A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories

Nitin R. Kapania[1]
Department of Mechanical Engineering,
Stanford University,
Stanford, CA 94305
e-mail: nkapania@stanford.edu

John Subosits
Department of Mechanical Engineering,
Stanford University,
Stanford, CA 94305
e-mail: subosits@stanford.edu

J. Christian Gerdes
Department of Mechanical Engineering,
Stanford University,
Stanford, CA 94305
e-mail: gerdes@stanford.edu

The problem of maneuvering a vehicle through a race course in minimum time requires computation of both longitudinal (brake and throttle) and lateral (steering wheel) control inputs. Unfortunately, solving the resulting nonlinear optimal control problem is typically computationally expensive and infeasible for real-time trajectory planning. This paper presents an iterative algorithm that divides the path generation task into two sequential subproblems that are significantly easier to solve. Given an initial path through the race track, the algorithm runs a forward–backward integration scheme to determine the minimum-time longitudinal speed profile, subject to tire friction constraints. With this fixed speed profile, the algorithm updates the vehicle's path by solving a convex optimization problem that minimizes the resulting path curvature while staying within track boundaries and obeying affine, time-varying vehicle dynamics constraints. This two-step process is repeated iteratively until the predicted lap time no longer improves. While providing no guarantees of convergence or a globally optimal solution, the approach performs very well when validated on the Thunderhill Raceway course in Willows, CA. The predicted lap time converges after four to five iterations, with each iteration over the full 4.5 km race course requiring only 30 s of computation time on a laptop computer. The resulting trajectory is experimentally driven at the race circuit with an autonomous Audi TTS test vehicle, and the resulting lap time and racing line are comparable to both a nonlinear gradient descent solution and a trajectory recorded from a professional racecar driver. The experimental results indicate that the proposed method is a viable option for online trajectory planning in the near future. [DOI: 10.1115/1.4033311]

1 Introduction

The problem of calculating the minimum lap time trajectory for a given vehicle and race track has been studied over the last several decades in the control, optimization, and vehicle dynamics communities. Early research by Hendrikx et al. [1] in 1996 used Pontryagin's minimum principle to derive coupled differential equations to solve for the minimum-time trajectory of a vehicle lane change maneuver. The minimum lap time problem drew significant interest from professional racing teams, and Casanova [2] published a method in 2000 capable of simultaneously optimizing both the path and speed profile for a fully nonlinear vehicle model using nonlinear programing (NLP). Kelly [3] further extended the results from Casanova by considering the physical effect of tire thermodynamics and applying more robust NLP solution methods such as feasible sequential quadratic programing. More recently, Perantoni and Limebeer [4] showed that the computational expense could be significantly reduced by applying curvilinear track coordinates, nonstiff vehicle dynamics, and the use of smooth computer-generated analytic derivatives. The authors simulated the optimal vehicle lap on the Catalunya race circuit with a spatial discretization of 2 m and a solve time of around 15 min.

More recently, the development of autonomous vehicle technology at the industry and academic level has led to research on optimal path planning algorithms that can be used for driverless cars. Theodosis and Gerdes published a gradient descent approach for determining time-optimal racing lines, with the racing line constrained to be composed of a fixed number of clothoid segments

[5]. Given the computational expense of performing nonlinear optimization, there has also been a significant research effort to find approximate methods that provide fast lap times. Timings and Cole [6] formulated the minimum lap time problem into a model predictive control (MPC) problem by linearizing the nonlinear vehicle dynamics at every time step and approximating the minimum-time objective by maximizing distance traveled along the path centerline. The resulting racing line for a 90 deg turn was simulated next to an NLP solution. Gerdts et al. [7] proposed a similar receding horizon approach, where the distance along a reference path was maximized over a series of locally optimal optimization problems that were combined with continuity boundary conditions. One potential drawback of the MPC approach is that an optimization problem must be reformulated and solved at every time step, which can still be computationally expensive. For example, Timings and Cole reported a computation time of 900 ms per 20 ms simulation step with the cplex quadratic program solver on a desktop personal computer.

While experimental validation was reported only by Theodosis and Gerdes [5] and Gerdts et al. [7], all of the aforementioned methods are feasible for experimental implementation, as an autonomous vehicle can apply a closed-loop controller to follow a time-optimal vehicle trajectory computed offline. However, there are significant benefits to developing a fast trajectory generation algorithm that can approximate the globally optimal trajectory in real-time. If the algorithm runtime is small compared to the actual lap time, the algorithm can run as a real-time trajectory planner and find a fast racing line for the next several turns of the racing circuit. This would allow the trajectory planner to modify the desired path based on the motion of competing race vehicles and estimates of road friction, tire wear, engine/brake dynamics, and other parameters learned over several laps of racing. Additionally, the fast trajectory algorithm can be used to provide a very good initial trajectory for a nonlinear optimization method.

This paper therefore presents an experimentally validated iterative algorithm that generates vehicle racing trajectories with low computational expense. To decrease computation time, the combined lateral/longitudinal optimal control problem is replaced by two sequential subproblems where minimum-time longitudinal speed inputs are computed given a fixed vehicle path, and then, the vehicle path is updated given the fixed speed commands.

Section 2 presents a mathematical framework for the trajectory generation problem and provides a linearized five-state model for the planar dynamics of a racecar following a set of speed and steering inputs, with lateral and heading error states computed with respect to a fixed path. Section 3 describes the method of finding the minimum-time speed inputs given a fixed path. While this task has been recently formulated as a convex optimization problem [8], a forward–backward integration scheme based on prior work [9] is used instead. Section 4 describes a method for updating the racing path given the fixed speed inputs using convex optimization, where the curvature norm of the driven path is explicitly minimized. The complete algorithm is outlined in Sec. 5, and the racing trajectory generated on the Thunderhill Raceway circuit in Willows, CA is compared to results from both a nonlinear optimization and data recorded from a professional racecar driver. In Sec. 6, the racing trajectory is then tested experimentally in an autonomous Audi TTS testbed via a previously published closed-loop path following controller [10]. The resulting lap time compares well with the lap time recorded for the nonlinear optimization trajectory. Section 7 concludes by discussing future implementation of the algorithm in a real-time path planner.

2 Path Description and Vehicle Model

Figure 1 describes the parameterization of the reference path that the vehicle will follow. The reference path is most intuitively described in Fig. 1(a) as a smooth curve of Cartesian east–north

coordinates, with road boundaries represented by similar Cartesian curves. However, for the purposes of quickly generating a racing trajectory, it is more convenient to parameterize the reference path as a curvature profile K that is a function of distance along the path s (Fig. 1(c)). Additionally, it is convenient to store the road boundary information as two functions $w_{in}(s)$ and $w_{out}(s)$, which correspond to the lateral distance from the path at s to the inside and outside road boundaries, respectively (Fig. 1(b)). This maximum lateral distance representation will be useful when constraining the generated racing path to lie within the road boundaries. The transformation from the local s, K coordinate frame to the global Cartesian coordinates E, N is given by the Fresnel integrals

$$E(s) = \int_0^s - \sin(\Psi_r(z))dz \tag{1a}$$

$$N(s) = \int_0^s \cos(\Psi_r(z))dz \tag{1b}$$

$$\Psi_r(s) = \int_0^s K(z)dz \tag{1c}$$

where $\Psi_r(s)$ is the heading angle of the reference path and z is a dummy variable.

With the reference path defined in terms of s and K, the next step is to define the dynamic model of the vehicle. For the purposes of trajectory generation, we assume the vehicle dynamics are given by the planar bicycle model in Fig. 2(a), with yaw rate r and sideslip β states describing the lateral dynamics. Additionally, the vehicle's offset from the reference path is given by the path lateral deviation state e and path heading error state $\Delta\Psi$ (Fig. 2(b)). Linearized equations of motion for all four states are given by

$$\dot{\beta} = \frac{F_{yf} + F_{yr}}{mU_x} - r \qquad \dot{r} = \frac{aF_{yf} - bF_{yr}}{I_z} \tag{2a}$$
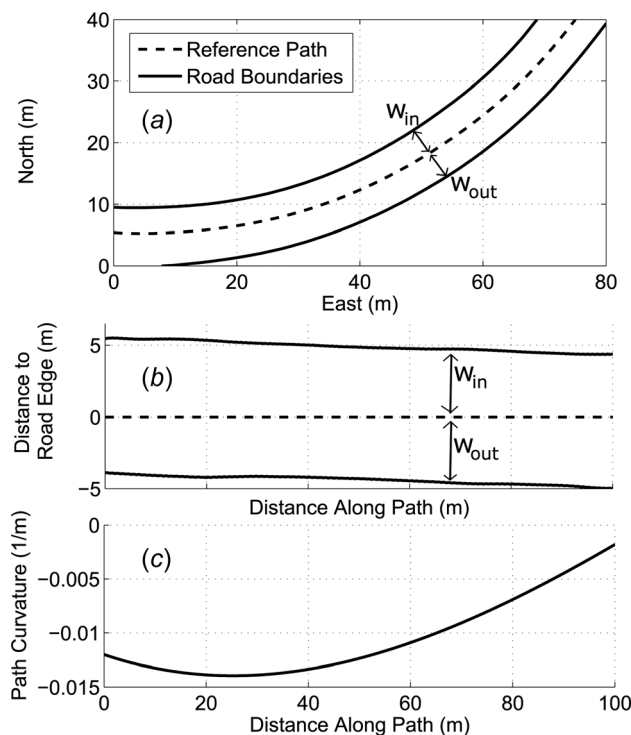

Fig. 1 (a) View of a sample reference path and road boundaries, plotted in the east–north Cartesian frame. (b) Lateral distance from path to inside road edge (positive) and outside road edge (negative) as a function of distance along path. (c) Curvature as a function of distance along path.
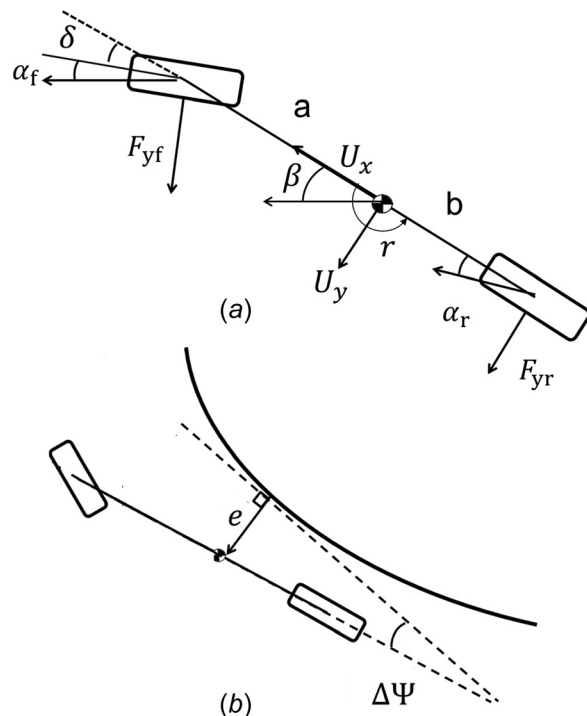

Fig. 2 (a) Schematic of bicycle model. (b) Diagram showing lateral path deviation e and path heading error $\Delta\Psi$ states.

$$\dot{e} = U_x(\beta + \Delta\Psi) \qquad \Delta\dot{\Psi} = r - U_x K \qquad (2b)$$

where U_x is the vehicle forward velocity and F_{yf} and F_{yr} are the front and rear lateral tire forces. The vehicle mass and yaw inertia are denoted by m and I_z, and the geometric parameters a and b are shown in Fig. 2(a). Note that while the vehicle longitudinal dynamics are not explicitly modeled, the bicycle model does allow for time-varying values of U_x.

3 Velocity Profile Generation Given Fixed Reference Path

Given a fixed reference path described by s and K, the first algorithm step is to find the minimum-time speed profile the vehicle can achieve without exceeding the available tire–road friction. The approach taken in this paper is a "three-pass" approach described in complete detail by Subosits and Gerdes [9], and originally inspired by the work of Velenis and Tsiotras [11]. Given the lumped front and rear tires from Fig. 2(a), the available longitudinal force F_x and lateral force F_y at each wheel are constrained by the friction circle

$$F_{xf}^2 + F_{yf}^2 \leq (\mu F_{zf})^2 \qquad (3a)$$

$$F_{xr}^2 + F_{yr}^2 \leq (\mu F_{zr})^2 \qquad (3b)$$

where μ is the tire–road friction coefficient and F_z is the available normal force. The first pass of the speed profile generation aims to find the maximum permissible steady-state vehicle speed given zero longitudinal force. For the simplified case where weight transfer and topography effects are neglected, this is given by

$$U_x(s) = \sqrt{\frac{\mu g}{|K(s)|}} \qquad (4)$$

where the result in Eq. (4) is obtained by setting $F_{yf} = [mb/(a+b)]U_x^2 K$ and $F_{zf} = mgb/(a+b)$. The results of this first pass for the sample curvature profile in Fig. 3(a) are shown in Fig. 3(b). The next step is a forward integration step, where the velocity of a given point is determined by the velocity of the previous point and the available longitudinal force $F_{x,\text{max}}$ for acceleration. This available longitudinal force is calculated in Ref. [9] by accounting for the vehicle engine force limit and the lateral force demand on all tires due to the road curvature

$$U_x(s + \Delta s) = \sqrt{U_x^2(s) + 2\frac{F_{x,\text{accel,max}}}{m}\Delta s} \qquad (5)$$

A key point of the forward integration step is that at every point, the value of $U_x(s)$ is compared to the corresponding value from Eq. (4), and the minimum value is taken. The result is shown graphically in Fig. 3(c). Finally, the backward integration step occurs, where the available longitudinal force for deceleration is again constrained by the lateral force demand on all tires

$$U_x(s - \Delta s) = \sqrt{U_x^2(s) - 2\frac{F_{x,\text{decel,max}}}{m}\Delta s} \qquad (6)$$

The value of $U_x(s)$ is then compared to the corresponding value from Eq. (5) for each point along the path, and the minimum value is chosen, resulting in the final velocity profile shown by the solid line in Fig. 3(d). While treatment of three-dimensional topography effects is not described in this paper, the method described in Ref. [9] and used for the experimental data collection determines the normal and lateral tire forces F_z and F_y at each point along the path by accounting for weight transfer, road bank, and grade.
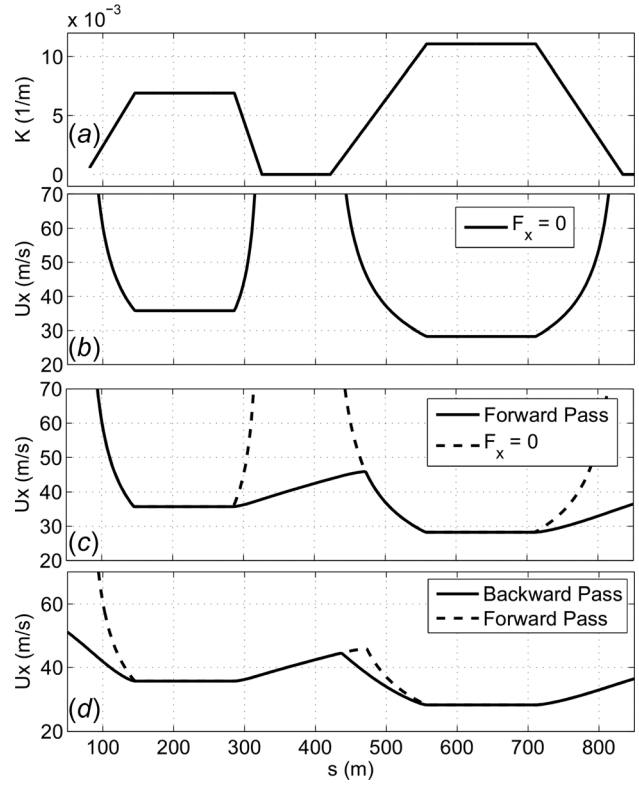

Fig. 3 (a) Sample curvature profile. (b) Velocity profile given zero longitudinal force. (c) Velocity profile after forward pass. (d) Final velocity profile after backward pass.

4 Updating Path Given Fixed Velocity Profile

4.1 Overall Approach and Minimum Curvature Heuristic. The second step of the trajectory generation algorithm takes the original reference path $K(s)$ and corresponding velocity profile $U_x(s)$ as inputs and modifies the reference path to obtain a new, ideally faster, racing line. Sharp [12] suggests a general approach for modifying an initial path to obtain a faster lap time by taking the original path and velocity profile and incrementing the speed uniformly by a small, constant "learning rate." An optimization problem is then solved to find a new reference path and control inputs that allow the vehicle to drive at the higher speeds without driving off the road. If a crash is detected, the speed inputs are locally reduced around the crash site and the process is repeated.

However, one challenge with this approach is that it can take several hundred iterations of locally modifying the vehicle speed profile, detecting crashes, and modifying the reference path to converge to a fast lap time. An alternative approach is to modify the reference path in one step by solving a single optimization problem. The lap time t for a given racing line is provided by the following equation:

$$t = \int_0^L \frac{ds}{U_x(s)} \qquad (7)$$

Equation (7) implies that minimizing the vehicle lap time requires simultaneously minimizing the total path length L while maximizing the vehicle's longitudinal velocity U_x. These are typically competing objectives, as lower curvature (i.e., higher radius) paths can result in longer path lengths but higher vehicle speeds when the lateral force capability of the tires is reached, as shown in Eq. (4). As mentioned in the Introduction, time-intensive NLP is required to manage this tradeoff and directly minimize Eq. (7).

The proposed approach is therefore to simplify the cost function by only minimizing the norm of the vehicle's driven curvature

$K(s)$ at each path modification step. Path curvature can be easily formulated as a convex function with respect to the vehicle state vector x, enabling the path modification step to be easily solved by leveraging the computational speed of convex optimization.

However, minimizing curvature is not the same as minimizing lap time and provides no guarantee of finding the time-optimal solution. The proposed cost function relies on the hypothesis that a path with minimum curvature is a good approximation for the minimum-time racing line. Lowering the curvature of the racing line is more important than minimizing path length for most race courses, as the relatively narrow track width provides limited room to shorten the overall path length. Simulated and experimental results in Secs. 5 and 6 will validate this hypothesis by showing similar lap time performance when compared to a gradient descent method that directly minimizes lap time. One reason for this comparable performance is that nonlinear methods are typically sensitive to initial conditions and are themselves only guaranteed to find a locally optimal solution.

4.2 Convex Problem Formulation. Formulating the path update step as a convex optimization problem requires an affine, discretized form of the bicycle model in Eq. (2). The equations of motion are nonlinear because the front and rear lateral tire forces become saturated as the vehicle drives near the limits of tire adhesion. The well-known brush tire model [13] captures the force saturation of tires as a function of lateral tire slip angle α as follows:

$$F_{y\diamond} = \begin{cases} -C_\diamond \tan\alpha_\diamond + \dfrac{C_\diamond^2}{3\mu F_{z\diamond}} |\tan\alpha_\diamond| \tan\alpha_\diamond \\ \quad - \dfrac{C_\diamond^3}{27\mu^2 F_{z\diamond}^2} \tan^3\alpha_\diamond, \quad |\alpha_\diamond| < \arctan\left(\dfrac{3\mu F_{z\diamond}}{C_\diamond}\right) \\ -\mu F_{z\diamond} \operatorname{sgn}\alpha_\diamond, \qquad \text{otherwise} \end{cases} \quad (8)$$

where the symbol $\diamond \in [f, r]$ denotes the lumped front or rear tire, and $C_\diamond$ is the corresponding tire cornering stiffness. The linearized tire slip angles α_f and α_r are functions of the vehicle lateral states and the steer angle input, δ

$$\alpha_f = \beta + \frac{ar}{U_x} - \delta \quad (9a)$$

$$\alpha_r = \beta - \frac{br}{U_x} \quad (9b)$$

The tire model (8) can be linearized at every point along the reference path assuming steady-state cornering conditions

$$F_{y\diamond} = \tilde{F}_{y\diamond} - \tilde{C}_\diamond(\alpha_\diamond - \tilde{\alpha}_\diamond) \quad (10a)$$

$$\tilde{F}_{y\diamond} = \frac{F_{z\diamond}}{g} U_x^2 K \quad (10b)$$

with parameters $\tilde{F}_y$, $\tilde{\alpha}$, and $\tilde{C}$ shown in Fig. 4. The affine, continuous bicycle model with steering input δ is then written in state-space form as

$$\dot{x}(t) = A(t)x + B(t)\delta + d(t) \quad (11a)$$

$$x = [e \quad \Delta\Psi \quad r \quad \beta \quad \Psi]^T \quad (11b)$$

where we have added a fifth state, vehicle heading angle Ψ, defined as the time integral of yaw rate r. This makes explicit computation of the minimum curvature path simpler. The state matrices $A(t)$, $B(t)$, and affine term $d(t)$ are given by
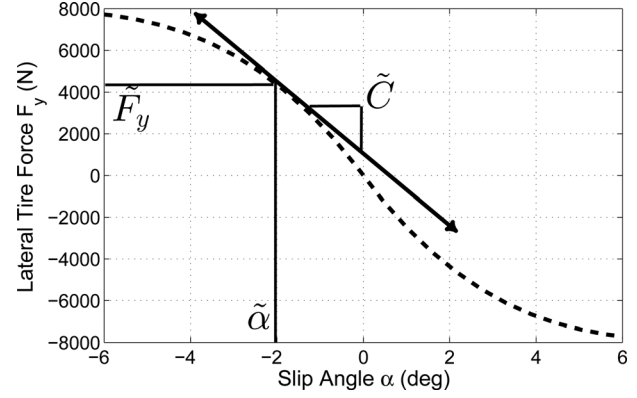
Fig. 4 Nonlinear tire force curve given by Fiala model, along with affine tire model linearized at $\alpha = \tilde{\alpha}$

$$A(t) = \begin{bmatrix} 0 & U_x(t) & 0 & U_x(t) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-\left(a^2\tilde{C}_f(t) + b^2\tilde{C}_r(t)\right)}{U_x(t)I_z} & \dfrac{b\tilde{C}_r(t) - a\tilde{C}_f(t)}{I_z} & 0 \\ 0 & 0 & \dfrac{b\tilde{C}_r(t) - a\tilde{C}_f(t)}{mU_x^2(t)} - 1 & \dfrac{-\left(\tilde{C}_f(t) + \tilde{C}_r(t)\right)}{mU_x(t)} & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (12)$$

$$B(t) = \begin{bmatrix} 0 & 0 & \dfrac{a\tilde{C}_f(t)}{I_z} & \dfrac{\tilde{C}_f(t)}{mU_x(t)} & 0 \end{bmatrix}^T \quad (13)$$

$$d(t) = \begin{bmatrix} 0 \\ -K(t)U_x(t) \\ \dfrac{a\tilde{C}_f(t)\tilde{\alpha}_f(t) - b\tilde{C}_r(t)\tilde{\alpha}_r(t) + a\tilde{F}_{yf}(t) - b\tilde{F}_{yr}(t)}{I_z} \\ \dfrac{\tilde{C}_f(t)\tilde{\alpha}_f(t) + \tilde{C}_r(t)\tilde{\alpha}_r(t) + \tilde{F}_{yf}(t) + \tilde{F}_{yr}(t)}{mU_x(t)} \\ 0 \end{bmatrix} \quad (14)$$

With the nonlinear model now approximated as an affine, time-varying model, updating the path is accomplished by solving the following convex optimization problem:

$$\underset{\delta,x}{\text{minimize}} \quad \sum_k \left(\frac{\Psi_k - \Psi_{k-1}}{s_k - s_{k-1}}\right)^2 + \lambda(\delta_k - \delta_{k-1})^2 \quad (15a)$$

$$\text{subject to} \quad x_{k+1} = A_k x_k + B_k \delta_k + d_k \quad (15b)$$

$$w_k^{out} \leq e_k \leq w_k^{in} \quad (15c)$$

$$x_1 = x_T \quad (15d)$$

where $k = 1 \cdots T$ is the discretized time index, and A_k, B_k, and d_k are discretized versions of the continuous state-space equations in Eq. (11). The objective function (15a) minimizes the curvature norm of the path driven by the vehicle, as path curvature is the derivative of the vehicle heading angle with respect to distance along the path s (1c). To maintain convexity of the objective function, the term $s_k - s_{k-1}$ is a constant rather than a variable and is updated for the next iteration after the optimization has been completed (see Sec. 5). Additionally, there is a regularization term

with weight λ added in the cost function to ensure a smooth steering profile for experimental implementation.

The equality constraint (15b) ensures that the vehicle follows the affine lateral dynamics. The inequality constraint (15c) allows the vehicle to deviate laterally from the reference path to find a new path with lower curvature, but only up to the road edges. Finally, the equality constraint (15d) is required for complete racing circuits to ensure the generated racing line is a continuous loop. The results of running the optimization are shown for an example turn in Fig. 5. The reference path starts out at the road centerline, and the optimization finds a modified path that uses all the available width of the road to lower the path curvature.

5 Algorithm Implementation and Simulated Results

5.1 Algorithm Implementation. The final algorithm for iteratively generating a vehicle racing trajectory is described in Algorithm 1. The input to the algorithm is any initial path through the racing circuit, parameterized in terms of distance along the path s, path curvature $K(s)$, and the lane edge distances w_{in} (s) and w_{out} (s) described in Fig. 1. Given the initial path, the minimum-time speed profile $U_x(s)$ is calculated as described in Fig. 3. Next, the path is modified by solving the previously described minimum curvature convex optimization problem (15).

Algorithm 1 Method for iterative trajectory generation

1: **procedure** GENERATETRAJECTORY$(s^\circ, K^\circ, w_{\text{in}}^\circ, w_{\text{out}}^\circ)$
2:     path $\leftarrow (s^\circ, K^\circ, w_{\text{in}}^\circ, w_{\text{out}}^\circ)$
3:     **while** $\Delta t^\star > \varepsilon$ **do**
4:         $U_x \leftarrow$ calculateSpeedProfile(path)
5:         path $\leftarrow$ minimizeCurvature(U_x, path)
6:         $t^\star \leftarrow$ calculateLapTime(U_x, path)
7:     **end while**
8:     **return** path, U_x
9: **end procedure**

The optimization only solves explicitly for the steering input $\delta^\star$ and resulting vehicle lateral states $x^\star$ at every time step. Included within $x^\star$ is the optimal vehicle heading $\Psi^\star$ and lateral deviation $e^\star$ from the initial path. To obtain the new path in terms of s and K, the east–north coordinates (E_k, N_k) of the updated vehicle path are updated as follows:

$$E_k \leftarrow E_k - e_k^\star \cos(\Psi_{r,k}) \qquad (16a)$$

$$N_k \leftarrow N_k - e_k^\star \sin(\Psi_{r,k}) \qquad (16b)$$
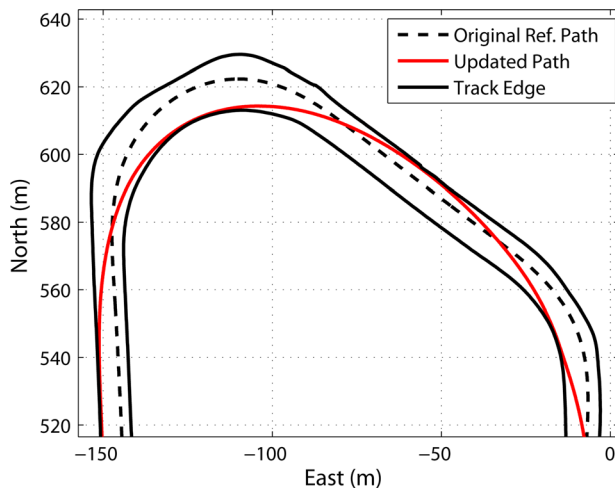

Fig. 5 Path update for an example turn

where Ψ_r is the path heading angle of the original path. Next, the new path is given by the following numerical approximation:

$$s_k = s_{k-1} + \sqrt{(E_k - E_{k-1})^2 + (N_k - N_{k-1})^2} \qquad (17a)$$

$$K_k = \frac{\Psi_k^\star - \Psi_{k-1}^\star}{s_k - s_{k-1}} \qquad (17b)$$

Notice that Eq. (17) accounts for the change in the total path length that occurs when the vehicle deviates from the original path. In addition to s and K, the lateral distances to the track edges w_{in} and w_{out} are different for the new path as well and are recomputed using the Cartesian coordinates for the inner and outer track edges and (E_k, N_k). The two-step procedure is iterated until the improvement in lap time $\Delta t^\star$ over the prior iteration is less than a small positive constant ε.

5.2 Algorithm Validation. The proposed algorithm is tested on the 4.5 km Thunderhill racing circuit in Willows, CA. The vehicle parameters used for the lap time optimization come from an Audi TTS experimental race vehicle (Fig. 6) and are shown along with the optimization parameters in Table 1. The initial path is obtained by collecting global positioning system (GPS) data of the inner and outer track edges and estimating the $(s, K, w_{\text{in}}, w_{\text{out}})$ parametrization of the track centerline via a separate curvature estimation subproblem similar to the one proposed in Ref. [4]. The algorithm is implemented in MATLAB, with the minimum curvature optimization problem (15) solved using the CVX software package [14].

5.3 Comparison With Other Methods. The generated racing path after five iterations is shown in Fig. 7. To validate the proposed algorithm, the racing line is compared with the results from a nonlinear gradient descent algorithm implemented by Theodosis and Gerdes [5] and an experimental trajectory recorded

Fig. 6 Audi TTS used for simulation parameters and experimental validation

Table 1 Optimization parameters

| Parameter	Symbol	Value	Units
Regularization parameter	λ	1	$1/m^2$
Stop criterion	ε	0.1	s
Vehicle mass	m	1500	kg
Yaw inertia	I_z	2250	kg/m^2
Front axle to CG	a	1.04	m
Rear axle to CG	b	1.42	m
Front cornering stiffness	C_f	160	kN rad^{-1}
Rear cornering stiffness	C_r	180	kN rad^{-1}
Friction coefficient	μ	0.95	—
Path discretization	Δs	2.75	m
Optimization time steps	T	1843	—
Max engine force	—	3750	N

CG: center of gravity.
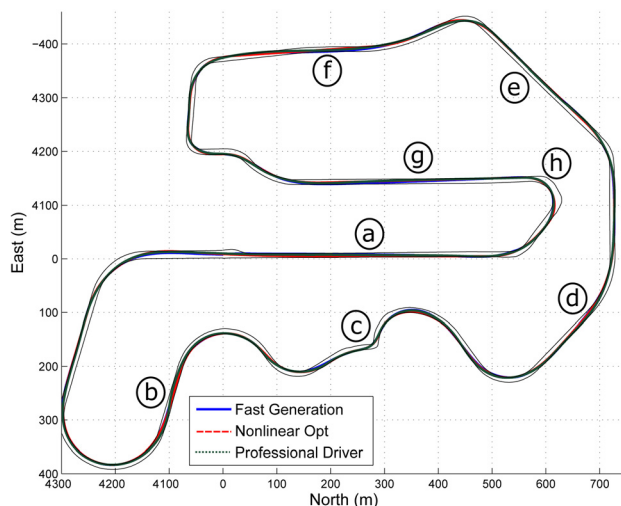
Fig. 7 Overhead view of Thunderhill Raceway along with generated path from algorithm. Car drives in alphabetical direction around the closed circuit. Labeled regions (a)–(h) are locations of discrepancies between the two-step algorithm solution and comparison solutions.

from a professional racecar driver in the testbed vehicle (Fig. 6). While time intensive to compute, the gradient descent approach generates racing lines with autonomously driven lap times within 1 s of lap times measured from professional racecar drivers.

To better visualize the differences between all three racing lines, Fig. 8 shows the lateral deviation from the track centerline as a function of distance along the centerline for all three trajectories. The left and right track boundaries w_{in} and w_{out} are plotted as well. Note that the two-step iterative algorithm provides a racing line that is qualitatively similar to the racing lines provided by the nonlinear gradient descent and human driver data. In particular, all three solutions succeed at effectively utilizing all of the available track width whenever possible and strike similar apex points on each of the circuit's 15 corners.

However, there are several locations on the track where there is a significant discrepancy on the order of several meters between the two-step algorithm's trajectory and the other comparison trajectories. These locations of interest are labeled (a)–(h) in Fig. 7. Note that sections (a), (e), (f), and (g) all occur on large, relatively straight portions of the racing circuit. In these straight

sections, the path curvature is relatively low and differences in lateral deviation from the track centerline have a relatively small effect on the lap time performance.

Of more significant interest are the sections labeled (b), (c), (d), and (h), which all occur at turning regions of the track. These regions are plotted in Figs. 9 and 10 for zoomed-in portions of the race track. While it is difficult to analyze a single turn of the track in isolation, discrepancies can arise between the two-step fast generation method and the gradient descent as the latter method trades off between a minimum curvature path and the path with shortest total distance. As a result, the gradient descent method finds regions where it may be beneficial to use less of the available road width in order to reduce the total distance traveled.

In region (b), for example, the fast generation algorithm exits the turn and gradually approaches the left side in order to create space for the upcoming right-handed corner. The nonlinear optimization, however, chooses a racing line that stays toward the right side of the track. In this case, the behavior of the human driver more closely matches that of the two-step fast generation algorithm. The human driver also drives closer to the fast generation solution in (h), while the gradient descent algorithm picks a path that exits the corner with a larger radius. In section (c), the gradient descent algorithm again prefers a shorter racing line that remains close the inside edge of the track, while the two-step algorithm drives all the way to the outside edge while making the right-handed turn. Interestingly, the human driver stays closer to the middle of the road, but more closely follows the behavior of the gradient descent algorithm. However, there are also regions of the track where the computational algorithms pick a similar path that differs from the human driver, such as region (d).

5.4 Lap Time Convergence and Predicted Lap Time. Figure 11 shows the predicted lap time for each iteration of the fast generation algorithm, with step 0 corresponding to the initial race track centerline trajectory. The lap time was estimated after each iteration by numerically simulating a vehicle following the desired path and velocity profile using a closed-loop controller. The equations of motion for the simulation were the nonlinear versions of Eq. (2) with tire forces given by the brush tire model in Eq. (8).

Figure 11 shows that the predicted lap time converges monotonically over four or five iterations, with significant improvements over the centerline trajectory occurring over the first two iterations. The predicted minimum lap time of 136.4 s is similar to the predicted lap time of 136.7 s from the nonlinear gradient descent approach, although in reality, the experimental lap time will depend significantly on unmodeled effects such as powertrain dynamics.
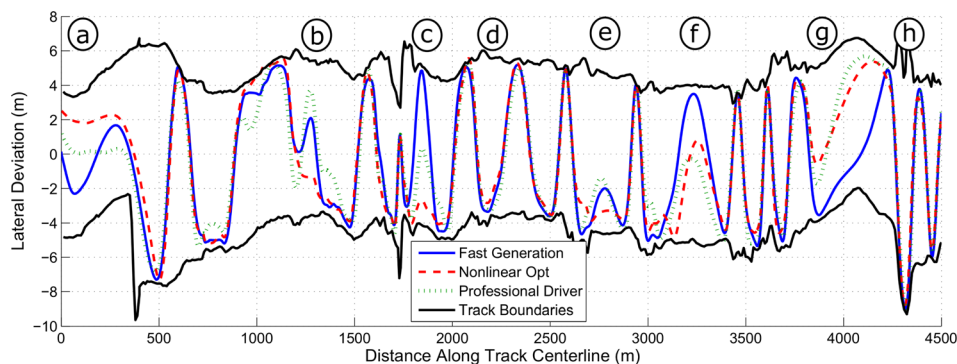

Fig. 8 Lateral path deviation of racing line from track centerline as a function of distance along the centerline. Note that upper and lower bounds on *e* are not always symmetric due to the initial centerline being a smooth approximation. Results are compared with racing line from a nonlinear gradient descent algorithm and experimental data recorded from a professional racecar driver.
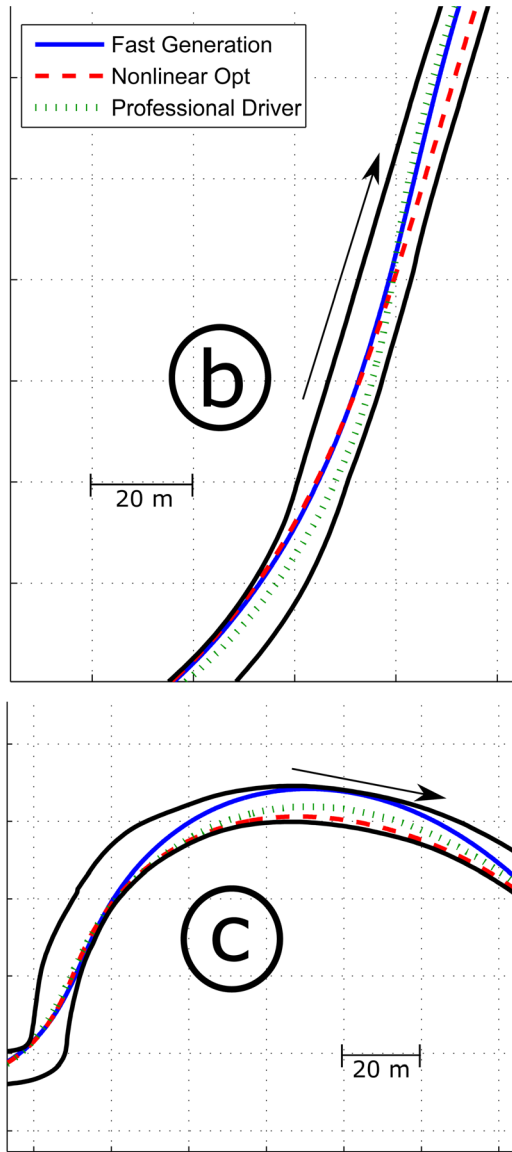
Fig. 9 Racing lines from the two-step fast generation approach, nonlinear gradient descent algorithm, and experimental data taken from professional driver. Car drives in direction of labeled arrow.
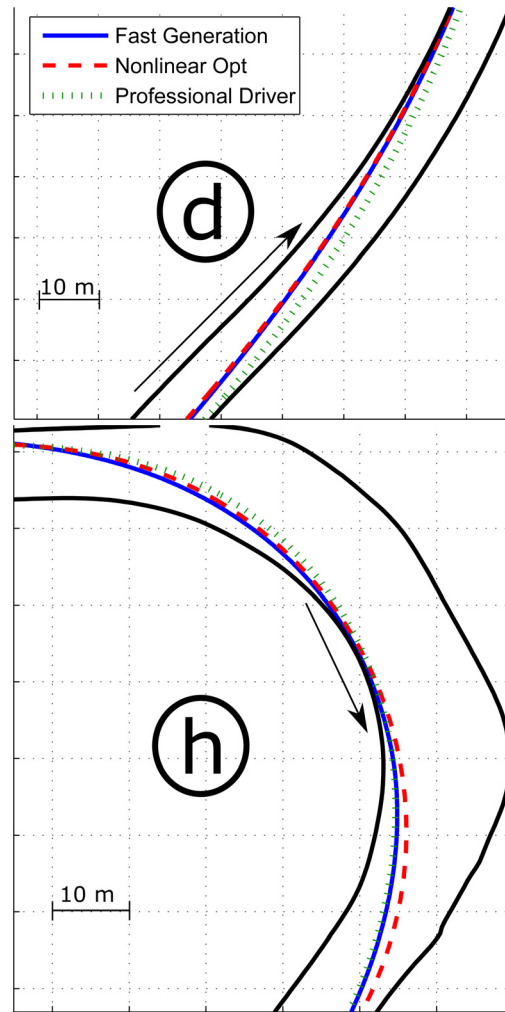

Fig. 10 Racing lines from the two-step fast generation approach, nonlinear gradient descent algorithm, and experimental data taken from professional driver. Car drives in direction of labeled arrow.

The final curvature and velocity profile for the two-step fast generation method are compared with the equivalent profiles for the gradient descent algorithm in Fig. 12. Notice that the piecewise linear nature of the nonlinear gradient descent method is due to the clothoid constraint imposed by Theodosis and Gerdes [5] for ease of autonomous path following. In general, the curvature and velocity profiles are very similar, although the fast generation algorithm results in a velocity profile with slightly lower cornering speeds but slightly higher top speeds. The predicted time difference between a car driving both trajectories is shown in Fig. 12(a), with a positive value corresponding the two-step algorithm being ahead. The trajectory from the two-step algorithm is predicted to outperform the gradient descent trajectory from (a)–(c), lose time from (c)–(e), and gain time from (e)–(h).

6 Experimental Validation

While the two-step algorithm works well in simulation, the most critical validation step is to have an autonomous race car drive the generated trajectory. This was accomplished by
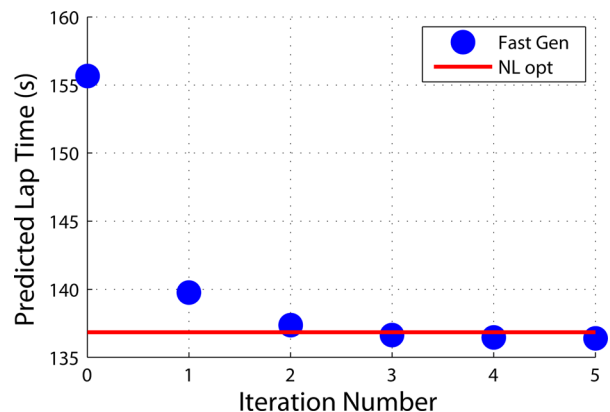

Fig. 11 Lap time as a function of iteration for the two-step fast trajectory generation method. Final lap time is comparable to that achieved with the nonlinear gradient descent approach. Iteration zero corresponds to the lap time for driving the track centerline.
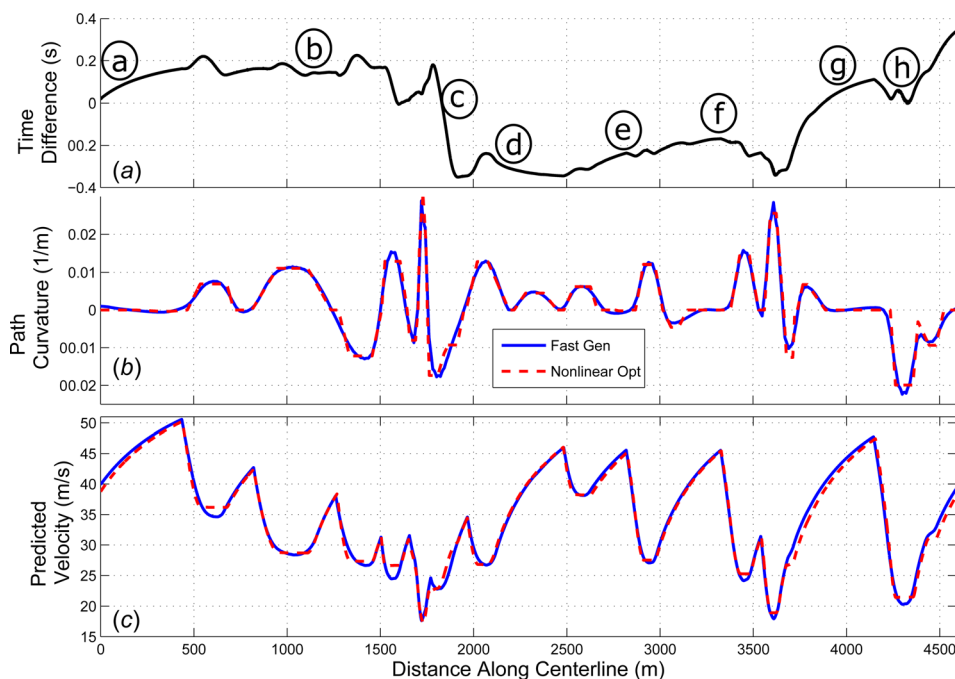
Fig. 12 (*a*) Predicted time difference between a car driving both trajectories, with a positive value corresponding the two-step algorithm being ahead. (*b*) Curvature profile $K(s)$ plotted versus distance along the path *s*. (*c*) Velocity profile $U_x(s)$ plotted versus distance along the path *s* for the two-step method and nonlinear gradient descent method.

collecting experimental data on "Shelley" (Fig. 6), an Audi TTS developed jointly by Stanford University and Audi's Electronics Research Laboratory (ERL). The TTS is equipped with an electronic power steering motor for autonomous steering and active brake booster and throttle by wire for longitudinal control.

Autonomous closed-loop following of the racing trajectory is accomplished by using an integrated differential GPS and inertial measurement unit to obtain the global vehicle position and velocity. A localization algorithm is applied to find the vehicle's position along the desired path (*s*) and lateral/heading deviation (*e* and $\Delta\Psi$) from the path. A previously developed feedback–feedforward steering algorithm [10] is then applied to keep the vehicle following the desired path at high lateral and longitudinal accelerations. By locating the desired position along the path, the current vehicle speed can be referenced to the desired vehicle speed from Fig. 12, and a simple proportional speed tracking controller with feedforward can be applied to track the desired speed profile. The same controller setup is also used to experimentally drive the trajectory generated by the nonlinear gradient descent algorithm. Closed-loop control is applied at a sample rate of 200 Hz using a dSPACE MicroAutobox unit. The experimental data for an autonomous lap of driving are shown in Fig. 13 for both the two-step trajectory and the trajectory from the gradient descent.

The resulting experimental lap time for the iterative two-step algorithm was 138.6 s, about 0.6 s faster than the experimental lap time for the gradient descent algorithm (139.2 s). For safety reasons, the trajectories were generated using a conservative peak road friction value of $\mu = 0.90$, resulting in peak lateral and longitudinal accelerations of 0.9 g. In reality, the true friction value of the road varies slightly, but is closer to $\mu = 0.95$ on average. As a result, both of these lap times are slightly slower than the fastest lap time recorded by a professional racecar driver (137.7 s) and the predicted lap times from Sec. 5. A summary of all lap times is provided in Table 2.

The experimental data in Fig. 13 generally match the simulated results in Fig. 12. The simulation predicted the trajectory from the iterative two-step algorithm would be 0.3 s quicker than that of

the nonlinear algorithm, compared to the 0.6 s speed advantage observed experimentally. The simulation also predicted a relative time advantage for the two-step algorithm from sections (a) to (c) and from (e) to (h), a trend seen in the experimental data as well. The two-step algorithm has relatively poor performance from section (c) to (d). This portion of the track corresponds to the sharp right-handed turn shown in Fig. 9, where the two-step solution differs significantly from the human driver data and the gradient descent solution. This turn also occurs on a steep downhill segment of the track, which was not accounted for by the fast generation algorithm. The experimental results indicate that the minimum curvature heuristic is relatively poor for this particular turn when compared to a nonlinear algorithm that explicitly minimizes the lap time. Accounting for three-dimensional topography effects in the curvature minimization or adding a term in the cost function to minimize distance traveled may improve the performance in the future.

Another reason for variation between the simulated and experimental time difference plots is variation in speed tracking. The speed tracking error for both racing lines is shown in Fig. 13(*c*). Interestingly, while the same speed tracking controller was used to test both racing lines, the controller has slightly better speed tracking performance when running the trajectory from the nonlinear optimization. This is possibly due to the longitudinal controller gains being originally tuned on the trajectory from Theodosis and Gerdes [5].

7 Discussion and Future Work

The primary benefit of the proposed algorithm is not improved lap time performance over the nonlinear algorithm but rather a radical improvement in computational simplicity and speed. Each two-step iteration of the full course takes only 26 s on an Intel i7 processor, whereas the nonlinear algorithm from Ref. [5] typically runs over the course of several hours on the same machine. The most significant computational expense for the proposed algorithm is solving the convex curvature minimization problem for all 1843 discrete time steps *T* over the 4.5 km racing circuit.
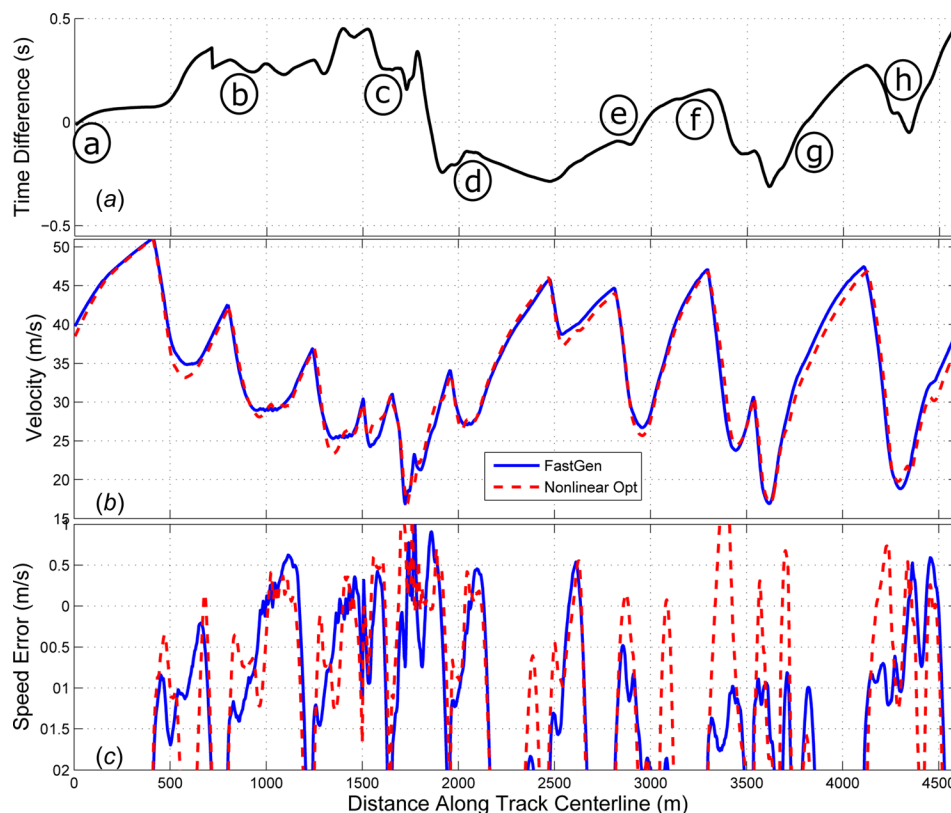
Fig. 13 Experimental data for an autonomous vehicle driving the trajectories provided by the two-step fast generation and gradient descent algorithms. (*a*) Relative time difference between vehicles driving both trajectories, with a positive time corresponding to the two-step algorithm being ahead. (*b*) Actual recorded velocity of vehicle. (*c*) Difference between actual and desired speed. Large negative values outside plotting range occur on straight sections of the track where the vehicle is limited by engine power and speed tracking error is poorly defined.

Table 2 Lap times in seconds

	Simulation	Experiment
Fast generation	136.4	138.6
Gradient descent	136.7	139.2
Human driver	N/A	137.7

Table 3 Iteration computation time

Lookahead (m)	T	Solve time (s)
450	184	5
900	369	6
1800	737	12
4500	1843	26

This computational efficiency will enable future work to incorporate the trajectory modification algorithm as an online "preview" path planner, which would provide the desired vehicle trajectory for an upcoming portion of the race track. Since the computation time of the algorithm is dependent on the preview distance, the high-level planner would not need to run at the same sample time as the vehicle controller. Instead, the planner would operate on a separate central processing unit and provide a velocity profile and racing line for only the next 1–2 km of the race track every few seconds, or plan a path for the next several hundred meters within a second.

Table 3 shows problem solve times for a varying range of lookahead lengths with the same discretization Δs and shows that the runtime scales roughly linearly with the lookahead distance. The above solve times are listed using the cvx convex optimization solver, which is designed for ease of use and is not optimized for embedded computing. Preliminary work has been successful in implementing the iterative two-step algorithm into C code using the CVXGEN software tool [15]. When written in optimized C code, the algorithm can solve the curvature minimization problem (15) in less than 0.005 s for a lookahead distance of 650 m.

The possibility of real-time trajectory planning for race vehicles creates several fascinating areas of future research. An automobile's surroundings are subject to both rapid and gradual changes over time, and adapting to unpredictable events requires a real-time trajectory planning algorithm. On a short time scale, the real-time trajectory planner could find a fast but stable recovery trajectory in the event of the race vehicle entering an understeer or oversteer situation. On an intermediate time scale, the fast executing two-step algorithm could continuously plan a racing line in the presence of other moving race vehicles by constraining the permissible driving areas to be collision-free convex "tubes" [16]. Finally, the algorithm could update the racing trajectory given estimates of the friction coefficient and other vehicle parameters learned gradually over several laps of racing.

8 Conclusion

This paper demonstrates an iterative algorithm for quickly generating vehicle racing trajectories, where each iteration is comprised of a sequential velocity update and path update step. Given an initial path through the race track, the velocity update step performs forward–backward integration to determine the minimum-

time speed inputs. Holding this speed profile constant, the path geometry is updated by solving a convex optimization problem to minimize path curvature. Experimental data confirm that the results generated by the algorithm for the Thunderhill Raceway circuit are comparable to those from a nonlinear gradient descent algorithm, with the primary advantage being a much faster computation time. An exciting opportunity for future research is incorporating the trajectory modification algorithm into an online path planner to provide racing trajectories in real-time.

Acknowledgment

References

[1] Hendrikx, J., Meijlink, T., and Kriens, R., 1996, "Application of Optimal Control Theory to Inverse Simulation of Car Handling," Veh. Syst. Dyn., **26**(6), pp. 449–461.

[2] Casanova, D., 2000, "On Minimum Time Vehicle Manoeuvring: The Theoretical Optimal Lap," Ph.D. thesis, Cranfield University, UK.

[3] Kelly, D. P., 2008, "Lap Time Simulation With Transient Vehicle and Tyre Dynamics," Ph.D. thesis, Cranfield University, UK.

[4] Perantoni, G., and Limebeer, D. J., 2014, "Optimal Control for a Formula One Car With Variable Parameters," Veh. Syst. Dyn., **52**(5), pp. 653–678.

[5] Theodosis, P. A., and Gerdes, J. C., 2011, "Generating a Racing Line for an Autonomous Racecar Using Professional Driving Techniques," ASME Paper No. DSCC2011-6097.

[6] Timings, J. P., and Cole, D. J., 2013, "Minimum Maneuver Time Calculation Using Convex Optimization," ASME J. Dyn. Syst. Meas. Control, **135**(3), p. 031015.

[7] Gerdts, M., Karrenberg, S., Müller-Beßler, B., and Stock, G., 2009, "Generating Locally Optimal Trajectories for an Automatically Driven Car," Optim. Eng., **10**(4), pp. 439–463.

[8] Lipp, T., and Boyd, S., 2014, "Minimum-Time Speed Optimisation Over a Fixed Path," Int. J. Control, **87**(6), pp. 1297–1311.

[9] Subosits, J. K., and Gerdes, J. C., 2015, "Autonomous Vehicle Control for Emergency Maneuvers: The Effect of Topography," American Control Conference (ACC), Chicago, IL, July 1–3, pp. 1405–1410.

[10] Kapania, N. R., and Gerdes, J. C., 2015, "Design of a Feedback-Feedforward Steering Controller for Accurate Path Tracking and Stability at the Limits of Handling," Veh. Syst. Dyn., **53**(12), pp. 1687–1704.

[11] Velenis, E., and Tsiotras, P., 2008, "Minimum-Time Travel for a Vehicle With Acceleration Limits: Theoretical Analysis and Receding-Horizon Implementation," J. Optim. Theory Appl., **138**(2), pp. 275–296.

[12] Sharp, R., 2014, "A Method for Predicting Minimum-Time Capability of a Motorcycle on a Racing Circuit," ASME J. Dyn. Syst. Meas. Control, **136**(4), p. 041007.

[13] Pacejka, H. B., 2012, Tire and Vehicle Dynamics, 3rd ed., Butterworth-Heinemann, Oxford, UK.

[14] Grant, M., and Boyd, S., 2014, "CVX: MATLAB Software for Disciplined Convex Programming," Version 2.1, http://cvxr.com/cvx

[15] Mattingley, J., and Boyd, S., 2012, "CVXGEN: A Code Generator for Embedded Convex Optimization," Optim. Eng., **13**(1), pp. 1–27.

[16] Erlien, S., Fujita, S., and Gerdes, J. C., 2013, "Safe Driving Envelopes for Shared Control of Ground Vehicles," Advances in Automotive Control Conference, Tokyo, Vol. 7, pp. 831–836.