2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Autonomous Vehicle Motion Planning via Recurrent Spline Optimization

Wenda Xu[1], Qian Wang[2], and John M. Dolan[3]

***Abstract*— Trajectory planning in dynamic environments can be decomposed into two sub-problems: 1) planning a path to avoid static obstacles, 2) then planning a speed profile to avoid dynamic obstacles. This is also called path-speed decomposition. In this work, we present a novel approach to solve the first sub-problem, motion planning with static obstacles. From an optimization perspective, motion planning for autonomous vehicles can be viewed as non-convex constrained nonlinear optimization, which requires a good enough initial guess to start and is often sensitive to algorithm parameters. We formulate motion planning as convex spline optimization. The convexity of the formulated problem makes it able to be solved fast and reliably, while guaranteeing a global optimum. We then reorganize the constrained spline optimization into a recurrent formulation, which further reduces the computational time to be linear in the optimization horizon size. The proposed method can be applied to both trajectory generation and motion planning problems. Its effectiveness is demonstrated in challenging scenarios such as tight lane changes and sharp turns.**

I. Introduction

Motion planning techniques can be classified into two groups according to the continuity of the state space: search-based planning [1], [2] and optimization-based planning [3], [4], [5]. In search-based planning, the state space is discretized into a grid or lattice. In optimization-based planning, the state space is continuous. Although search-based planning guarantees a global optimum with respect to its resolution, it may lose accuracy due to discretization. Optimization-based planning often falls into a local optimum because the planning problem is generally not convex and the global optimum is hard to find for such problems through optimization techniques.

In this work, we present a novel convex optimization-based approach for the autonomous vehicle motion planning problem. There are three types of constraints making the problem non-convex: obstacles, the nonlinear system model, and system limits (e.g. curvature constraint). First, based on differential flatness theory, we show that the nonlinear kinematic bicycle model can be automatically satisfied with a spline representation, which makes the nonlinear system constraints linear. Second, we build corridors based on vertical cell decomposition, which converts obstacles into convex constraints. Third, we handle the curvature constraint through iteratively convex optimization.

[1]Wenda Xu is with ECE, Carnegie Mellon University, Pittsburgh, PA 15213 wendax@andrew.cmu.edu

[2]Qian Wang was with the Applied Dynamics & Control Group, International Center for Automotive Research (CU-ICAR), Clemson University, Greenville, SC 29607 qwang8@g.clemson.edu

[3]John M. Dolan is with Faculty of the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 jdolan@andrew.cmu.edu

Our contributions are summarized as follows: 1) We convert motion planning into a convex quadratic program, which enables a reliably global optimal trajectory. 2) By reformulating the constrained spline optimization into optimal control form, we further reduce the computational time to be linear in the optimization horizon size. 3) Our method doesn't require an initial feasible solution, allowing an infeasible start. 4) Our approach allows large step size thanks to the linear system model based on differential flatness theory.

The rest of the paper is organized as follows: Related work is presented in Section II, where trajectory generation and optimization-based planning methods are discussed. The kinematic bicycle model and differential flatness are described in Sections III-A and III-B respectively. Constrained spline optimization is introduced in Section III-D. Heading and curvature constraints are discussed in Section III-E. Section III-F presents the reformulation of the constrained spline optimization into optimal control. Experiments for trajectory generation and planning are described in Section IV.

II. Related Work

A. *Trajectory Generation*

There are two types of trajectory generation methods: forward and inverse kinematics. In forward kinematics, the initial state of the robot and control input are given, and the final state can be obtained through integrating the control input. Typical numerical integration methods include the Euler, Simpson, and Runge-Kutta methods. In inverse kinematics, the initial and final state of the robot are given, and the control input is to be determined. This usually requires solving a system of algebraic equations if the trajectory can be represented by polynomials [6] or a trajectory optimization problem for general nonlinear systems [7], [8].

Simultaneously handling path and curvature constraints efficiently is a challenging problem. [8] uses curvature polynomials to ensure smooth curvature. However, path constraints cannot be directly imposed, as they need to be integrated from curvature. [6] generates the lateral and longitudinal trajectory using quintic polynomials versus time, and it uses a Frenet frame referenced to the road center line to combine lateral and longitudinal motion. This makes the trajectory longitudinally coincide with the road shape. However, curvature has a complicated expression in the Frenet frame, making it inefficient to impose curvature constraints. A spline optimization method [9] that is based on Frenet frame has the same limitation for curvature. We solve this problem by defining the spline in global coordinates, where both the path and curvature constraints can be reformulated to be linear functions of the states, as discussed in Sec III-D and III-E.

978-1-7281-9077-8/21/$31.00 ©2021 IEEE

B. Optimal Control and Trajectory Optimization

Trajectory optimization is a technique to provide an open-loop solution to an optimal control problem (OCP). A special case of the general nonlinear optimal control problem is the linear quadratic regulator (LQR), where the system dynamics can be described as linear differential equations and the cost can be described by a quadratic function. For a linear time-invariant (LTI) system, an analytic solution for LQR exists by directly solving the Riccati equation [10]. For a linear time-varying (LTV) system, the optimal policy can be computed recursively based on the Riccati equation. For nonlinear systems, iterative linear quadratic regulator (iLQR) [11], [12] uses an iterative version of LQR to find the optimal trajectory, where the first-order derivatives of the dynamics are used. [13] proposes LQR-smoothing, which applies both a backward and forward LQR pass. In the original LQR setting, no constraints can be added for state and control variables. Constraints in [13] are handled by converting into cost with a barrier function. Unlike the LQR-based methods, our method allows directly imposing hard constraints on state and control variables.

For the more general nonlinear optimal control problem with state and control constraints, direct trajectory optimization methods such as shooting [14] and collocation [15] are commonly used. Direct methods first transform trajectory optimization into a constrained nonlinear program (NLP). The NLP can then be solved by numerical optimization methods such as sequential quadratic programming (SQP) [16] or the interior point method [17], [18], [19]. The process of transforming an optimal control problem (OCP) into a nonlinear program (NLP) is called transcription. Typical transcription approaches include single shooting, multiple shooting, and collocation [20]. In single shooting, only the controls are discretized and used as decision variables. In multiple shooting [16], the trajectory is divided into multiple segments. Equality constraints are imposed at the joint points of the segments to ensure continuity. In collocation, both the controls and the states are represented as piecewise polynomials. Our quadratic spline formulation (10) can be viewed as a collocation method, and recurrent spline formulation (22) can be viewed as a multiple shooting method.

III. Optimization with Kinematic Bicycle Model

A. Kinematic Bicycle Model

The kinematic model is subject to nonholonomic constraints due to the rolling without slipping condition between the wheels and the ground [21]. The front wheel is steerable and the rear wheel orientation is fixed. As shown in Fig. 1, (x, y) are the global coordinates of the rear wheel, (x_f, y_f) are the global coordinates of the front wheel, θ is the orientation of the car body in the global frame, δ is the steering angle in the body frame, ω is the angular velocity of the steering wheel, L is the distance between the wheels, and v is the longitudinal velocity along the body at the rear wheel. The state of the kinematic model is defined as $\mathbf{x} = (x, y, \theta, \delta)$, and the control input is $\mathbf{u} = (v, \omega)$. The system equation of a kinematic car model $\dot{\mathbf{x}} = f_k(\mathbf{x}, \mathbf{u})$ is

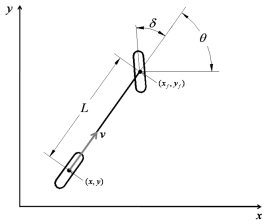

Fig. 1. Kinematic bicycle model

$$\dot{x} = v\cos\theta \tag{1a}$$
$$\dot{y} = v\sin\theta \tag{1b}$$
$$\dot{\theta} = v\tan\delta/L \tag{1c}$$
$$\dot{\delta} = \omega \tag{1d}$$

B. Differential Flatness

1) Differential Flatness Theory: A system is differentially flat if there exists an output $y \in \mathbb{R}^m$ (as a function of the state, input, and input derivatives), called a flat output, such that the state $x \in \mathbb{R}^n$ and the control input $u \in \mathbb{R}^m$ of the system can be expressed as algebraic functions of the flat output y and its time derivatives up to a certain order [22], [23]. Roughly speaking, a system is flat if there exists a set of outputs (equal in number to the number of control inputs) such that all states and control inputs can be determined from these outputs without integration. Mathematically, the system is flat if we can find output y of the form

$$y = \gamma(x, u, \dot{u}, \ldots, u^{(r)}) \tag{2}$$

such that

$$\begin{aligned} x &= \alpha(y, \dot{y}, \ldots, y^{(q)}) \\ u &= \varphi(y, \dot{y}, \ldots, y^{(q)}) \end{aligned} \tag{3}$$

As a consequence, once a flat output trajectory y is assigned, the associated trajectories of the state x and control inputs u are uniquely determined. The resulting trajectory of x will automatically satisfy the nonholonomic constraints.

2) Differential Flatness for the Kinematic Bicycle Model: For the kinematic bicycle model $\dot{\mathbf{x}} = f_k(\mathbf{x}, \mathbf{u})$ defined in (1), the state is $\mathbf{x} = (x, y, \theta, \delta)$, and the control input is $\mathbf{u} = (v, \omega)$. We will show that the global coordinates of the rear wheel (x, y), which have the same dimension as the control input $\mathbf{u}$, are a flat output of the kinematic bicycle system. Assuming the car only moves forward, the velocity and the orientation of the car can be obtained from (1).

$$v = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{4}$$
$$\theta = \arctan\left(\dot{y}/\dot{x}\right) \tag{5}$$

Differentiating (5) yields

$$\dot{\theta} = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2} \tag{6}$$

Plugging (6) into (1c) gives

$$\delta = \arctan\frac{\left(\ddot{y}\dot{x} - \ddot{x}\dot{y}\right)L}{\left(\dot{x}^2 + \dot{y}^2\right)^{\frac{3}{2}}} \tag{7}$$

Differentiating (7) yields

$$\omega = \frac{((\dddot{y}\dot{x} \quad \ddot{x}\ddot{y})(\dot{x}^2+\dot{y}^2) \quad 3(\ddot{y}\dot{x} \quad \dot{x}\ddot{y})(\dot{x}\ddot{x}+\dot{y}\ddot{y}))\sqrt{\dot{x}^2+\dot{y}^2}L}{(\dot{x}^2+\dot{y}^2)^3+L^2(\ddot{y}\dot{x} \quad \dot{x}\ddot{y})^2} \tag{8}$$

So far we have shown that, given output trajectory (x, y) and their derivatives up to the third order, we can generate the other states (θ, δ) in (5) (7) and control inputs (v, ω) in (4) (8). Therefore, the kinematic bicycle model is differentially flat. As a consequence, if the trajectory (x, y) is three times differentiable, it will automatically satisfy the nonholonomic kinematic constraints.

C. Handling Obstacles with Corridor

For on-road driving scenarios, the number of traffic participants that may interact with the autonomous vehicle is often limited. Therefore, instead of applying dynamic programming to a lattice grid [24], we use vertical cell decomposition [25] to partition the free-space into a set of cells. Each cell is either a trapezoid or a triangle. The corridor is built by a list of connected convex cells. The algorithm has computational efficiency of $O(n \log n)$, where n is the number of vertices in the environment.

D. Constrained Spline Optimization

We have shown in Section III-B.2 that the kinematic bicycle model is differentially flat. Therefore, we can use a polynomial to represent the path. Based on (8), the polynomial needs to be at least three times differentiable. We choose a parametric form $(x, y) = (f(s), g(s))$ to represent a curve of any shape, where s is a parameter and is chosen as the arc length of the reference path, and $f(s)$ and $g(s)$ are cubic splines defined by piecewise-cubic polynomials. It is notable that s is not the arc length of the final trajectory because the latter is unknown until the optimization has converged.

The reference path can be chosen as the center line of the lane or the straight line connecting start and goal state depending on different applications. It is notable that, unlike many other methods, the reference path does not need to be collision-free with obstacles in our approach. The points from the reference path are denoted as $\{(s_i, \bar{x}_i, \bar{y}_i, \bar{\theta}_i),\ i = 0, 1, \ldots, N\}$. The i^{th} piece of the spline is defined below, where $s_i \leq s < s_{i+1}$.

$$\begin{aligned} f_i(s) &= a_{i0} + a_{i1}(s - s_i) + a_{i2}(s - s_i)^2 + a_{i3}(s - s_i)^3 \\ g_i(s) &= b_{i0} + b_{i1}(s - s_i) + b_{i2}(s - s_i)^2 + b_{i3}(s - s_i)^3 \end{aligned} \tag{9}$$

We formulate the motion planning problem as a constrained spline optimization. Different from the smoothing spline proposed in [26], where the constraints can only be added to the square sum offset between the optimized and original path, our formulation allows imposing the constraints on any point. The resulting constrained quadratic program is formulated below

$$\min_{a_{ij},\, b_{ij}} \quad \sum_{i=0}^{N} w_1(\dddot{f}_i(s_i)^2 + \ddot{g}_i(s_i)^2) + w_2(\ddot{f}_i(s_i)^2 + \dddot{g}_i(s_i)^2) \tag{10a}$$

s.t.

$$f_0(s_0) = x_{init}, \quad g_0(s_0) = y_{init}, \tag{10b}$$

$$f_N(s_N) = x_{goal}, \quad g_N(s_N) = y_{goal}, \tag{10c}$$

$$f_i(s_i) = f_{i+1}(s_i), \quad g_i(s_i) = g_{i+1}(s_i), \tag{10d}$$

$$\dot{f}_i(s_i) = \dot{f}_{i+1}(s_i), \quad \dot{g}_i(s_i) = \dot{g}_{i+1}(s_i), \tag{10e}$$

$$\ddot{f}_i(s_i) = \ddot{f}_{i+1}(s_i), \quad \ddot{g}_i(s_i) = \ddot{g}_{i+1}(s_i), \tag{10f}$$

$$-\Delta_l \leq -(f_i(s_i) - \bar{x}_i)\sin\bar{\theta}_i + (g_i(s_i) - \bar{y}_i)\cos\bar{\theta}_i \leq \Delta_r, \tag{10g}$$

$$-\Delta_b \leq (f_i(s_i) - \bar{x}_i)\cos\bar{\theta}_i + (g_i(s_i) - \bar{y}_i)\sin\bar{\theta}_i \leq \Delta_f \tag{10h}$$

The optimization objective (10a) is the square sum of the second- and third-order derivatives of position (x, y), which stands for maximizing smoothness and minimizing energy. w_1 and w_2 are weighting parameters. Constraints (10b) and (10c) are the boundary conditions for start and goal positions. Sometimes, the goal is defined as a region instead of a fixed position; (10c) can be modified to affine inequality constraints in this case. (10d)-(10f) ensure x and y and their first- and second-order derivatives are continuous at the joints. Based on the differential flatness theory discussed in III-B.2, these continuity constraints ensure the kinematic constraints are satisfied. (10g) defines two affine inequality constraints at each point (one for the right and one for the left border of the corridor). (10h) defines another two affine inequality constraints at each point to limit the longitudinal movements. The resulting region is a convex feasible set.

The optimization problem defined in (10) contains a convex quadratic objective and linear equality and inequality constraints. Thus, it is a convex quadratic program, and the global optimal solution can be obtained by any QP solver (e.g. OSQP [27], CVXPY [28]), or the primal-dual interior point method in polynomial time [29], [30]. It has computational complexity of $O(N^3(n_p+n_c)^3)$ if its sparsity is not exploited, where N is the number of steps, n_p is the number of coefficients to be optimized at each step ($n_p = 8$ in (10)), and n_c is the number of constraints at each step.

E. Heading and Curvature Constraints

For planning, it is critical to ensure the heading and curvature be continuous at the start point, and curvature be within the maximum range along the path. Given heading θ_0 at the starting point, the constraint for heading can be expressed as a linear function of $\dot{f}$ and $\dot{g}$ based on (5).

$$-\dot{f}_0(s_0)\sin\theta_{init} + \dot{g}_0(s_0)\cos\theta_{init} = 0 \tag{11}$$

We can express curvature as $\kappa = \lambda\ddot{x} + \mu\ddot{y}$, where

$$\lambda = -\frac{\dot{y}}{(\dot{x}^2+\dot{y}^2)^{\frac{3}{2}}}, \quad \mu = \frac{\dot{x}}{(\dot{x}^2+\dot{y}^2)^{\frac{3}{2}}} \tag{12}$$

We observe that curvature κ is a linear function of the second-order derivatives $\ddot{x}$ and $\ddot{y}$ given the first order derivatives $\dot{x}$ and $\dot{y}$. Therefore, we can design an iterative scheme to approximate curvature. In each iteration, λ and μ are determined by the value of $\dot{x}$ and $\dot{y}$ that are obtained

from the last iteration. Curvature converges to the true value as the path converges to the final optimized path.

The only special case is that $\dot{x}$ and $\dot{y}$ are not available in the first iteration. In this case, we will show how to approximate curvature with the reference path. Denote the arc length of the final optimized path as σ. By the Pythagorean theorem, we get $d\sigma = \sqrt{dx^2 + dy^2}$. Taking derivatives on both sides with respect to s, we get

$$\frac{d\sigma}{ds} = \sqrt{\left(\frac{dx}{ds}\right)^2 + \left(\frac{dy}{ds}\right)^2} = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{13}$$

If σ is close to the arc length of the reference path s, we obtain $\frac{d\sigma}{ds} \approx 1$. Therefore,

$$\sqrt{\dot{x}^2 + \dot{y}^2} \approx 1 \tag{14}$$

If the heading from the reference path $\bar{\theta}$ is also close to the heading of the final path θ, we get $\bar{\theta} \approx \theta$. Substituting this and (14) into λ, we get

$$\lambda = \frac{-\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} = \frac{-1}{\dot{x}^2 + \dot{y}^2} \frac{\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}} \approx -\sin\bar{\theta} \tag{15}$$

Similarly, we get $\mu \approx \cos\bar{\theta}$. Substituting (15) into (12) yields that curvature can be approximated by the heading of the reference path in the first iteration of the optimization.

$$\kappa \approx -\ddot{x}\sin\bar{\theta} + \ddot{y}\cos\bar{\theta} \tag{16}$$

With (12) and (16), the curvature at the start point can be expressed as a linear equality constraint, and the curvature limit along the path can be expressed as linear inequality constraints. We have shown in (11) that the heading constraint is also a linear equality constraint. Therefore, with the additional heading and curvature constraints introduced in this section, the optimization problem (10) is still a constrained convex quadratic program.

F. From Constrained Spline Optimization to Optimal Control

In this section, we reformulate the constrained spline optimization in (10) to be an optimal control problem. The special structure in the optimal control problem can then be exploited, yielding a more efficient solver.

1) Optimal Control Formulation: Denoting $h = s_{i+1} - s_i$, based on (9) and its derivatives, we get

$$\begin{aligned} x_{i+1} &= f_i(s_{i+1}) = a_{i0} + a_{i1}h + a_{i2}h^2 + a_{i3}h^3 \\ \dot{x}_{i+1} &= \dot{f}_i(s_{i+1}) = a_{i1} + 2a_{i2}h + 3a_{i3}h^2 \\ \ddot{x}_{i+1} &= \ddot{f}_i(s_{i+1}) = 2a_{i2} + 6a_{i3}h \\ \dddot{x}_{i+1} &= \dddot{f}_i(s_{i+1}) = 6a_{i3} \end{aligned} \tag{17}$$

Reorganizing (17) into matrix form yields

$$\begin{bmatrix} x_{i+1} \\ \dot{x}_{i+1} \\ \ddot{x}_{i+1} \\ \dddot{x}_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & h & h^2 & h^3 \\ 0 & 1 & 2h & 3h^2 \\ 0 & 0 & 2 & 6h \\ 0 & 0 & 0 & 6 \end{bmatrix} \begin{bmatrix} a_{i0} \\ a_{i1} \\ a_{i2} \\ a_{i3} \end{bmatrix} \tag{18}$$

Based on (9), we also have

$$\begin{aligned} x_i &= f_i(s_i) = a_{i0}, & \dot{x}_i &= \dot{f}_i(s_i) = a_{i1} \\ \ddot{x}_i &= \ddot{f}_i(s_i) = 2a_{i2}, & \dddot{x}_i &= \dddot{f}_i(s_i) = 6a_{i3} \end{aligned} \tag{19}$$

Substituting (19) into (18) and denoting $\tilde{x}_i = (x_i, \dot{x}_i, \ddot{x}_i)$, $\tilde{u}_i^x = \dddot{x}_i$ yields

$$\begin{aligned} \tilde{x}_{i+1} &= A\tilde{x}_i + B\tilde{u}_i^x \\ A = \begin{bmatrix} 1 & h & \frac{1}{2}h^2 \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix} & \quad B = \begin{bmatrix} \frac{1}{6}h^3 \\ \frac{1}{2}h^2 \\ h \end{bmatrix} \end{aligned} \tag{20}$$

The linear system in (20) can be readily extended to higher dimensions by denoting $\tilde{y}_i = (y_i, \dot{y}_i, \ddot{y}_i)$, $\tilde{u}_i^y = \dddot{y}_i$, $\tilde{\mathbf{x}}_\mathbf{i} = (\tilde{x}_i, \tilde{y}_i)$, $\tilde{\mathbf{u}}_\mathbf{i} = (\tilde{u}_i^x, \tilde{u}_i^y)$.

$$\begin{aligned} \tilde{\mathbf{x}}_{\mathbf{i+1}} &= \tilde{A}\tilde{\mathbf{x}}_\mathbf{i} + \tilde{B}\tilde{\mathbf{u}}_\mathbf{i} \\ \tilde{A} = \begin{bmatrix} A & \\ & A \end{bmatrix} & \quad \tilde{B} = \begin{bmatrix} B & \\ & B \end{bmatrix} \end{aligned} \tag{21}$$

By changing the optimization variables in (10) from spline coefficients (a_{ij}, b_{ij}) to $(\tilde{\mathbf{x}}_\mathbf{i}, \tilde{\mathbf{u}}_\mathbf{i})$, and replacing the continuity constraints defined in (10d)-(10f) to the LTI system (21), we can reformulate (10) as an optimal control problem:

$$\min_{\tilde{\mathbf{x}}_\mathbf{i},\, \tilde{\mathbf{u}}_\mathbf{i}} \quad \sum_{i=0}^{N} w_1\left(\ddot{x}_i^2 + \ddot{y}_i^2\right) + w_2\left(\dddot{x}_i^2 + \dddot{y}_i^2\right) \tag{22a}$$

$$\text{s.t.} \quad x_0 = x_{init}, \quad y_0 = y_{init}, \tag{22b}$$

$$x_N = x_{goal}, \quad y_N = y_{goal}, \tag{22c}$$

$$\tilde{\mathbf{x}}_{\mathbf{i+1}} = \tilde{A}\tilde{\mathbf{x}}_\mathbf{i} + \tilde{B}\tilde{\mathbf{u}}_\mathbf{i}, \tag{22d}$$

$$-\Delta_l \le -(x_i - \bar{x}_i)\sin\bar{\theta}_i + (y_i - \bar{y}_i)\cos\bar{\theta}_i \le \Delta_r, \tag{22e}$$

$$-\Delta_b \le (x_i - \bar{x}_i)\cos\bar{\theta}_i + (y_i - \bar{y}_i)\sin\bar{\theta}_i \le \Delta_f, \tag{22f}$$

$$-\dot{x}_0\sin\theta_{init} + \dot{y}_0\cos\theta_{init} = 0, \tag{22g}$$

$$\lambda_0\ddot{x}_0 + \mu_0\ddot{y}_0 = \kappa_{init}, \tag{22h}$$

$$\underline{\kappa} \le \lambda_i\ddot{x}_i + \mu_i\ddot{y}_i \le \bar{\kappa} \tag{22i}$$

As in (10), (22b) and (22c) are boundary constraints, (22d) is the linear system constraint, and (22e) and (22f) define two affine constraints for lateral and longitudinal offset, respectively. Additionally, (22g) and (22h) respectively define constraints for heading and curvature at the starting state. (22i) defines the curvature constraints along the path. As denoted in (12), (λ_i, μ_i) are functions of $\dot{x}$ and $\dot{y}$, and their values are updated through iterations, except that they are approximated by the reference path at the first iteration.

2) Computational Complexity for Optimal Control Problem (22): Next, we will show that the convex quadratic program (22) can be solved in $O(N(n_x + n_u + n_c)^3)$, where N is the horizon length, n_x and n_u are the number of states and control inputs, and n_c is the number of constraints except the system model at each step. A brief introduction to the primal-dual interior point method is presented here. More

details can be found in [17], [31]. The problem defined in (22) can be rearranged into a generic convex quadratic form.

$$\begin{aligned} \min_{w} \quad & \frac{1}{2} w^T Q w + c^T w \\ \text{s.t.} \quad & Fw = g, \\ & Cw \leq d \end{aligned} \tag{23}$$

The Karush–Kuhn–Tucker (KKT) conditions yield

$$\begin{aligned} Qw + c + F^T \pi + C^T \lambda &= 0 \\ -Fw + g &= 0 \\ -Cw + d - t &= 0 \\ T\Lambda e &= 0 \\ (\lambda, t) &\geq 0 \end{aligned} \tag{24}$$

where π and λ are the Lagrange multipliers of the equality and inequality constraints, respectively. t is a vector of slack variables for the inequality constraints. $T = \text{diag}(t_1, t_2, \ldots, t_n)$, $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, $e = (1, 1, \ldots, 1)^T$, and n is the number of inequality constraints. The primal-dual interior point method uses Newton's method to solve the system $f(w, \pi, \lambda, t) = 0$ defined in (24) iteratively. The Newton direction is computed by solving the following linear system.

$$\begin{bmatrix} Q & F^T & C^T & 0 \\ -F & 0 & 0 & 0 \\ -C & 0 & 0 & -I \\ 0 & 0 & T & \Lambda \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta \pi \\ \Delta \lambda \\ \Delta t \end{bmatrix} = \begin{bmatrix} r_Q \\ r_F \\ r_C \\ r_T \end{bmatrix} \tag{25}$$

The right-hand-side vector is denoted as (r_Q, r_F, r_C, r_T), which decides different search directions. The basic primal-dual method uses the pure Newton direction. The Mehrotra predictor-corrector algorithm combines Newton and centering directions. By eliminating $\Delta\lambda$ and Δt in (25), a reduced KKT system is obtained.

$$\begin{bmatrix} Q + C^T \Lambda T^{\ 1} C & F^T \\ -F & 0 \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta \pi \end{bmatrix} = \begin{bmatrix} r_Q - C^T T^{\ 1} (\Lambda r_C + r_T) \\ r_F \end{bmatrix} \tag{26}$$

For the recurrent spline optimization (22), (26) can be rewritten in block tridiagonal form [17]. The block tridiagonal matrix allows the linear system (26) to be solved in $O(N(n_x + n_u + n_c)^3)$ using the Schur-based method (e.g. fast-mpc [32], FORCES [19]) or Riccati-based method (e.g. hpipm [33]), as opposed to $O(N^3(n_x + n_u + n_c)^3)$ for KKT systems with generic matrix form.

IV. Experimental Results

The proposed algorithms are tested on two types of problems: trajectory generation and motion planning. Trajectory generation is a boundary value problem where only the start and goal state, and the system model and actuation limit are considered. Additionally, motion planning also handles obstacles. For each problem, two scenarios are included: tight lane change (2m lateral offset within 6m) and sharp turn (a 10m-by-4m 90-degree turn). We compare the two methods proposed in this paper (quadratic spline optimization and recurrent spline optimization) to four other methods: κ-poly (curvature polynomial from [8]), iLQR (extended LQR from [13]), acado (ACADO [34] with default settings, an active-set based multiple-shooting approach), and ms-ipopt [35] (an interior-point based multiple-shooting approach, implemented with CasADi [36]). The kinematic bicycle model (1) is used for these four methods. The tests are implemented on a Desktop PC with an Intel i7-9700K CPU (8 core 3.60GHz) and 32GB memory.

A. Trajectory Generation

TABLE I

Runtime comparison for trajectory generation with different horizon.

	Lane Change [ms]			Sharp Turn [ms]		
N	10	40	160	10	40	160
κ-poly [8]	0.32	0.32	**0.38**	0.41	0.42	**0.46**
iLQR [13]	2.18	3.13	4.63	failed	2.41	8.31
ms-ipopt [35]	3.24	4.82	11.26	7.62	17.64	11.66
acado [34]	9.29	56.22	923.41	16.59	72.16	1275.74
spline-qp (10)	0.46	1.94	9.29	0.42	1.53	8.23
spline-ocp (22)	**0.08**	**0.21**	0.62	**0.11**	**0.24**	0.69

TABLE II

Runtime comparison for trajectory generation with vs. without curvature constraints for N=40.

	Lane Change [ms]		Sharp Turn [ms]	
Curvature	w/o	w/	w/o	w/
κ-poly [8]	0.32	N/A	0.42	N/A
iLQR [13]	3.13	N/A	2.41	N/A
ms-ipopt [35]	4.82	6.43	4.24	8.14
acado [34]	56.22	68.63	72.16	102.33
spline-qp (10)	1.94	6.24	1.53	18.83
spline-ocp (22)	**0.21**	**0.57**	**0.24**	**1.14**

We first compare the runtime for a boundary problem with only start and goal state as constraints in Table I, where N is the number of steps. The states are 2D position, heading, and curvature. The number of optimization variables is fixed for the curvature polynomial method (denoted as κ-poly), and is linear in N for all other methods.

Runtime for κ-poly grows slowly with N as only integration steps are increased. The control input (curvature) in κ-poly is represented as a single polynomial, which reduces the variables to be optimized. However, the trade-off is that it also reduces the flexibility of the trajectory. It may be unable to generate trajectories with desirable curvature at some sharp turns. In contrast, our spline-based method can generate curves of any shape.

The acado method is the slowest one among all methods because the condensing technology it uses is not beneficial for our problem type. The active-set method used by acado also does not scale well with horizon. The acado method is more suitable for control problems where the needed horizon is much less than that for the planning problem.

The iLQR method fails to converge for the sharp turn test with $N = 10$. This is because as N is reduced, the gap between two consecutive states increases. Therefore,

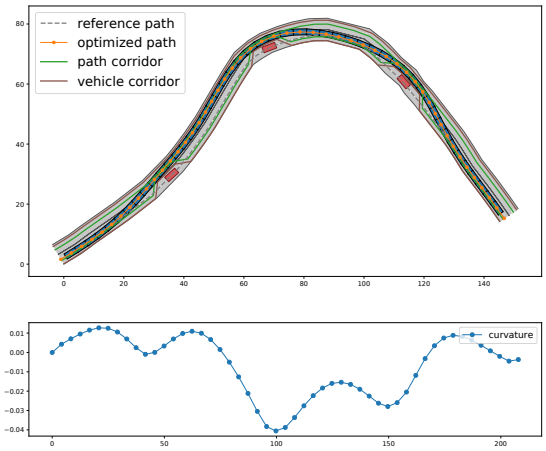

Fig. 2. Avoiding static obstacles on a curvy road with 0.53 ms runtime.

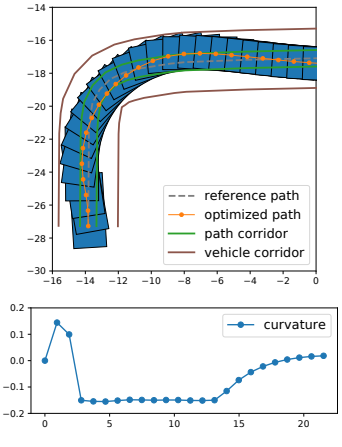

Fig. 3. Making a sharp turn while staying in lane.

iLQR becomes more sensitive to the integration error due to linearizing the nonlinear kinematic model. In contrast, our spline-based methods have a linear system model thanks to differential flatness theory, which allows a large gap between states.

The ms-ipopt method takes longer for $N = 40$ than $N = 160$ in the sharp turn test. This is another example of the NLP problem in iLQR and ms-ipopt having a less consistent convergence rate compared to our convex QP formulation. The proposed spline-ocp method has the lowest runtime at $N = 10$, and it grows linearly with horizon N.

Table II shows the runtime when we add curvature constraints along the path. The values for iLQR and κ-poly are not available since they cannot add curvature constraints. For the proposed methods, it usually takes 2-3 iterations for difficult cases to satisfy the curvature constraints, as is reflected in the runtimes.

B. Motion Planning with Obstacles

We also compare runtime for motion planning with obstacles. κ-poly and iLQR are not included again since their formulations do not allow path constraints. A collision-free corridor is first generated based on vertical cell decomposition, as described in III-C. The constraints formed by the corridor are convex for each state. The results are shown in Table III. Both acado and ms-ipopt fail to converge at $N = 10$ in the lane change test. This is due to a combination of the large linearization error, as explained before, coupled with the curvature and obstacle constraints making the problem harder to solve.

Two qualitative examples are shown in Fig 2 and Fig 3, which are from the CommonRoad benchmark [37]. Fig 2 shows a scenario where the vehicle avoids multiple obstacles on a curvy road. In this example, it takes 0.53 ms to finish the optimization for a 200 m path with $N = 50$. The average gap between two states is 4 m, which is much larger than a normal MPC setting. This is thanks to the linear spline model in our method, as mentioned before. Fig 3 shows the autonomous vehicle making a sharp turn while staying in lane and obeying curvature. The maximum curvature is set to 0.15. The planned path cuts the corner of the corridor nicely to reach the minimal turning radius.

TABLE III
RUNTIME COMPARISON FOR PLANNING WITH OBSTACLES.

	Lane Change [ms]			Sharp Turn [ms]		
N	10	40	160	10	40	160
ms-ipopt [35]	failed	6.78	19.96	5.09	9.74	34.48
acado [34]	failed	68.43	1123.14	18.44	142.85	9428.77
spline-qp (10)	2.15	9.76	46.85	2.92	22.42	397.76
spline-ocp (22)	**0.27**	**0.63**	**2.92**	**0.32**	**1.35**	**6.76**

V. CONCLUSIONS

Our work is inspired by exploring the relationship between smoothing splines [26], iLQR [11], and multiple shooting-based trajectory optimization [14]. It can be viewed as an extension of the smoothing spline (with additional path and curvature constraints), with a multiple shooting formulation, and solved in a LQR way. It provides a new way to solve a normal spline smoothing problem. It can also be used to smooth reference path in addition to the trajectory generation and motion planning applications shown in this paper.

The way we use the reference path is different from that of many other methods. Unlike most of the other planning approaches, the reference path in our method also doesn't need to be collision-free. We use the reference to help with defining obstacle and curvature constraints, and it can be easily obtained. For example, the reference path can be chosen as the straight line connecting start and goal state for trajectory generation problems, and the center line of the road for motion planning problems.

Several directions may be explored as future work. Our method can potentially be extended to trajectory planning with dynamic obstacles by replacing the arc length s with time t. However, it is hard to guess the final time horizon before the optimization is finished. Therefore, the total time also needs to be optimized, which makes the problem very nonlinear and non-convex. It is also difficult to convert the dynamic obstacles into convex constraints. Our method can also be extended to 3D space similar to the extension from 1D to 2D in (21).

REFERENCES

[1] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," *Advances in Neural Information Processing Systems (NIPS)*, vol. 16, 2004.

[2] S. LaValle and J. Kuffner Jr, "Randomized kinodynamic planning," in *Robotics and Automation (ICRA), IEEE International Conference on*, vol. 1, 1999, pp. 473–479.

[3] N. Ratliff, M. Zucker, J. A. D. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.

[4] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[5] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9. Citeseer, 2013, pp. 1–10.

[6] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2010, pp. 987–993.

[7] B. Nagy and A. Kelly, "Trajectory generation for car-like robots using cubic curvature polynomials," *Field and Service Robots*, vol. 11, 2001.

[8] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *The International Journal of Robotics Research*, vol. 22, no. 7-8, p. 583, 2003.

[9] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo em motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[10] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[11] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 300–306.

[12] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[13] J. P. van den Berg, "Extended lqr: Locally-optimal feedback control for systems with non-linear dynamics and non-quadratic cost," in *International Symposium on Robotics Research (ISRR)*, 2013.

[14] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 65–93.

[15] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[16] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[17] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of optimization theory and applications*, vol. 99, no. 3, pp. 723–757, 1998.

[18] J. Mattingley and S. Boyd, "Cvxgen: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[19] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 668–674.

[20] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[21] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot motion planning and control*. Springer, 1998, pp. 171–253.

[22] P. Rouchon, M. Fliess, J. Lévine, and P. Martin, "Flatness, motion planning and trailer systems," in *IEEE Conference on Decision and Control*, vol. 3. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEE), 1993, pp. 2700–2700.

[23] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[24] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[25] B. Chazelle, "Approximation and decomposition of shapes," *Algorithmic and Geometric Aspects of Robotics*, vol. 1, pp. 145–185, 1985.

[26] C. H. Reinsch, "Smoothing by spline functions," *Numerische mathematik*, vol. 10, no. 3, pp. 177–183, 1967.

[27] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[28] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[29] A. Nemirovskii and Y. Nesterov, "Interior point polynomial algorithms in convex programming," *Society for Industrial and Applied Mathematics*, 1994.

[30] M. Andersen, J. Dahl, Z. Liu, L. Vandenberghe, S. Sra, S. Nowozin, and S. Wright, "Interior-point methods for large-scale cone programming," *Optimization for machine learning*, vol. 5583, 2011.

[31] G. Frison, "Numerical methods for model predictive control," Ph.D. dissertation, University of Padua, 2012.

[32] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on control systems technology*, vol. 18, no. 2, pp. 267–278, 2009.

[33] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," 2020.

[34] B. Houska, H. J. Ferreau, and M. Diehl, "Acado toolkit-an open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

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

[36] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[37] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719 – 726.