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

**Nitin R. Kapania** *
School of Mechanical Engineering
Stanford University
Stanford, CA 94065
nkapania@stanford.edu

**John Subosits**
School of Mechanical Engineering
Stanford University
Stanford, CA 94065
subosits@stanford.edu

**J. Christian Gerdes**
School of Mechanical Engineering
Stanford University
Stanford, CA 94065
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) as well as lateral (steering wheel) control inputs. Unfortunately, finding combined longitudinal and lateral control inputs to minimize lap time typically requires solving a nonlinear, nonconvex optimization problem, which can be computationally expensive. 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 vehicle path through the race track, the algorithm runs a forwards-backwards integration scheme to determine an aggressive minimum-time longitudinal speed profile, subject to tire friction constraints. Given this speed profile, the algorithm updates the vehicle's path by solving a convex optimization problem that minimizes the curvature of the updated path while staying within track boundaries and obeying affine, time-varying vehicle dynamics constraints at each time step. This two-step process is repeated iteratively until the predicted lap time fails to improve. While the algorithm does not explicitly minimize lap time and provides no guarantees of convergence or a globally optimal solution, the approach works well in practice when tested on the Thunderhill Raceway course in Willows, CA. A minimum lap time is met after only three iterations, and the resulting vehicle path and speed profile matches very well with a nonlinear gradient descent solution as well as paths observed by a professional race car driver.*

*Address all correspondence to this author.

## INTRODUCTION

The problem of calculating the minimum lap-time path and velocity profile for a given vehicle and race track has been well 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 optimal 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 non-linear programming (NLP). This work was further extended by Kelley [3] to consider advanced effects such as 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 optimal racing lines, with the racing line constrained to be composed of a fixed number of clothoid segments [4]. While the path itself was optimized offline, feedforward steering and throttle/brake inputs were calculated real-time in an autonomous race vehicle [5]. Given the relative 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 min-

imum time objective by imizing distance traveled along the path centerline. Gerdts et al [7] proposed a similar receding horizon approach, where 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 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.

s paper considers an alternative, iterative approach to simplifying the trajectory generation process. Each iteration consists of a two-step process where the minimum-time longitudinal speed commands are solved given a fixed vehicle path, and then the vehicle path is updated given the more aggressive longitudinal speed profile. An estimated lap time is computed after every iteration until the predicted lap time fails to improve. The fully nonlinear optimal control problem where longitudinal and lateral inputs are solved in parallel is therefore replaced by two simpler sub-problems where longitudinal and lateral inputs are solved sequentially. This sequential approach was described by Sharp [8] as a method of systematically learning faster motorcycle trajectories through a racing circuit. At each iteration, the authors incremented the speed inputs by a fixed amount and determined if the motorcycle could drive the course without crashing. If a crash was detected, the speed inputs were locally reduced around the crash site and the process was repeated. The resulting algorithm was able to reduce the lap time of the Catalunya race circuit by six seconds over around twenty iterations.

The paper is divided as follows. Section II presents a mathematical framework for the overall trajectory generation problem and provides a linearized five-state model for the planar dynamics of a race car following a set of speed and steering inputs, with lateral and heading error states computed with respect to a fixed path. Section III 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 [9], a forward-backwards integration scheme based on prior work [10] is used instead. Section IV 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 actively minimized. Section V presents the final algorithm and compares results generated on the Thunderhill Raceway circuit in Willows, CA to results from both a full nonlinear optimization as well as a professional race car driver.

## 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

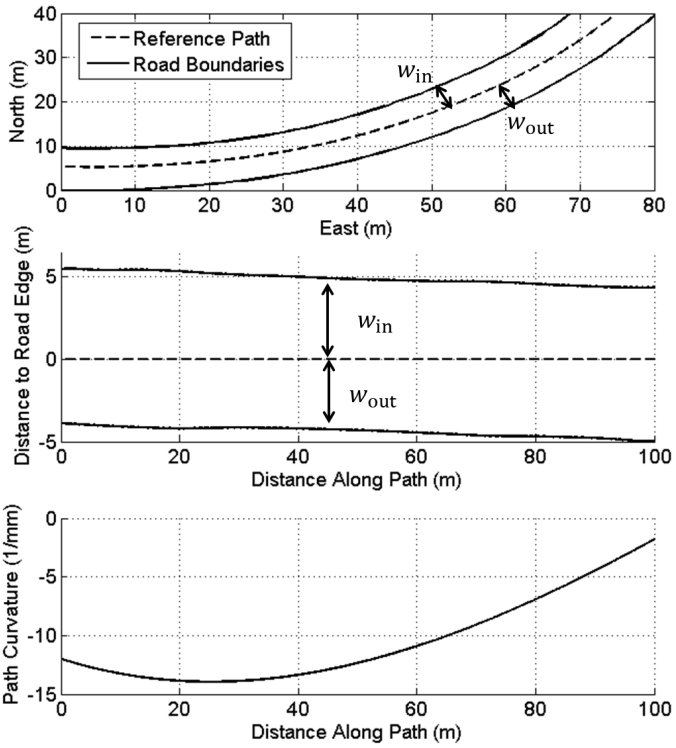


Figure 1. (a) View of a sample reference path and road boundaries, plotted in the East-North Cartesian frame (b) Lateral distance from path to inside road edge (positive) and outside road edge (negative) as a function of path length. (c) Curvature as a function of path length.

coordinates, with road boundaries represented by similar Cartesian curves as well. 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 in Section 4. The transformation between the local $s$, $K$ coordinate frame and the global Cartesian coordinates $E$, $N$ are given by the Fresnel integrals:

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

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

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

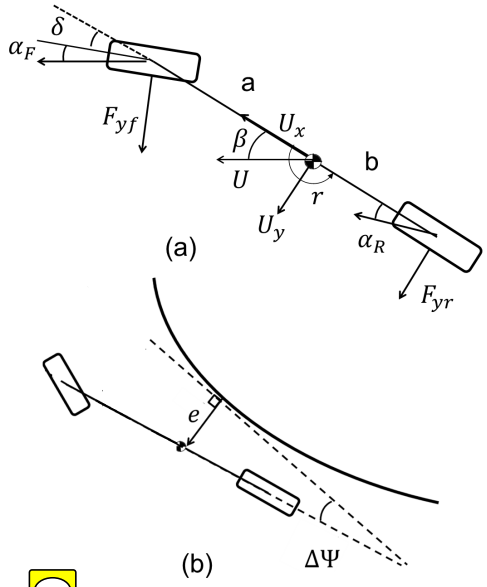


Figure 2. (a) Schematic of bicycle model. (b) Diagram showing lateral path deviation $e$ and path heading error $\Delta\Psi$ states.

where $\Psi_r(s)$ 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 $\beta$ 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:

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

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

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 $U_x$. This allows the path modification step in Section 4 to account for the speed inputs provided by the longitudinal planner in Section 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 [10], and originally inspired by work from Velenis et al. [11]. 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:

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

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

Where $\mu$ is the tire-road friction coefficient and $F_z$ the available normal force at the tire. The method described in [10] 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 such as surface bank and grade. For the purposes of describing the algorithm here, we will consider only the primary effects of road curvature, vehicle speed, and static weight distribution. The first pass of the speed profile generation is to find the maximum permissible steady state vehicle speed given zero longitudinal force. This is given by:

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

Where the result in (4) is obtained by setting $F_{yf} = \frac{mb}{a+b}U_x^2 K$ and $F_{zf} = \frac{mgb}{a+b}$. The results of this first pass for the sample curvature profile in Fig. 3a are shown in Fig. 3b. The next step is a forwards 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, after accounting for the vehicle engine limit as well as the lateral force demand on both front and rear tires due to the road curvature:

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

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

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

The value of $U_x(s)$ is then compared to the equivalent 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

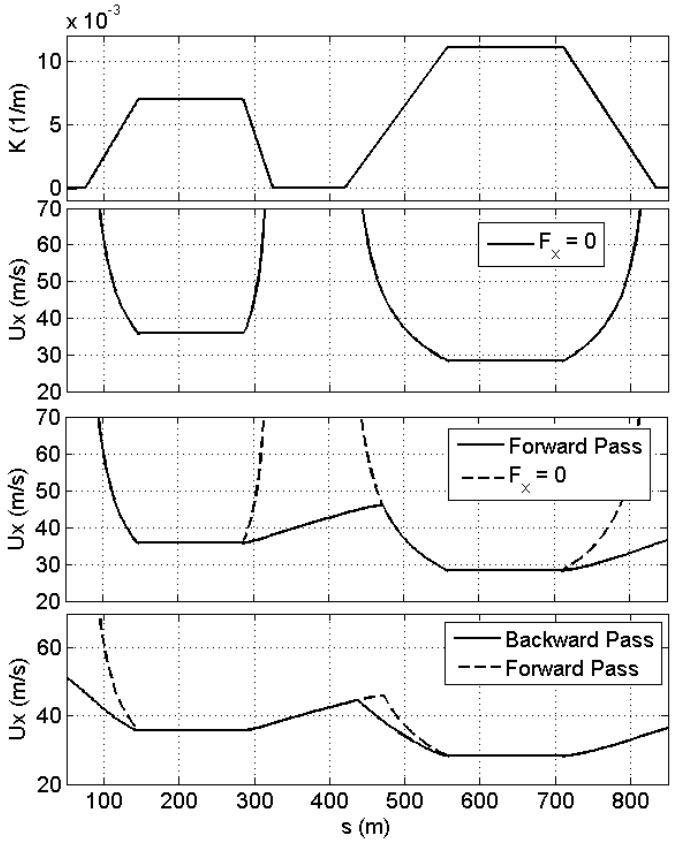


Figure 3. (a) Sample curvature profile. (b) Velocity profile given zero longitudinal force. (c) Velocity profile after forward pass (d) Final velocity profile after backwards pass.

in Fig. 3d. Given a reasonable path discretization step, the forward and backwards passes can be performed relatively quickly.

## 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. A general approach for accomplishing the lap-to-lap time speedup was presented for a motorcycle by Sharp [8], who suggested taking a feasible reference 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.

However, one challenge with this approach is that it can take 20-50 iterations of sequentially incrementing the vehicle speed profile and modifying the reference path to converge to a fast lap time. An alternative approach we propose is to minimize the norm of the vehicle curvature $K(s)$ at each path modification step. Intuitively, as the speed profile becomes more aggressive, 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 trade off 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 in Section 2. The equations of motion in (2) are already linearized, but unfortunately, the front and rear lateral tire forces become nonlinear functions of the vehicle states $\beta$ and $r$ as the vehicle drives near the limits of tire adhesion. The well-known brush Fiala model [12] captures the force saturation of tires as a function of lateral tire slip angle $\alpha$ as follows:

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

Where the symbol $\star \in [\mathrm{f},\mathrm{r}]$ denotes the lumped front or rear tire, and $C_\star$ the corresponding tire stiffness. The linearized tire slip angles $\alpha_\mathrm{f}$ and $\alpha_\mathrm{r}$ are functions of the vehicle lateral states and the steer angle input, $\delta$.

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

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

The Fiala tire model in (7) can be linearized at every point along the reference path given the velocity profile $U_x(s)$ and curvature profile $K(s)$:

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

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

with parameters $\tilde{F}_\mathrm{y}$, $\tilde{\alpha}$ and $\tilde{C}$ shown in Fig. 4. Note that the linearization point in 9b assumes steady state cornering conditions.

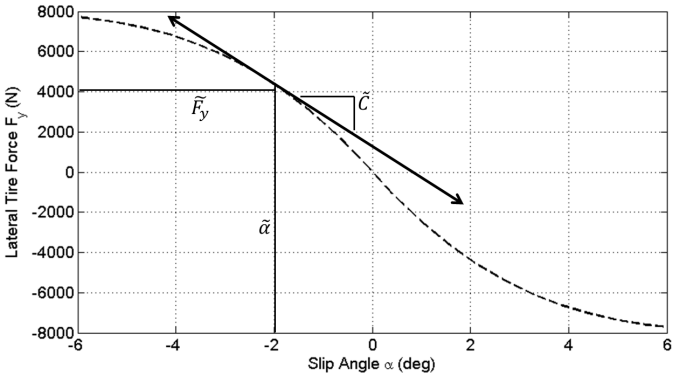


Figure 4. Nonlinear tire force curve given by Fiala model, along with affine tire model linearized at $\alpha = \tilde{\alpha}$.

The affine, continuous bicycle model with steering input $\delta$ is then written in state-space form as:

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

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

$$A(t) = \begin{bmatrix} 0 & U_x(t) & 0 & U_x(t) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-(a^2\tilde{C}_f(t)+b^2\tilde{C}_r(t))}{U_x(t)I_z} & \frac{b\tilde{C}_r(t)-a\tilde{C}_f(t)}{I_z} & 0 \\ 0 & 0 & \frac{b\tilde{C}_r(t)-a\tilde{C}_f(t)}{mU_x^2(t)} - 1 & \frac{-(\tilde{C}_f(t)+\tilde{C}_r(t))}{mU_x(t)} & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{11}$$

$$B(t) = [0 \;\; 0 \;\; \frac{a\tilde{C}_f(t)}{I_z} \;\; \frac{\tilde{C}_f(t)}{mU_x(t)} \;\; 0]^T \tag{12}$$

$$d(t) = \begin{bmatrix} 0 \\ -K(t)U_x(t) \\ \frac{a\tilde{C}_f(t)\tilde{\alpha}_f(t) - b\tilde{C}_r(t)\tilde{\alpha}_r(t) + a\tilde{F}_{yf}(t) - b\tilde{F}_{yr}(t)}{I_z} \\ \frac{\tilde{C}_f(t)\tilde{\alpha}_f(t) + \tilde{C}_r(t)\tilde{\alpha}_r(t) + \tilde{F}_{yf}(t) + \tilde{F}_{yr}(t)}{mU_x(t)} \\ 0 \end{bmatrix} \tag{13}$$

Note that the state matrices 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:

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

Also note that we have added a fifth state, vehicle heading angle $\Psi$, defined as the time integral of yaw rate $r$. This will make explicit computation of the minimum curvature path simpler. With the nonlinear model now approximated as an affine, time-varying model, updating the path is accomplished by solving the following convex optimization problem:

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

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

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

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

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

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

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 spatial derivative of the vehicle heading angle (1c). The term $s_k - s_{k-1}$ is assumed to be constant constant and is taken from the original path. If the path length $s_k$ was treated as an optimization variable rather than a constant, the minimum curvature vehicle path could be obtained exactly after one iteration. However, this would make the objective function non-convex.

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 a hairpin turn in Fig. 5. The reference path starts out at the road centerline, and the optimization finds a modified path that uses up 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 $K(s)$, as well as the lane edge distances $w_{in}(s)$ and $w_{out}(s)$ as described in Section 2. Given the initial path, the minimum time speed profile $U_x(s)$ is calculated as described in Section 3. Next, given the initial path and speed profile, the path is modified by solving the minimum

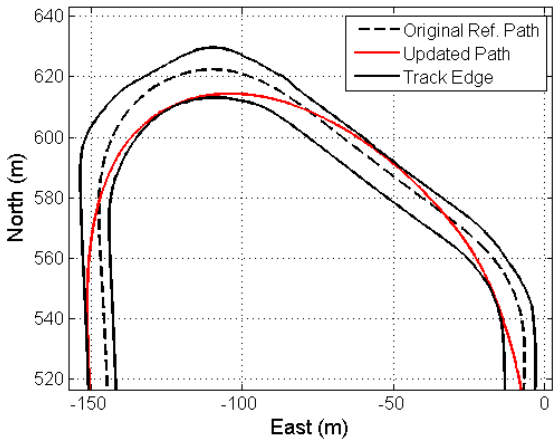


Figure 5. Path update for a hairpin turn.

curvature convex optimization problem (15) described in Section 4.

An important note is that the optimization only solves explicitly for the steering input $\delta^\star$ and resulting vehicle lateral states $x^\star$ at every time step, most importantly 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 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:

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

Where $\Psi_r$ 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 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 $w_{out}$ will be different for the new path as well. These are updated as follows:

$$w_{in} \leftarrow w_{in} - e^\star \tag{18a}$$
$$w_{out} \leftarrow w_{out} + e^\star \tag{18b}$$

1: **procedure** GENERATETRAJECTORY($s^0, K^0, w_{in}^0, w_{out}^0$)
2: path $\leftarrow (s^0, K^0, w_{in}^0, w_{out}^0)$
3: **while** $\Delta t^\star < \varepsilon$ **do**
4: $U_x \leftarrow$ calculateSpeedPr(path)
5: $t^\star \leftarrow$ calculateLapTime(path, $U_x$)
6: path $\leftarrow$ minimizeCurvature($U_x$, 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.

Note that (18) simply computes new lane boundaries by offsetting the original track boundaries by the optimal lateral deviation. A more accurate, but slightly more computationally intensive alternative is to compute new values of $w_{in}(s)$ and $w_{out}(s)$ directly using the Cartesian coordinates for the inner and outer track edges. The two-step procedure is iterated until the predicted lap time $t^\star$ ceases to improve by a desired amount.

## 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 $(s, K, w_{in}, w_{out})$ 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. The generated racing path after four iterations is shown in Fig. 7.

## Comparison with Other Methods

To validate the proposed algorithm, the resulting trajectory is compared with results from a nonlinear gradient descent algorithm implemented by Theodosis and Gerdes [4], as well as an experimental trajectory taken by a professional race car driver. While time-intensive to compute, the nonlinear descent approach has been shown experimentally to provide racing lines that perform very well when compared to experienced human drivers. The vehicle paths for both methods are shown in Fig. 9 for several zoomed-in portions of the race track. The two-step method compares very well with the gradient descent method and the human experimental data, although there are several interesting

Table 1. Optimization Parameters

| Parameter | Symbol | Value | Units |
|---|---|---|---|
| Vehicle mass | $m$ | 1500 | kg |
| Yaw Inertia | $I_z$ | 2250 | $\mathrm{kg \cdot m^2}$ |
| Front axle to CG | $a$ | 1.04 | m |
| Rear axle to CG | $b$ | 1.42 | m |
| Front cornering stiffness | $C_F$ | 160 | $\mathrm{kN \cdot rad^{-1}}$ |
| Rear cornering stiffness | $C_R$ | 180 | $\mathrm{kN \cdot rad^{-1}}$ |
| Friction Coefficient | $\mu$ | .95 | – |
| Path Discretization | $\Delta s$ | 2.75 | $m$ |
| Optimization Time Steps | $T$ | 1843 | - |
| Max Engine Force | - | 3750 | N |
| Max Brake Force | - | 14000 | N |

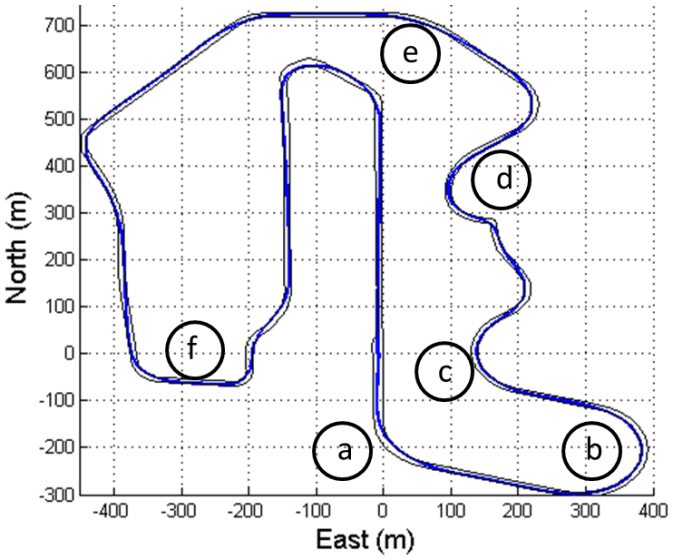


Figure 7. Overhead view of Thunderhill Racing circuit along with generated path from algorithm. Labeled regions a-f will be discussed in more detail.

discrepancies, such as the turn entry in (c) and turn exit in (d). This is because the nonlinear method balances the tradeoff between a minimum curvature path and the path with shortest total distance, and finds regions where it may be beneficial to use less of the available road width.

To further compare the path generation methods, Fig. 10 shows the path curvature profile $K(s)$ for the three racing trajectories. Again, the curvature profiles look similar, with some small discrepancies around $s = 1100$m and $s = 1400$m. The resulting velocity profiles are shown in Fig. 11 for the two-step method and gradient descent method. As a fair comparison,

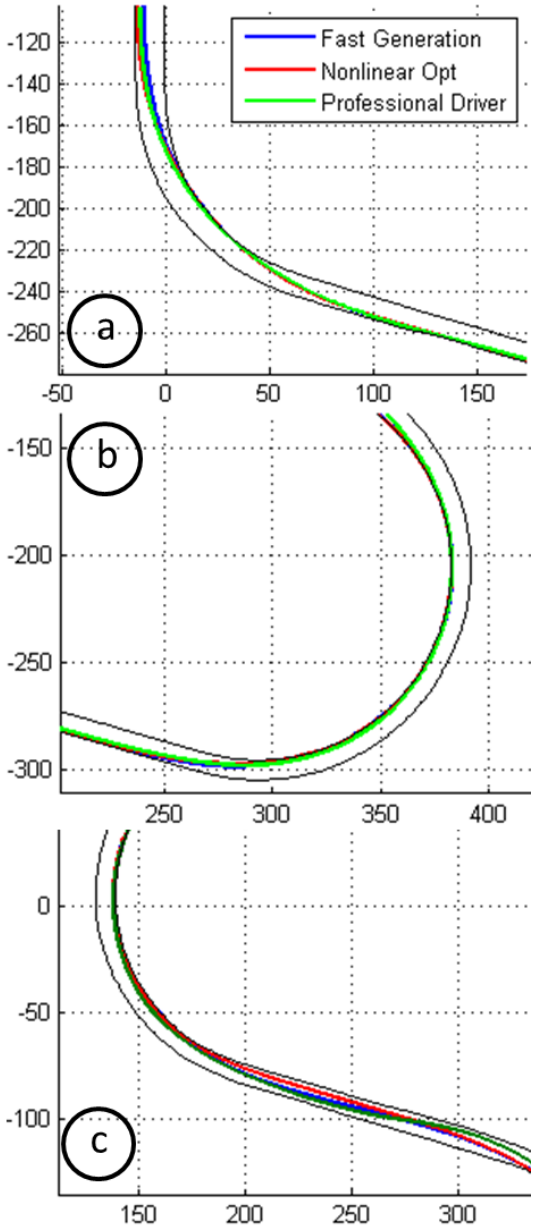


Figure 8. Racing lines from the two-step fast generation approach, nonlinear gradient descent algorithm, and experimental data taken from professional driver. Labeled regions a-c correspond to zoomed-in locations on Fig. 7.

the velocity profiles were both generated using the three-pass 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.

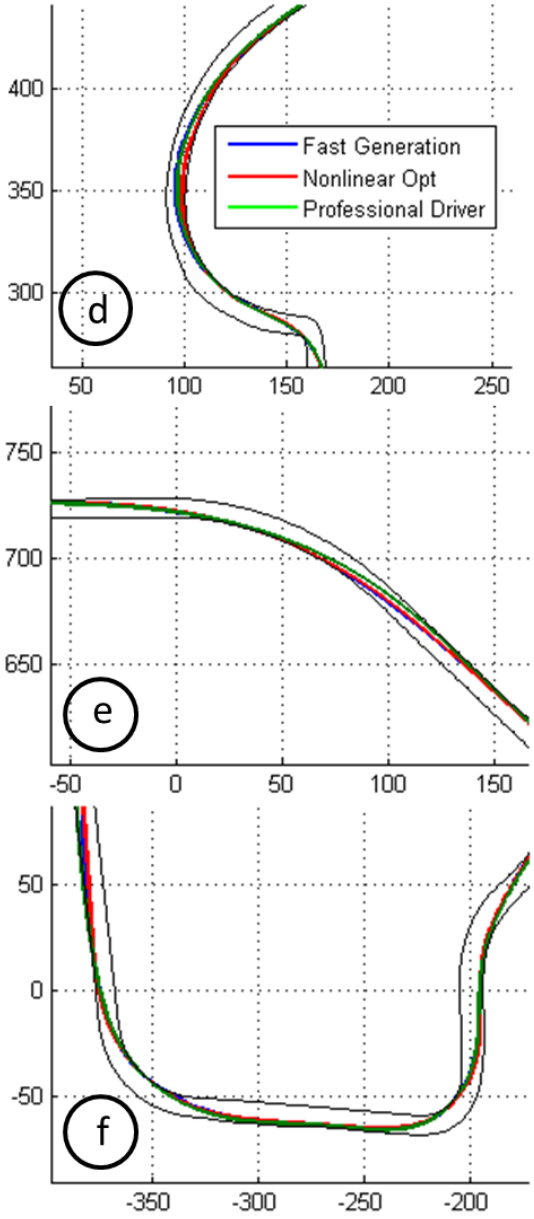


Figure 9. Racing lines from the two-step fast generation approach, nonlinear gradient descent algorithm, and experimental data taken from professional driver. Labeled regions correspond to zoomed-in locations on Fig. 7.

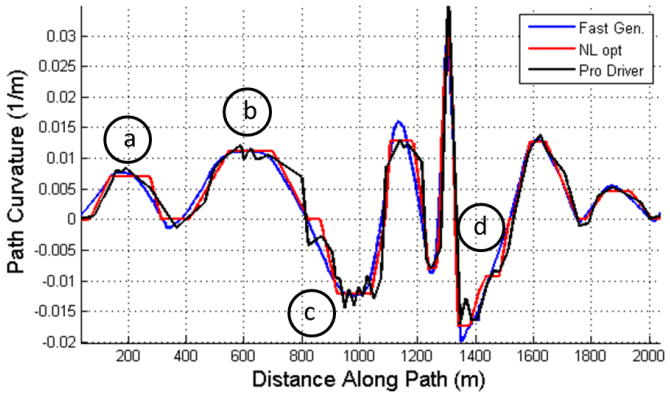


Figure 10. Curvature profile $K(s)$ plotted vs. distance along the path $s$. For improved visibility, results are shown for the first 2km of the track. The noise in the professional driver curvature profile is due to the relative difficulty of converting experimental GPS coordinates into a curvature profile.

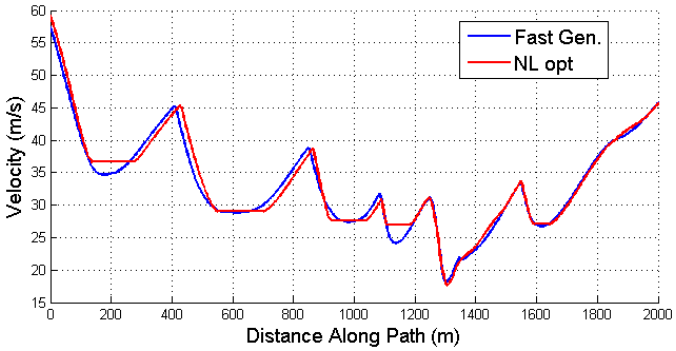


Figure 11. Velocity profile $U_x(s)$ plotted vs. distance along the path $s$ for the two-step method and nonlinear gradient descent method.

## 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), we obtained a more precise lap time by numerically simulating a vehicle following the desired path and velocity profile using the feedback-feedforward controller presented in [5]. The equations of motion are the fully nonlinear versions of (2) with tire forces given by the Fiala 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, and 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 a given nonlinear algorithm but rather a radical improvement in computational simplicity and speed, as each two-step iteration

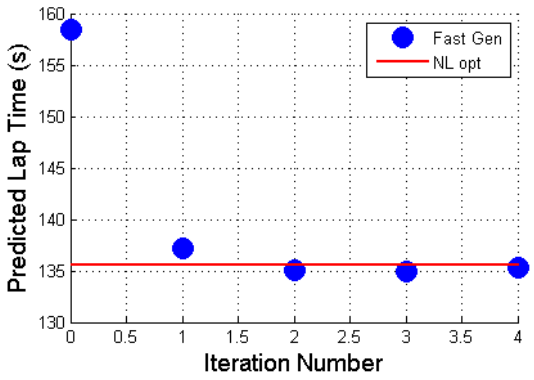


Figure 12. Lap time as a function of iteration for the two-step fast trajectory generation method. Final lap time is comparable to that achieved with the nonlinear gradient descent approach. Iteration zero corresponds to the lap time for driving the initial path.

takes only 26 seconds on an Intel i7 processor. The most significant computational expense is solving the convex curvature minimization problem for all 1843 discrete time steps of the race vehicle over the 5 km course.

## 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 any initial path through the race track, the velocity update step performs forwards-backwards integration to determine an aggressive longitudinal speed profile. Holding this speed profile constant, the path geometry is updated by solving a convex optimization problem with a minimum curvature cost function. The algorithm converges after only three iterations when applied to the Thunderhill Raceway course, and the final vehicle trajectory compares well with results obtained from both a previously published nonlinear gradient descent algorithm as well as experimental data observed from a professional driver. Future work will apply the racing trajectory on a vehicle testbed to determine the experimental performance of the algorithm. Additionally, given the fast computation time of the algorithm, an exciting opportunity for future research is modifying an existing racing trajectory by incorporating parameter information "learned" from an autonomous vehicle driving the desired path at the friction limits.

## 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 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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Lipp, T., and Boyd, S., 2014. "Minimum-time speed optimisation over a fixed path". *International Journal of Control,* **87**(6), pp. 1297–1311.

[10] Subosits, J. K., and Gerdes, J. C., 2015. "Autonomous vehicle control for emergency maneuvers: The effect of topography". In American Control Conference.

[11] 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.

[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.