turn to minimize jerk/acceleration, but then speeds up again on the straight portion of the path.

A further example of the three archetypes is given in Fig. 1 in which a lane change maneuver is preformed from $(0, 0, 0, 0)$ to $(50, 6, 0, 0)$ with an initial and final velocity of 10 m/s. Cars are drawn ever 0.75 seconds, with the dotted vertical lines representing horizontal position of the top-most car at each time step. The comfort driver has an average speed of 9.4 m/s and completes the maneuver in 5.30 seconds, while the mixed and speed drivers have average speeds and final times of (10.09 m/s, 4.91 seconds), and (10.80 m/s, 4.55 seconds), respectively. The car following the red trajectory arrives at the goal almost a full time-step ahead of the blue-trajectory car.

Path Evaluation: We now evaluate the quality of the *paths* generated using our technique. We isolate the effect of the path on the final cost by comparing two trajectory methods:

- *Ours:* Paths and velocity profiles are computed using the methods detailed in this paper.
- *Benchmark:* velocity profiles are computed using the techniques outlined here, but paths are computed from [14].

We define the $\%$ Savings as the difference in cost (given in (7)) for trajectories generated using Benchmark and Ours as a percentage of the cost of Benchmark-based trajectories.

For this experiment, 1300 start-goal pairs were randomly generated with $\kappa_s, \kappa_f \in [-\kappa_{\max}, \kappa_{\max}], v_s = v_f \in (0, v_{\max}], x_s, y_s \in [-20\, m, 20\, m]$, and weights ranging from 0 to 1. Initial and final velocities were chosen to reflect the Euclidean distance between start-goal pairs, and pairs with no feasible solution were disregarded. The results of these experiments are tabulated in Table I. Here, a feature (IS time, IS yaw, IS acceleration, IS jerk) is said to be *dominant* if the corresponding weight is greater than 0.5. If no weight is greater than 0.5, the

TABLE I
AVERAGE COST SAVINGS. BREAKDOWN BY DOMINANT FEATURE AND INITIAL/FINAL VELOCITY

		Avg. Percent Savings (%)
Dominant Feature	IS Time	20.00 (20.7)
	IS Yaw	26.24 (24.8)
	IS Acceleration	30.26 (24.2)
	IS Jerk	64.09 (26.7)
	Blended	42.06 (25.0)
Initial/Final Velocity	Low $(0, v_{\max}/3]$	39.41 (29.9)
	Med $(v_{\max}/3, 2v_{\max}/3]$	38.57 (29.0)
	High $(2v_{\max}/3, v_{\max}]$	33.50 (27.8)

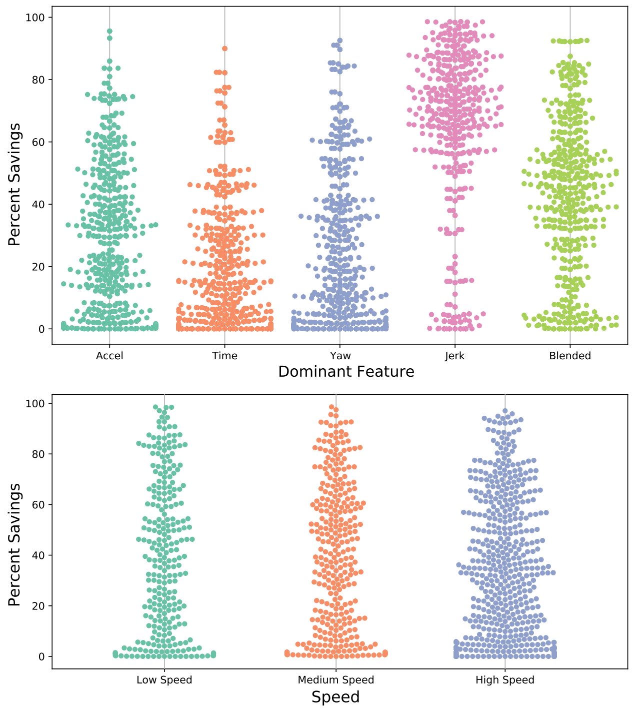

Fig. 7. Percent savings distribution by dominant feature (top), and initial/final velocity (bottom).

features are said to be *blended*. It was found that the average percent savings was 36.35% across all experiments. Moreover, if the velocity is not optimized using our method, but is assumed to be constant, we see see a 43.3% savings. The largest savings is typically found when IS jerk is the dominant feature implying that the techniques presented here have greater value for users who prefer comfort to low travel times, but can still reduce costs by 20% on average for speed users. A detailed view of the distribution of the percent savings for the experiments is given in Fig. 7. In 34% of experiments, the savings was at least 50%, while in 17% of experiments, the savings was at least 70%, and 10% of experiments had a savings of at least 80%.

Tracking Error and Simulation Cost: The above results imply that the proposed methods result in a cost reduction if tracked perfectly by a vehicle. However, it is unlikely that any reference will be perfectly tracked. While we do not propose a tracking controller in this paper (nor do we propose a local planner), we will address two important questions. First will imperfections in the vehicles controller negate the theoretical cost savings?

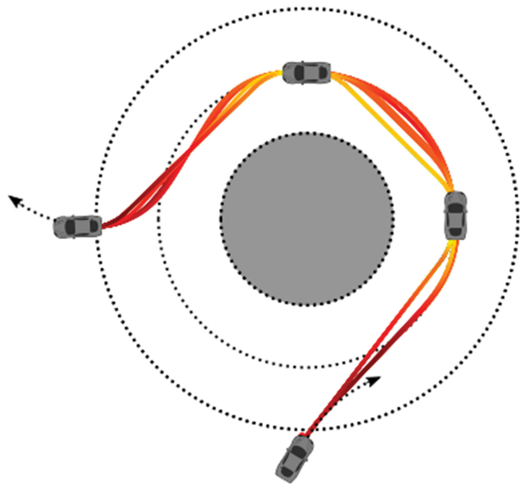

Fig. 8. Optimal trajectories for weights between way-points of a roundabout. Cars represent fixed way-points (position, curvature, velocity) while color gradient of each trajectory represents velocity.

Second, does any cost savings come at the expense of the tracking error? That is, are reference trajectories generated using our method harder to track than the Benchmark? We use a basic out-of-the-box feedback controller that has not been optimized, and we do not re-plan reference trajectories. This is to illustrate that even with a sub-optimal controller, there is still substantial cost savings, and typically a *reduction* in the tracking error. We again consider the methods "Ours" and "Benchmark" defined above. The experiments in this section were carried out using MATLAB's seven degree of freedom driving simulator in its autonomous driving toolbox. All car dimensions, force coefficients, and noise parameters were left at their default values. The reference trajectories were tracked using MATLAB's Stanley controller with default gains. Here, the maximum error is the maximum lateral error, and the average error is the integral of the lateral error normalized to arc-length of the path. We consider three maneuvers between start goal pairs chosen to coincide with possible way-points of a typical roundabout [32, Chapter 6] (Fig. 8) with lane radius (middle circle) of 45 ft. The initial way-point for this experiment is on a connecting road 65 ft away from the center of the roundabout. Velocities at the way-points are 17 mph, 14 mph, and 17 mph, respectively. Trajectories between these way-points were planned for 50 weights: 5 values of the time weight w_t, (0, 0.25, 0.5, 0.75, 1) and 10 randomly selected values of the other weights for each value of w_t (ensuring that the sum of the weights equaled 1). Trajectories lateral errors for one example weight $(w_{\mathcal{J}} = 1, w_t = w_a = w_y = 0)$ are shown in Fig. 9. Left images show the trajectories obtained using the two methods. Blue, red, and green paths represent theoretical, tracked, and driven paths, respectively. Short line segments connect driven configurations with the reference configurations being tracked at each time step. The high peaks in error (right images) experienced by cars following the Benchmark reference, are spread more evenly along the driven trajectory for cars tracking the proposed reference owing to the gradual changes in curvature. Maximum error, average error, and cost savings of each experiment are tabulated in Fig. 10, and Table II. Savings in every metric is highest for weights that favor comfort over

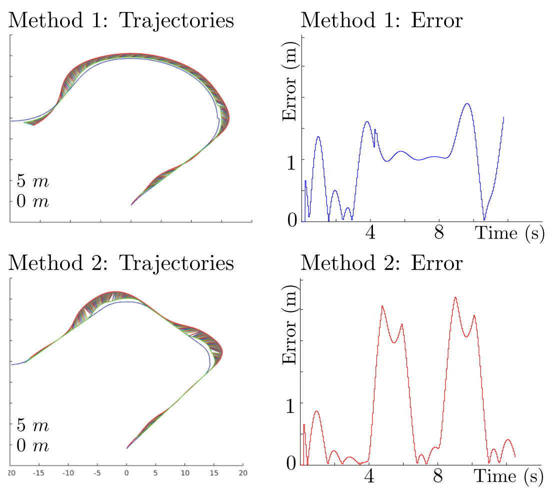

Fig. 9. (Left) Simulated trajectories for example roundabout maneuver. Blue, green, and red lines are reference, tracked, and driven paths. Connecting lines show tracking error. (Right) Lateral Error over time. Methods 1 (Ours) and 2 (Benchmark).

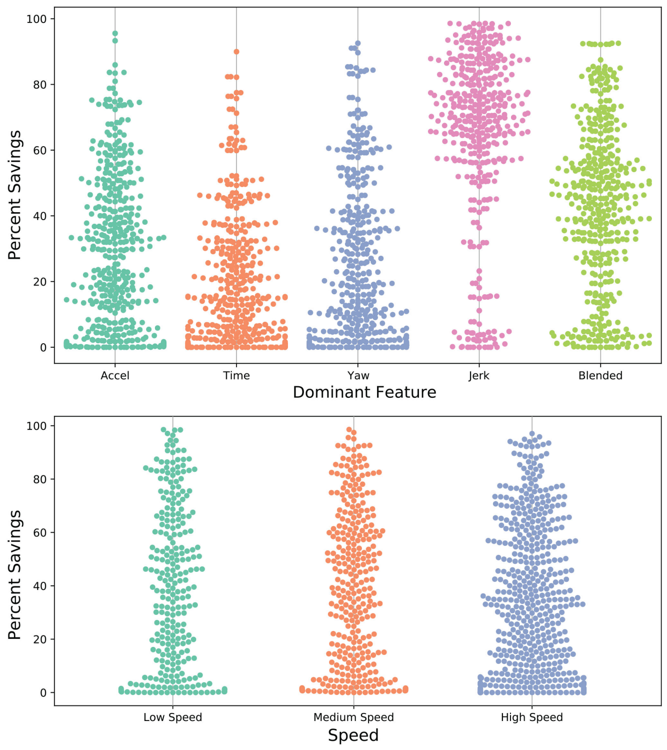

Fig. 10. Average maximum errors (top) and average average errors (bottom) for Methods 1 and 2 categorized by time weight.

TABLE II
AVERAGE COST SAVINGS, MAXIMUM LATERAL ERROR AND AVERAGE LATERAL ERROR FOR TWO METHODS (O = OURS, B = BENCHMARK)

Time weight (w_t)	Cost Savings (%)	Max Err (m)		Avg Err (m)	
		O	B	O	B
0	35.8	2.05	2.82	0.13	0.14
0.25	18.7	2.11	2.83	0.13	0.14
0.5	13.7	2.33	2.85	0.13	0.14
0.75	8.9	2.40	2.85	0.14	0.14
1	1.0	2.75	2.77	0.15	0.15

TABLE III
COMPARISON OF RUN-TIMES. OUR METHODS (OURS) COMPARED AGAINST STATE-OF-THE-ART (SOTA) FOR EACH SUB-ROUTINE OF THE PROCEDURE

Sub-routine	SOTA source	Run-times (s) SOTA (C)	Run-times (s) Ours (Python)
Calculate a path (Step 1)	[14]	6.8×10^{-5}	4.3×10^{-3}
Optimize velocity (Step 2)	[19]	2.2×10^{-4}	1.2×10^{-2}
Total (recursive Step 1 & 2)	[14] & [19]	1.2×10^{-3}	7.3×10^{-2}

time. Further, the maximum lateral tracking error can be reduced by approximately 0.8 m on average for certain weights. The roundabout radius and velocities were scaled upwards (again following the guidelines of [32]). However, the results were very similar to those reported above.

Run-time: The focus of this work was on improving the quality of computed trajectories given user weights, and we did not focus on optimizing for run time. We believe that with proper software engineering, one could substantially reduce the run-time of our procedure. Run-times for each step in our procedure are compared with those of the state of the art (SOTA) in Table III. Reference [19] is the SOTA for velocity profile planning given a path due to its use of third order trapezoidal methods. Trapezoidal methods are a widely used method for velocity profile planning [33], [34]. These last use higher-order velocity models than those employed by [19], improving the quality at the expense of computation time. We therefore use computation times reported in [19] as a lower bound. The SOTA is on the order of 100 times faster than the methods proposed here. However, our algorithm is encoded entirely in Python which tends to run 10–100 times slower than C for iterative procedures like ours.

VII. CONCLUDING REMARKS

We consider the problem of computing trajectories between start and goal states that optimize a trade off between comfort at time. We offer a simplified approximation of optimization problem (8) that replaces the two continuous functions with n constants. Methods to compute paths given one of these constants, and a velocity profile given this path and the $n-1$ remaining constants are also provided. The resulting technique is verified with numerical examples.

APPENDIX

Proof of Lemma IV.1: Consider a set of curves $\mathcal{G}$. Let $G(\Delta):\mathbb{R}_{\geq s_3}\to\mathcal{G}$ be the function that sends values of $\Delta\geq s_3$ to the corresponding G^3 curve in $\mathcal{G}$. Let $x(s,\Delta)$ $y(s,\Delta)$, $\theta(s,\Delta)$, $\kappa(s,\Delta)$ denote the path state profiles of the curve $G(\Delta)$ at arc-length s, and let $x_f(\Delta), y_f(\Delta), \theta_f(\Delta), \kappa_f(\Delta)$ denote the final path states of $G(\Delta)$. From Equations (1), (12), we make the following observation:

$$\theta(s,\Delta) = \begin{cases}\theta(s,s_3), & \text{if } s\leq s_3\\ \theta(s_3,s_3)+\kappa_{\text{top}}(s-s_3), & \text{if } s_3<s\leq\Delta\\ \kappa_{\text{top}}(\Delta-s_3)+\theta(s-(\Delta-s_3),s_3), & \text{if } \Delta<s\leq s_6.\end{cases} \quad (25)$$

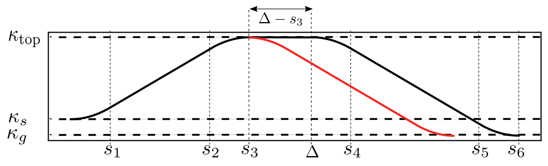

Fig. 11. Curvature profile of two curves in the set $\mathcal{G}$. (Red): curvature of the curve $G(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta = s_3)$. (Black): curvature of the curve $G(\boldsymbol{p}_s, \kappa_{\text{top}}, \kappa_f, \Delta)$ for $\Delta > s_3$.

As implied by Fig. 11, if s_6 is the final arc-length of the curve $G(\Delta)$, then $s_6 - (\Delta - s_3)$ is equal to the final arc-length of the curve $G(s_3)$. Therefore, we can conclude from (25) that $\theta_f(\Delta) = \kappa_{\text{top}}(\Delta - s_3) + \theta_f(s_3)$. Setting this last to θ_g, and solving for Δ completes the proof. $\square$

Proof Of Theorem IV.2: This proof uses the notation of the Proof of Lemma IV.1. Given a set of curves $\mathcal{G}$, it can be shown from equations (25), and (1), that

$$\begin{aligned}
\theta_f(\Delta) &= \theta_f(s_3) + \kappa_{\text{top}}(\Delta - s_3),\\
x_f(\Delta) &= x(s_3, s_3) - \frac{\sin(\theta(s_3)) - \sin(\kappa_{\text{top}}(\Delta - s_3) + \theta(s_3))}{\kappa_{\text{top}}}\\
&\quad + \cos(\kappa_{\text{top}}(\Delta - s_3))(x_f(s_3) - x(s_3, s_3))\\
&\quad - \sin(\kappa_{\text{top}}(\Delta - s_3))(y_f(s_3) - y(s_3, s_3)),\\
y_f(\Delta) &= y(s_3, s_3) - \frac{-\cos(\theta(s_3)) + \cos(\kappa_{\text{top}}(\Delta - s_3) + \theta(s_3))}{\kappa_{\text{top}}}\\
&\quad + \sin(\kappa_{\text{top}}(\Delta - s_3))(x_f(s_3) - x(s_3, s_3))\\
&\quad + \cos(\kappa_{\text{top}}(\Delta - s_3))(y_f(s_3) - y(s_3, s_3)).
\end{aligned} \tag{26}$$

Observe now that the shortest distance from the point $\boldsymbol{x}_c = [x_c, y_c]^T$ given in (14) to the line passing through $(x_f(\Delta), y_f(\Delta))$ whose slope is $\tan\theta_f(\Delta)$, is given by

$$d(\Delta) = \frac{x_c \tan\theta_f(\Delta) - y_c + y_f(\Delta) - x_f(\Delta)\tan\theta_f(\Delta)}{\sqrt{\tan\theta_f(\Delta)^2 + 1}}.$$

Replacing (26) in $d(\Delta)$, yields a complete cancellation of the term Δ. The remaining parameters $\kappa_{\text{top}}, s_3, \boldsymbol{x}_c, etc.$, are identical for all members of $\mathcal{G}$, thus we can conclude that $d(\Delta)$ is the same for all members of $\mathcal{G}$ as desired. $\square$

Acknowledgment

The authors would like to thank the anonymous reviewers for greatly helping us to improve the performance of our velocity profile computation technique.

References

[1] C. Urmson *et al.*, “Autonomous driving in urban environments: Boss and the urban challenge,” *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[3] Y. Zhang *et al.*, “Hybrid trajectory planning for autonomous driving in highly constrained environments,” *IEEE Access*, vol. 6, pp. 32 800–32 819, 2018.

[4] J. Levinson *et al.*, “Towards fully autonomous driving: Systems and algorithms,” in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2011, pp. 163–168.

[5] J. Ziegler, P. Bender, T. Dang, and C. Stiller, “Trajectory planning for bertha-A local, continuous method,” in *Proc. IEEE Intell. Veh. Symp.*, 2014, pp. 450–457.

[6] K. J. Kyriakopoulos and G. N. Saridis, “Minimum jerk for trajectory planning and control,” *Robotica*, vol. 12, no. 2, pp. 109–113, 1994.

[7] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, “Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions,” *Transp. Res. Part C, Emerg. Technol.*, vol. 60, pp. 416–442, 2015.

[8] L. T. Biegler, “Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation,” *Comput. Chem. Eng.*, vol. 8, no. 3/4, pp. 243–247, 1984.

[9] B. Barsky and T. DeRose, “Geometric continuity of parametric curves: Three equivalent characterizations,” *IEEE Comput. Graph. Appl.*, vol. 9, no. 6, pp. 60–69, Nov. 1989.

[10] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, “A real-time motion planner with trajectory optimization for autonomous vehicles,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 2061–2067.

[11] T. Fraichard and A. Scheuer, “From reeds and Shepp’s to continuous-curvature paths,” *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 1025–1035, Dec 2004.

[12] J. A. R. Silva and V. Grassi, “Clothoid-based global path planning for autonomous vehicles in urban scenarios,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4312–4318.

[13] R. Oliveira, P. F. Lima, M. Cirillo, J. Møartensson, and B. Wahlberg, “Trajectory generation using sharpness continuous dubins-like paths with applications in control of heavy-duty vehicles,” in *Proc. Eur. Control Conf.*, 2018, pp. 935–940.

[14] H. Banzhaf, N. Berinpanathan, D. Nienhüser, and J. M. Zöllner, “From $G2$ to $G3$ continuity: Continuous curvature rate steering functions for sampling-based nonholonomic motion planning,” in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2018, pp. 326–333.

[15] A. Botros and S. L. Smith, “Computing a minimal set of t-spanning motion primitives for lattice planners,” in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2019, pp. 2328–2335.

[16] J. Ziegler *et al.*, “Making bertha drive-An autonomous journey on a historic route,” *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 2, pp. 8–20, Summer 2014.

[17] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, “Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications,” *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 2, pp. 740–753, Apr. 2016.

[18] J. W. Ro, P. S. Roop, and A. Malik, “A new safety distance calculation for rear-end collision avoidance,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1742–1747, Mar. 2021.

[19] R. Haschke, E. Weitnauer, and H. Ritter, “On-line planning of time-optimal, jerk-limited trajectories,” in *Proc. IEEE /RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 3248–3253.

[20] H. Guo, C. Shen, H. Zhang, H. Chen, and R. Jia, “Simultaneous trajectory planning and tracking using an MPC method for cyber-physical systems: A case study of obstacle avoidance for an intelligent vehicle,” *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4273–4283, Sep. 2018.

[21] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a frenet frame,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010, pp. 987–993.

[22] K. Bergman, O. Ljungqvist, T. Glad, and D. Axehill, “An optimization-based receding horizon trajectory planning algorithm,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 550–15 557, 2020.

[23] F. Gao *et al.*, “Optimal trajectory generation for quadrotor teach-and-repeat,” *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1493–1500, Apr. 2019.

[24] S. H. Schot, “Jerk: The time rate of change of acceleration,” *Amer. J. Phys.*, vol. 46, no. 11, pp. 1090–1094, 1978.

[25] M. I. Kamien and N. L. Schwartz, *Dynamic Optimization: The Calculus of Variations and Optimal Control in Economics and Management*, 2nd ed. New York, NY, USA: Dover Publications, 2013.

[26] C. Cartis, J. Fiala, B. Marteau, and L. Roberts, “Improving the flexibility and robustness of model-based derivative-free optimization solvers,” *ACM Trans. Math. Softw. (TOMS)*, vol. 45, no. 3, pp. 1–41, 2019.

[27] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[28] P. E. Gill, W. Murray, and M. A. Saunders, “SNOPT: An SQP algorithm for large-scale constrained optimization,” *SIAM Rev.*, vol. 47, no. 1, pp. 99–131, 2005.

[29] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *Proc. Amer. Control Conf.*, 2007, pp. 2296–2301.

[30] S. Gulati, C. Jhurani, and B. Kuipers, "A nonlinear constrained optimization framework for comfortable and customizable motion planning of nonholonomic mobile robots-Part I," 2013, *arXiv:1305.5024*.

[31] I. Bae, J. Moon, and J. Seo, "Toward a comfortable driving experience for a self-driving shuttle bus," *Electronics*, vol. 8, no. 9, 2019, Art. no. 943.

[32] B. Robinson *et al.*, "Roundabouts: An informational guide," Turner-Fairbank Highway Res. Center, Fed. Highway Admins., U.S. Dept. Transp., McLean, VA, USA. [Online]. Available: https://www.fhwa.dot.gov/publications/research/safety/00068/

[33] Y. Fang, J. Hu, W. Liu, Q. Shao, J. Qi, and Y. Peng, "Smooth and time-optimal S-curve trajectory planning for automated robots and machines," *Mechanism Mach. Theory*, vol. 137, pp. 127–153, 2019.

[34] M. Raineri and C. Guarino LoBianco, "Jerk limited planner for real-time applications requiring variable velocity bounds," in *Proc. IEEE 15th Int. Conf. Automat. Sci. Eng.*, 2019, pp. 1611–1617.

Alexander Botros is currently working toward the Ph.D. degree with the Autonomous Systems Lab, University of Waterloo, Waterloo, ON, Canada. He completed his undergraduate and graduate engineering work with Concordia University, Montreal, QC, Canada. His research interests include local planning and trajectory generation for autonomous vehicles, and computation of minimal t-spanning sets of edges for state lattices with the goal of using these sets as motion primitives for autonomous vehicles.

Stephen L. Smith (Senior Member, IEEE) received the B.Sc. degree from Queen's University, Kingston, ON, Canada, in 2003, the M.A.Sc. degree from the University of Toronto, Toronto, ON, Canada, in 2005, and the Ph.D. degree from the University of California, Santa Barbara, Santa Barbara, CA, USA, in 2009. From 2009 to 2011, he was a Postdoctoral Associate with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. He is currently an Associate Professor of electrical and computer engineering with the University of Waterloo and a Canada Research Chair of autonomous systems. His main research interests include control and optimization for autonomous systems, with emphasis on robotic motion planning and coordination.