Proceedings of the ASME 2015 Dynamic Systems and Control Conference
DSCC2015
October 28-30, 2015, Columbus, Ohio, USA

DSCC2015-9757

A SEQUENTIAL TWO-STEP ALGORITHM FOR FAST GENERATION OF VEHICLE RACING TRAJECTORIES

Nitin R. Kapania *
School of Mechanical Engineering
Stanford University
Stanford, CA 94305
nkapania@stanford.edu

John Subosits
School of Mechanical Engineering
Stanford University
Stanford, CA 94305
subosits@stanford.edu

J. Christian Gerdes
School of Mechanical Engineering
Stanford University
Stanford, CA 94305
gerdes@stanford.edu

ABSTRACT

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 speed profile fixed, 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 well when tested on the Thunderhill Raceway course in Willows, CA. The lap time reaches a minimum value after only three iterations, with each iteration over the full 5 km race course requiring only thirty seconds of computation time on a laptop computer. The resulting vehicle path and speed profile match very well with a nonlinear gradient descent solution and a path driven by a professional racecar driver, indicating that the proposed method is a viable option for online trajectory planning in the near future.

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 for 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 programming (NLP). Kelley [3] further extended the results from Casanova to consider transient vehicle dynamics and tire thermodynamics.

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 [4]. 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 [5] 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. Gerdts et al. [6] proposed a similar receding horizon approach, where distance along a reference

*Address all correspondence to this author.

path was maximized over a series of locally optimal optimization problems that were combined with continuity boundary conditions. One potential drawback of the model predictive control 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 920 milliseconds per 20 millisecond simulation step on a desktop PC.

In general, the aforementioned methods are still feasible for experimental implementation. As demonstrated in [7], 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. 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 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 iterative algorithm that generates vehicle racing trajectories with low computational expense. To speed up computation time, the combined lateral/longitudinal optimal control problem is replaced by two sequential sub-problems 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. The following section presents a mathematical framework for the overall 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. The next section describes the method of finding the minimum time speed inputs given a fixed path. While this sub-problem has been recently formulated as a convex optimization problem [8], a forward-backward integration scheme based on prior work [9] is used instead. Next, the paper 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 last section presents the final algorithm and compares results generated on the Thunderhill Raceway circuit in Willows, CA to results from both a nonlinear optimization and data recorded from a professional racecar driver. The paper concludes by discussing future implementation of the algorithm in a real-time path planner.

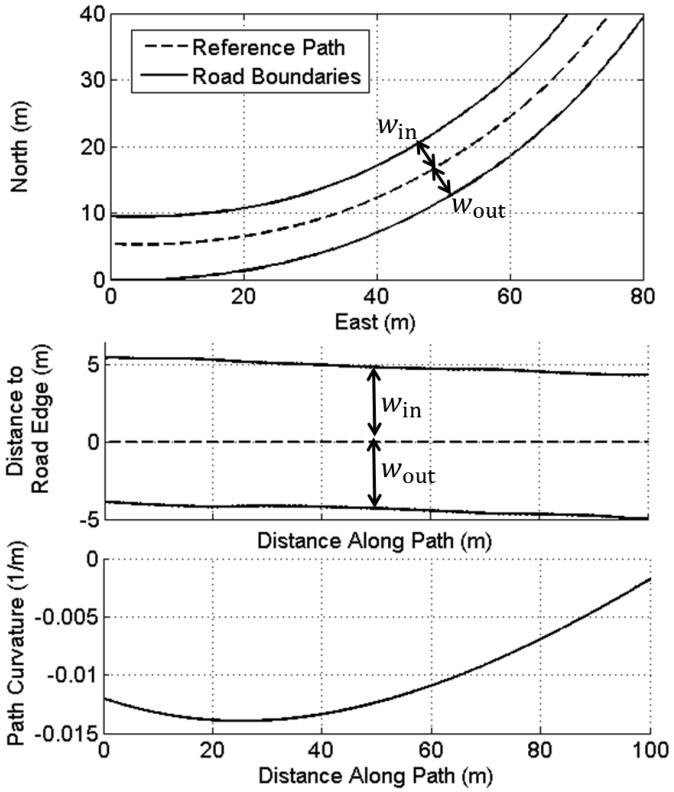

Figure 1. (a) View of a sample reference path and road boundaries, plotted in the East-North Cartesian frame (b) Lateral distance from path to inside road edge (positive) and outside road edge (negative) as a function of distance along path. (c) Curvature as a function of distance along path.

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. 1a 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. 1c). Additionally, it is convenient to store the road boundary information as two functions $w_{\text{in}}(s)$ and $w_{\text{out}}(s)$, which correspond to the lateral distance from the path at s to the inside and outside road boundaries, respectively (Fig. 1b). 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 are given by the Fresnel integrals:

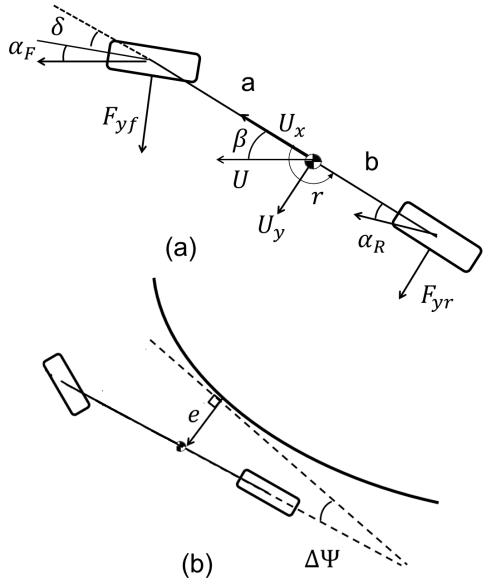

Figure 2. (a) Schematic of bicycle model. (b) Diagram showing lateral path deviation e and path heading error $\Delta\Psi$ states.

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

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

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

where $\Psi_{\mathrm{r}}(s)$ is the heading angle of the reference path. 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 (Fig. 2a), 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. 2b). Equations of motion for all four states are given by:

$$\dot{\beta} = \frac{F_{\mathrm{yf}} + F_{\mathrm{yr}}}{mU_{\mathrm{x}}(t)} - r \qquad \dot{r} = \frac{aF_{\mathrm{yf}} - bF_{\mathrm{yr}}}{I_z} \tag{2a}$$

$$\dot{e} = U_{\mathrm{x}}(\beta + \Delta\Psi) \qquad \Delta\dot{\Psi} = r - U_{\mathrm{x}}K \tag{2b}$$

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. 2a. Note that while the vehicle longitudinal dynamics are not explicitly modeled, the bicycle model does allow for time-varying values of U_{x}.

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 work from Velenis et al.. [10]. Given the lumped front and rear tires from Fig. 2a, the available longitudinal force F_{x} and lateral force F_{y} at each wheel is given by the friction circle constraint:

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

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

where μ is the tire-road friction coefficient and F_{z} is the available normal force at the tire. The method described in [9] determines the normal and lateral tire forces F_{z} and F_{y} at each point along the path by accounting for factors such as longitudinal weight transfer and three-dimensional topography effects (e.g. road bank and grade). However, for this paper, we will consider only the primary effects of road curvature, vehicle speed, and static weight distribution. The first pass of the speed profile generation aims to find the maximum permissible steady state vehicle speed given zero longitudinal force. This is given by:

$$U_{\mathrm{x}}(s) = \sqrt{\frac{\mu g}{|K(s)|}} \tag{4}$$

where the result in (4) is obtained by setting $F_{\mathrm{yf}} = \frac{mb}{a+b}U_{\mathrm{x}}^2 K$ and $F_{\mathrm{zf}} = \frac{mgb}{a+b}$. The results of this first pass for the sample curvature profile in Fig. 3a are shown in Fig. 3b. 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 $F_{x,\max}$ for acceleration. This available longitudinal force is calculated in [9] by accounting for the vehicle engine force limit and the lateral force demand on all tires due to the road curvature:

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

A key point of the forward integration step is that at every point, the value of $U_{\mathrm{x}}(s)$ is compared to the corresponding value from (4), and the minimum value is taken. The result is shown graphically in Fig. 3c. Finally, the backward integration step occurs, where the available longitudinal force for deceleration is con-

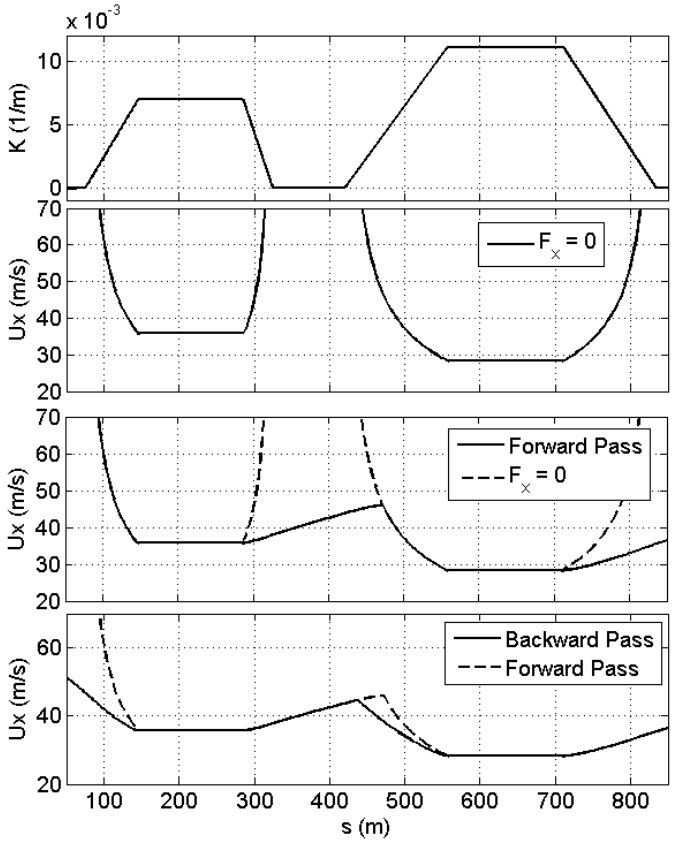

Figure 3. (a) Sample curvature profile. (b) Velocity profile given zero longitudinal force. (c) Velocity profile after forward pass (d) Final velocity profile after backward pass.

strained by the lateral force demand at both tires:

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

The value of $U_{\text{x}}(s)$ is then compared to the corresponding value from (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. 3d.

UPDATING PATH GIVEN FIXED VELOCITY PROFILE

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 [11] 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 minimize the norm of the vehicle curvature $K(s)$ at each path modification step. Intuitively, as the speed profile increases, the reference path will need to be modified to have lower peak curvature values in order to avoid saturating the lateral force capabilities of the tire. However, the minimum curvature path is not, in general, the minimum time path. In general, the minimum time path must tradeoff between the competing objectives of minimizing the path curvature to increase cornering speed, while also shortening the overall length of the path.

Convex Problem Formulation

Formulating the path update step as a convex optimization problem requires an affine, discretized form of the bicycle model presented earlier. The equations of motion in (2) are already linearized, but 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 [12] captures the force saturation of tires as a function of lateral tire slip angle α as follows:

$$F_{y\star} = \begin{cases} -C_\star \tan\alpha_\star + \frac{C_\star^2}{3\mu F_{z\star}} |\tan\alpha_\star| \tan\alpha_\star \\ \qquad -\frac{C_\star^3}{27\mu^2 F_{z\star}^2} \tan^3\alpha_\star, & |\alpha_\star| < \arctan\left(\frac{3\mu F_{z\star}}{C_\star}\right) \\ -\mu F_{z\star} \operatorname{sgn}\alpha_\star, & \text{otherwise} \end{cases} \tag{7}$$

where the symbol $\star \in [\text{f},\text{r}]$ denotes the lumped front or rear tire, and $C_\star$ 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:

$$\alpha_{\text{f}} = \beta + \frac{ar}{U_{\text{x}}} - \delta \tag{8a}$$

$$\alpha_{\text{r}} = \beta - \frac{br}{U_{\text{x}}} \tag{8b}$$

The brush tire model in (7) can be linearized at every point along the reference path assuming steady state cornering conditions:

$$F_{y\star} = \tilde{F}_{y\star} - \tilde{C}_\star(\alpha_\star - \tilde{\alpha}_\star) \tag{9a}$$

$$\tilde{F}_{y\star} = \frac{F_{z\star}}{g} U_{\text{x}}^2 K \tag{9b}$$

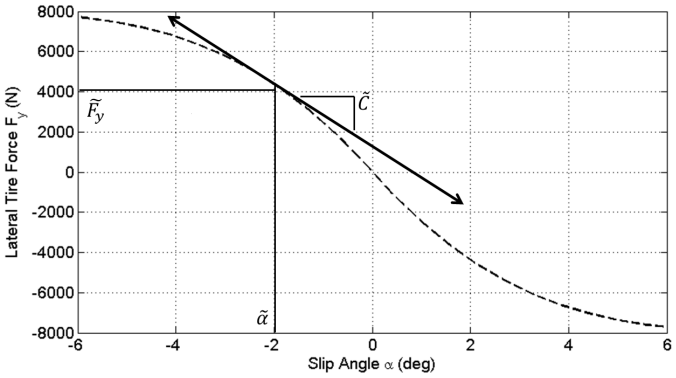

Figure 4. Nonlinear tire force curve given by Fiala model, along with affine tire model linearized at $\alpha = \tilde{\alpha}$.

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

$$\dot{x}(t) = A(t)x + B(t)\delta + d(t) \tag{10a}$$

$$x = [e \;\; \Delta\Psi \;\; r \;\; \beta \;\; \Psi]^T \tag{10b}$$

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 $d(t)$ are given by:

$$A(t) = \begin{bmatrix} 0 & U_{\mathrm{x}}(t) & 0 & U_{\mathrm{x}}(t) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-(a^2\tilde{C}_{\mathrm{f}}(t)+b^2\tilde{C}_{\mathrm{r}}(t))}{U_{\mathrm{x}}(t)I_{\mathrm{z}}} & \frac{b\tilde{C}_{\mathrm{r}}(t)-a\tilde{C}_{\mathrm{f}}(t)}{I_{\mathrm{z}}} & 0 \\ 0 & 0 & \frac{b\tilde{C}_{\mathrm{r}}(t)-a\tilde{C}_{\mathrm{f}}(t)}{mU_{\mathrm{x}}^2(t)} - 1 & \frac{-(\tilde{C}_{\mathrm{f}}(t)+\tilde{C}_{\mathrm{r}}(t))}{mU_{\mathrm{x}}(t)} & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{11}$$

$$B(t) = [0 \;\; 0 \;\; \frac{a\tilde{C}_{\mathrm{f}}(t)}{I_{\mathrm{z}}} \;\; \frac{\tilde{C}_{\mathrm{f}}(t)}{mU_{\mathrm{x}}(t)} \;\; 0]^T \tag{12}$$

$$d(t) = \begin{bmatrix} 0 \\ -K(t)U_{\mathrm{x}}(t) \\ \frac{a\tilde{C}_{\mathrm{f}}(t)\tilde{\alpha}_{\mathrm{f}}(t) - b\tilde{C}_{\mathrm{r}}(t)\tilde{\alpha}_{\mathrm{r}}(t) + a\tilde{F}_{\mathrm{yf}}(t) - b\tilde{F}_{\mathrm{yr}}(t)}{I_z} \\ \frac{\tilde{C}_{\mathrm{f}}(t)\tilde{\alpha}_{\mathrm{f}}(t) + \tilde{C}_{\mathrm{r}}(t)\tilde{\alpha}_{\mathrm{r}}(t) + \tilde{F}_{\mathrm{yf}}(t) + \tilde{F}_{\mathrm{yr}}(t)}{mU_{\mathrm{x}}(t)} \\ 0 \end{bmatrix} \tag{13}$$

The state matrices in (11) are functions of time, not distance along the path s. Time as a function of distance along the path $t(s)$ is obtained by computing the integral:

$$t(s) = \int_0^s \frac{dz}{U_{\mathrm{x}}(z)} \tag{14}$$

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

$$\text{minimize} \quad \sum_k \left(\frac{\Psi_k - \Psi_{k-1}}{s_k - s_{k-1}} \right)^2 \tag{15a}$$

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

$$w_k^{out} \le e_k \le w_k^{in} \tag{15c}$$

$$\left| \beta_k + \frac{a r_k}{U_{x,k}} - \delta_k \right| \le \alpha_{\mathrm{f,max}} \tag{15d}$$

$$\left| \beta_k - \frac{b r_k}{U_{x,k}} \right| \le \alpha_{\mathrm{r,max}} \tag{15e}$$

$$|\delta_k - \delta_{k-1}| \le \delta_{\mathrm{slew}} \tag{15f}$$

where $k = 1 \ldots T$ is the discretized time index, and A_k, B_k, and d_k are discretized versions of the continuous state-space equations in (10). 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 following section).

The equality constraint (15b) ensures 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. The inequality constraints (15d) and (15e) ensure that the vehicle does not exceed the peak force capability of the tires, and (15f) imposes a slew rate limit on the steering actuator. 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.

ALGORITHM IMPLEMENTATION AND RESULTS

Algorithm Implementation

The final algorithm for iteratively generating a vehicle racing trajectory is described in Fig. 6. 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

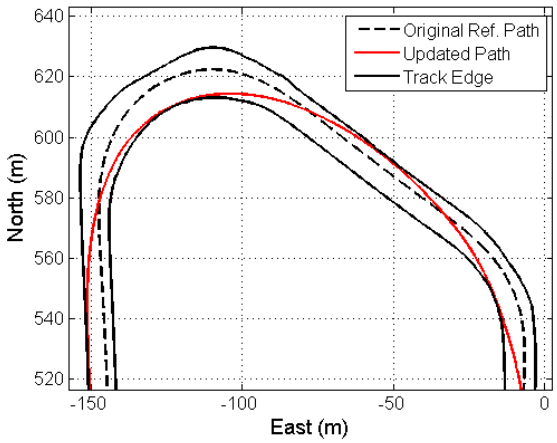

Figure 5. Path update for an example turn.

lane edge distances $w_{\text{in}}(s)$ and $w_{\text{out}}(s)$ 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).

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. (Note that $\star$ here refers to an optimal variable assignment, not to be confused with the notation in (7)). 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:

$$E_k \leftarrow E_k - e_k^\star \cos(\Psi_{\text{r},k}) \tag{16a}$$
$$N_k \leftarrow N_k - e_k^\star \sin(\Psi_{\text{r},k}) \tag{16b}$$

where Ψ_{r} is the path heading angle of the original path. Next, the new path is given by the following numerical approximation:

$$s_k = s_{k-1} + \sqrt{(E_k - E_{k-1})^2 + (N_k - N_{k-1})^2} \tag{17a}$$
$$K_k = \frac{\Psi_k^\star - \Psi_{k-1}^\star}{s_k - s_{k-1}} \tag{17b}$$

Notice that (17b) 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 change in lap time $\Delta t^\star$ from the prior iteration is greater than a small positive constant ε. Note that a positive value of $\Delta t^\star$ indicates the current iteration yields a slower lap time compared to the prior iteration. Recall that this is possible given that we have no guarantees of convergence, and that the minimum curvature solution is not necessarily the minimum time solution.

1: **procedure** GENERATETRAJECTORY($s^0, K^0, w_{\text{in}}^0, w_{\text{out}}^0$)
2: path $\leftarrow (s^0, K^0, w_{\text{in}}^0, w_{\text{out}}^0)$
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**

Figure 6. Iterative algorithm for fast generation of vehicle trajectories. Each iteration consists of a sequential two-step approach where the velocity profile is generated given a fixed path and then the path is updated based on the solution from a convex optimization problem.

Algorithm Validation

The proposed algorithm is tested on the 5 km Thunderhill racing circuit in Willows, California, USA. The vehicle parameters used for the lap time optimization come from an Audi TTS experimental race vehicle, with relevant parameters shown in Table 1. The initial path is obtained by collecting 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.

The algorithm is implemented in MATLAB, with the minimum curvature optimization problem (15) solved using the CVX software package [13]. For the purpose of simplicity, topography effects such as bank and grade were neglected.

Comparison with Other Methods

The generated racing path after four iterations is shown in Fig. 7. To validate the proposed algorithm, the racing line is compared with results from a nonlinear gradient descent algorithm implemented by Theodosis and Gerdes [4] and an experimental trajectory recorded from a professional racecar driver. While time-intensive to compute, the gradient descent approach generates racing lines with autonomously driven lap times within one second of lap times measured from professional racecar drivers.

In general, the two-step method provides a racing line that is very similar to the line from the gradient descent method and the human experimental data. However, several interesting discrepancies are plotted in Fig. 8 and Fig. 9 for zoomed-in portions of the race track. The discrepancies arise because the nonlinear method balances the tradeoff between a minimum curvature path

Table 1. Optimization Parameters

Parameter	Symbol	Value	Units
Vehicle mass	m	1500	kg
Yaw Inertia	I_z	2250	$\mathrm{kg}\cdot\mathrm{m}^2$
Front axle to CG	a	1.04	m
Rear axle to CG	b	1.42	m
Front cornering stiffness	C_F	160	$\mathrm{kN}\cdot\mathrm{rad}^{-1}$
Rear cornering stiffness	C_R	180	$\mathrm{kN}\cdot\mathrm{rad}^{-1}$
Friction Coefficient	μ	0.95	–
Path Discretization	Δs	2.75	m
Optimization Time Steps	T	1843	-
Max Engine Force	-	3750	N

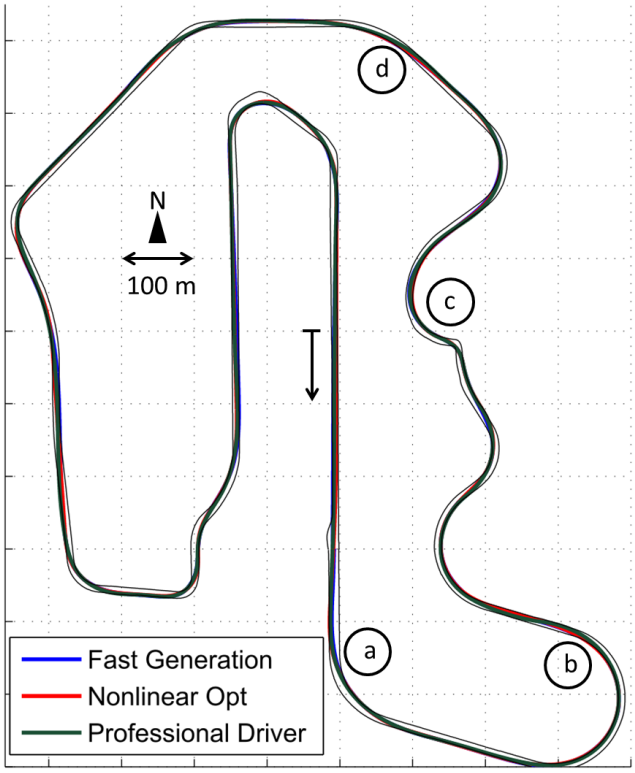

Figure 7. Overhead view of Thunderhill Raceway park along with generated path from algorithm. Car drives in indicated direction around the closed circuit. Labeled regions a-d will be discussed in more detail.

and the path with shortest total distance, and finds regions where it may be beneficial to use less of the available road width or drive a higher curvature path in order to reduce the total distance traveled. However, like most nonlinear racing line optimization techniques, the method in [4] is only guaranteed to converge to

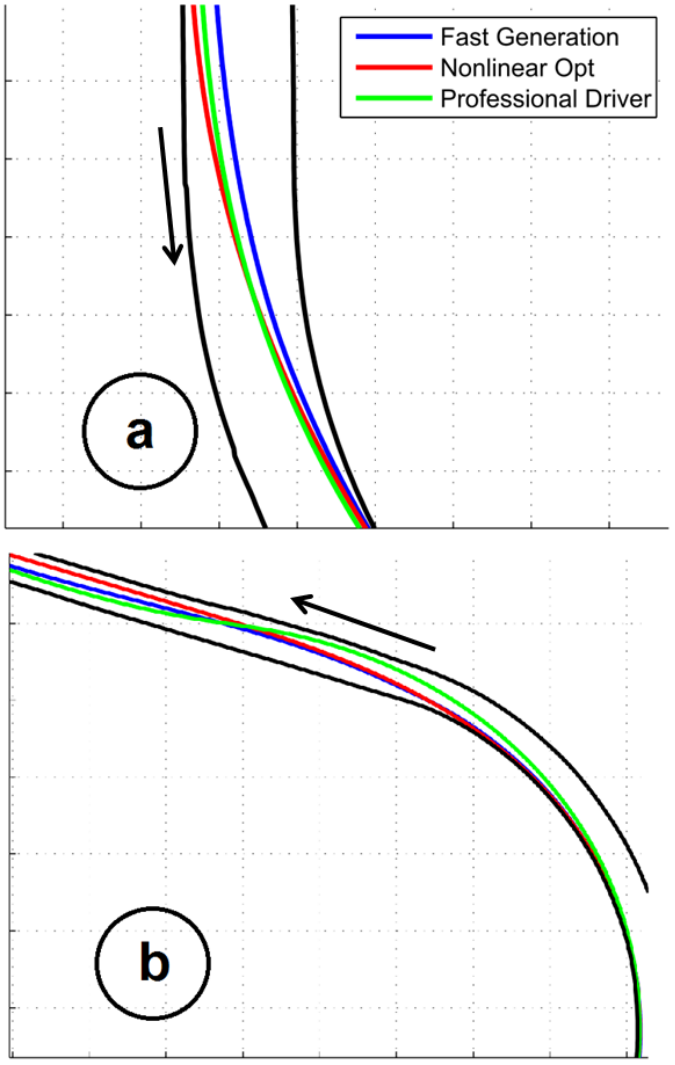

Figure 8. Racing lines from the two-step fast generation approach, nonlinear gradient descent algorithm, and experimental data taken from professional driver. Arrows indicate direction of path. Labeled regions a-b correspond to zoomed-in locations on Fig. 7.

a locally minimal solution. Interestingly, in both (b) and (c), the two-step method provides a closer match to the professional driver, while the gradient descent solution matches the professional driver better in (a). For the case of (d), both numerical methods provide similar results, but this result is different from the path chosen by the human driver.

To further compare the path generation methods, Fig. 10 shows the path curvature profile $K(s)$ for both algorithms. Again, the curvature profiles look similar, with small discrepancies around $s = 900$m, $s = 1100$m and $s = 1400$m. The piecewise linear nature of the nonlinear gradient descent method is due to the clothoid constraint imposed by [4]. The resulting velocity

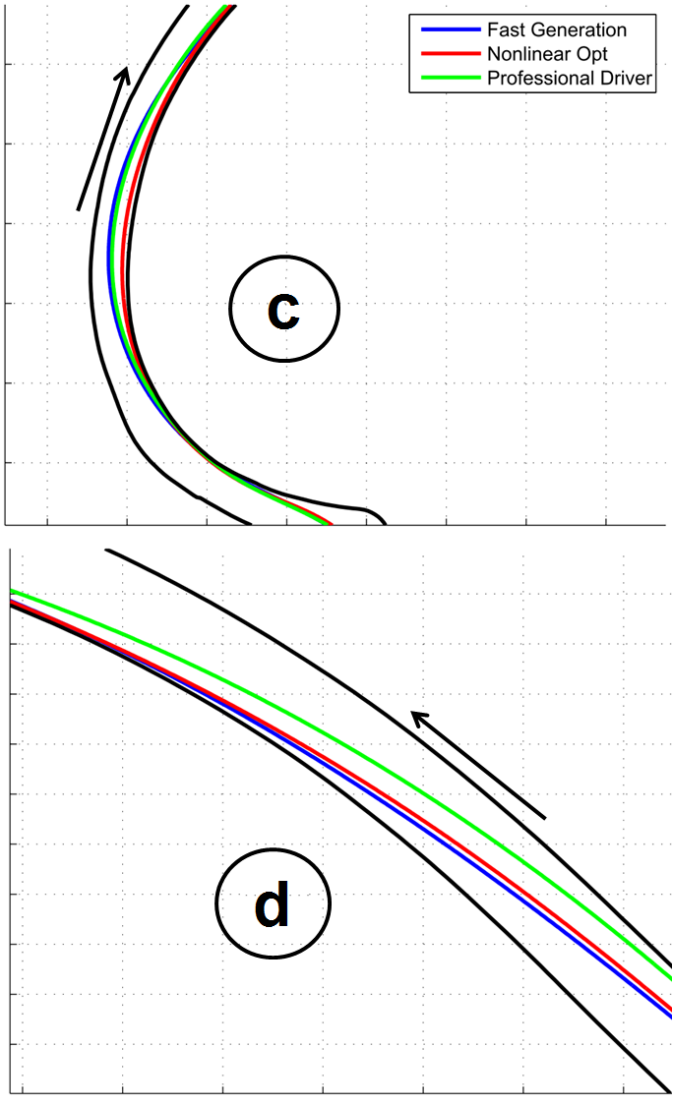

Figure 9. Racing lines from the two-step fast generation approach, nonlinear gradient descent algorithm, and experimental data taken from professional driver. Arrows indicate direction of path. Labeled regions c-d correspond to zoomed-in locations on Fig. 7.

profiles are shown in Fig. 11 for the two-step method and gradient descent method. Interestingly, the gradient descent method is able to achieve higher cornering speeds, although the two-step method is able to gain a speed advantage by having the vehicle accelerate earlier after a turn is finished.

Lap Time Convergence and Computation Time

Fig. 12 shows the predicted lap time for each iteration of the two-step algorithm, with step 0 corresponding to the race track centerline. While the lap time can be estimated by (14),

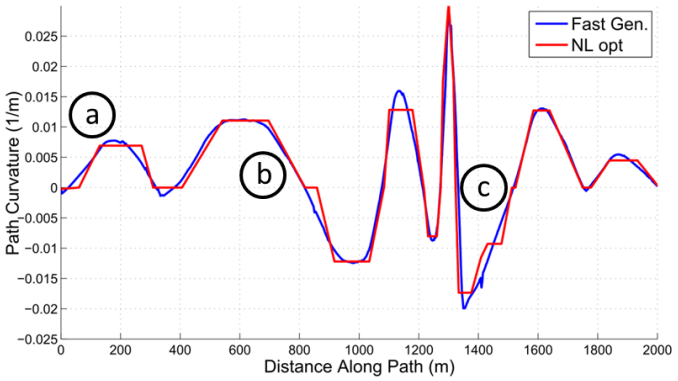

Figure 10. Curvature profile $K(s)$ plotted vs. distance along the path s. For improved visibility, results are shown for the first 2 km of the track.

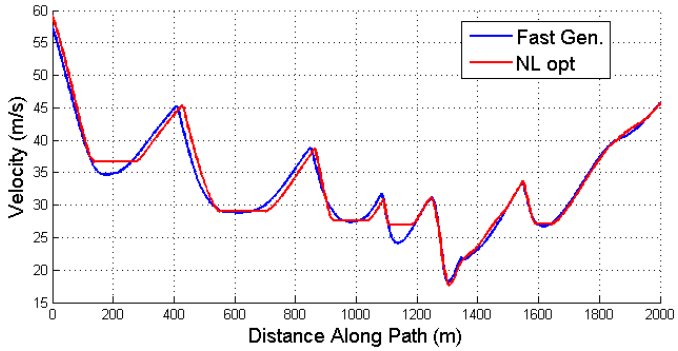

Figure 11. Velocity profile $U_x(s)$ plotted vs. distance along the path s for the two-step method and nonlinear gradient descent method.

we obtained a more precise lap time by numerically simulating a vehicle following the desired path and velocity profile using the feedback-feedforward control formulation presented in [7]. The equations of motion are the nonlinear versions of (2) with tire forces given by the brush tire model in (7). The benefit of using the more precise lap time simulation is accounting for the path tracking dynamics of the vehicle and ensuring the car actually stays on the track at all times when subjected to a more accurate vehicle dynamics model. After three iterations, the predicted lap time achieves a minimum value of 135.0 seconds. The predicted lap time on the fourth iteration degrades slightly to 135.3 seconds, at which the algorithm terminates. The minimum lap time is similar to the predicted lap time of 135.5 seconds from the nonlinear gradient descent approach, although in reality, the actual experimental lap time achieved will depend significantly on factors such as three-dimensional path topography and engine dynamics.

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

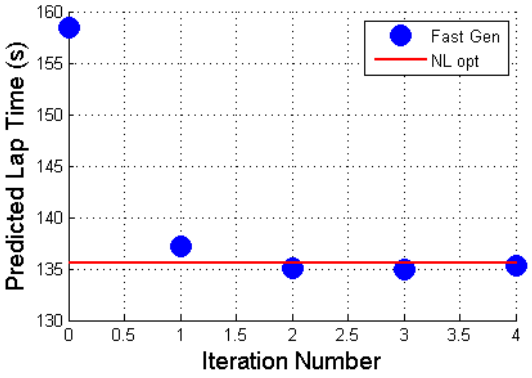

Figure 12. Lap time as a function of iteration for the two-step fast trajectory generation method. Final lap time is comparable to that achieved with the nonlinear gradient descent approach. Iteration zero corresponds to the lap time for driving the initial path.

speed. Each two-step iteration of the full course takes only 26 seconds on an Intel i7 processor, whereas the nonlinear algorithm from [4] 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 of the race vehicle over the 5 km racing circuit. 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 heavily 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 CPU and provide a velocity profile and racing line for only the next 1-2 kilometers of the race track every 5-10 seconds, while incorporating estimates of the friction coefficient and other vehicle parameters learned over several laps of racing.

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. The trajectory generated by the algorithm for the Thunderhill Raceway circuit is comparable to a trajectory from a previously published nonlinear gradient descent algorithm and an experimental trajectory collected from a professional driver. Future work will apply the racing trajectory on a vehicle testbed to determine the experimental performance of the algorithm. Finally, an exciting opportunity for future research is incorporating the trajectory modification algorithm into an online path planner to provide preview trajectories in real time.

ACKNOWLEDGMENT

The authors would like to thank Marcial Hernandez for assistance with fitting an initial curvature profile from GPS point cloud data, as well as Vincent Laurense, John Kegelman and Ohi Dibua of the Dynamic Design Laboratory. Kapania and Subosits are both supported by Stanford Graduate Fellowships.

REFERENCES

[1] Hendrikx, J., Meijlink, T., and Kriens, R., 1996. "Application of optimal control theory to inverse simulation of car handling". *Vehicle System Dynamics,* **26**(6), pp. 449–461.

[2] Casanova, D., 2000. "On minimum time vehicle manoeuvring: The theoretical optimal lap".

[3] Kelly, D. P., 2008. "Lap time simulation with transient vehicle and tyre dynamics".

[4] Theodosis, P. A., and Gerdes, J. C., 2011. "Generating a racing line for an autonomous racecar using professional driving techniques". In Dynamic Systems and Control Conference, pp. 853–860.

[5] Timings, J. P., and Cole, D. J., 2013. "Minimum maneuver time calculation using convex optimization". *Journal of Dynamic Systems, Measurement, and Control,* **135**(3), p. 031015.

[6] Gerdts, M., Karrenberg, S., Müller-Beßler, B., and Stock, G., 2009. "Generating locally optimal trajectories for an automatically driven car". *Optimization and Engineering,* **10**(4), pp. 439–463.

[7] Kritayakirana, K., and Gerdes, J. C., 2012. "Autonomous vehicle control at the limits of handling". *International Journal of Vehicle Autonomous Systems,* **10**(4), pp. 271–296.

[8] Lipp, T., and Boyd, S., 2014. "Minimum-time speed optimisation over a fixed path". *International Journal of 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". In American Control Conference.

[10] Velenis, E., and Tsiotras, P., 2008. "Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation". *Journal of Optimization Theory and Applications,* **138**(2), pp. 275–296.

[11] Sharp, R., 2014. "A method for predicting minimum-time capability of a motorcycle on a racing circuit". *Journal*

of Dynamic Systems, Measurement, and Control, **136**(4), p. 041007.
[12] Pacejka, H. B., 2012. *Tire and Vehicle Dynamics*, 3rd ed. Butterworth-Heinemann.
[13] Grant, M., and Boyd, S., 2014. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, Mar.