Learning at the Racetrack: Data-driven Methods to Improve Racing Performance over Multiple Laps

Nitin R. Kapania and J Christian Gerdes

***Abstract*— Autonomous vehicles will generate tremendous data from the variety of sensors they employ to track the surrounding environment. This data is inherently valuable, as it gives algorithm designers the potential to leverage prior experience in order to improve driving performance over time. This paper uses the lens of autonomous racing to provide an example of how data from previous iterations of driving can be used to improve quantitative metrics of performance. Two complementary algorithms are demonstrated in this paper. The first type uses iterative learning control (ILC) to simultaneously improve lateral and longitudinal tracking of the desired racing trajectory from lap-to-lap, while the second type of algorithm is focused on altering the trajectory itself using a search method. When driven experimentally at the limits of handling, the result is a reduction in lap time of nearly 1.4 seconds, a major improvement.**

***Index Terms*—Autonomous driving, iterative learning control, A* search**

I. Introduction

A. Problem Description and Proposed Solution

Several methods exist in the literature to generate autonomous racing trajectories ([1], [2], [3], [4]), which typically comprise of a longitudinal speed profile $U_x(s)$ and a desired curvature profile $\kappa(s)$ as a function of distance along the track. These trajectories can then be tracked using one of several path following controllers, such as ([5], [6], [7], [8], [9]).

Using the trajectory generated by [4] and the trajectory following controller presented by [10], the authors generated a test run of an autonomous race vehicle (see Fig.1(a)) at the limits of handling on the Thunderhill Raceway course in Willows, CA (Fig. 1(b)). The results are shown in Fig. 2.

While the vehicle is able to generate lap times that are comparable to an amateur expert, the resulting lap times are 2-3 seconds below that of a professional human race car driver. There are several ways to analyze why this difference arises, but a simple insight that makes the case for lap-to-lap learning comes from viewing the trajectory tracking performance and friction utilization of the controller, shown in Fig. 2.

One of the issues shown in Fig. 2 is the relatively poor controller tracking on a few labeled sections of the race track. This includes errors in the lateral path tracking (sections ②, ④ - ⑦) and failing to drive at the desired speed (sections ①, ②, ⑦), resulting directly in a loss of lap time. The speed and lateral path tracking will be improved in this paper with the addition of iterative learning control (ILC) algorithms.

The second issue shown in Fig. 2 is the inconsistent usage of the tire friction capacity, as judged by the tire slip norm metric. The tire slip norm, formalized in [11], is given by ζ:

This work was supported in part by the National Science Foundation Graduate Research Fellowship Program and the Stanford Graduate Fellowship.

Nitin Kapania is a postdoctoral fellow at Stanford University and a visiting scholar at UC Berkeley (e-mail: nkapania@stanford.edu).

J. C. Gerdes is a Professor at Stanford University (e-mail: gerdes@stanford.edu).

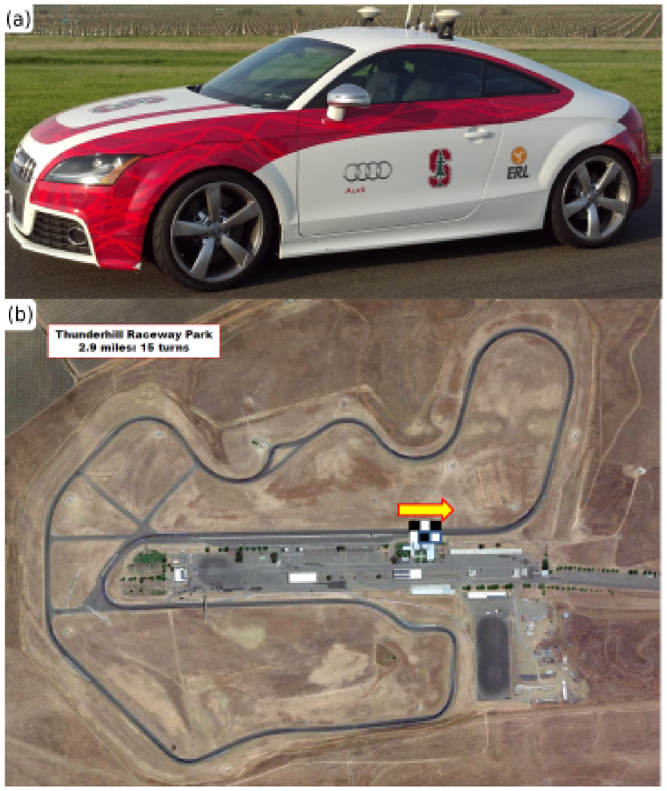

Fig. 1. (a) "Shelley" - Stanford's autonomous self-driving race vehicle. (b) Top-down view of Thunderhill Raceway course.

$$\zeta = \sqrt{\left(\frac{\alpha}{\alpha_p}\right)^2 + \left(\frac{\sigma}{\sigma_p}\right)^2} \tag{1}$$

Where σ and α are the longitudinal and lateral tire slip for a given tire, and σ_p and α_p are empirically determined *peak slip* values resulting in maximum longitudinal and lateral tire force generation. As a result, $\zeta < 1$ corresponds to the tires having excess force generation capacity, while $\zeta > 1$ corresponds to tire saturation.

Fig. 2 shows inconsistent usage of tire friction across many turns. On some turns (sections ② and ⑧), the vehicle significantly exceeds the limits of handling, and the car's stability control systems kick in to regain control, slowing the car down in the process. On other turns (sections ③, ⑤, and ⑦), the vehicle uses only a portion of the available tire force, indicating the vehicle can actually drive with higher acceleration on the next lap. Learning from prior runs to find the optimal acceleration (or μ parameter) for each part of the track will also be accomplished in this paper via an A* search algorithm.

B. Relation to Prior Work

1) Iterative Learning Control: Iterative learning control (ILC) is based on the notion that the performance of a system that executes the *same task* multiple times can be improved by learning from previous

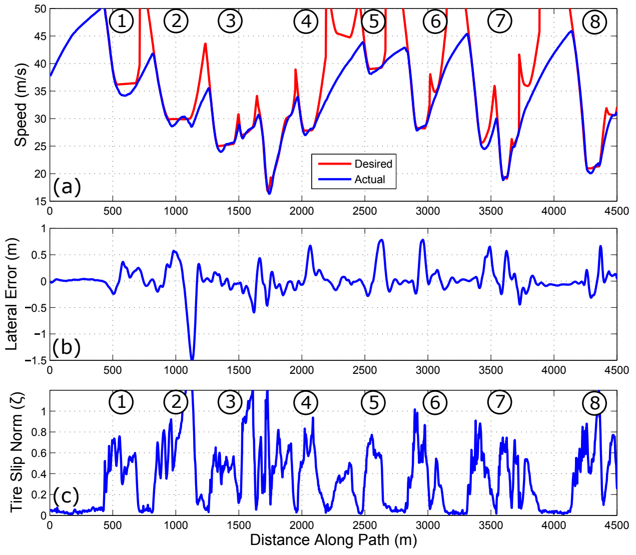

Fig. 2. Controller tracking performance and tire slip norm on a test run at the limits of handling ($\mu = 0.94$).(a) Desired vs actual speed of the vehicle. (b) Lateral tracking error and (c) tire slip norm as a function of distance along the path.

executions (trials, iterations, or in our case, laps of racing) [12]. On every iteration, a control signal is applied to a system in order to follow an ideal, unchanging "reference trajectory". The tracking error for that iteration is recorded, and a learning algorithm is applied to improve the control signal and achieve more accurate system performance on the next iteration. There are a variety of learning algorithms used, but most attempt to correct the tracking error by using a model of the system to determine the augmentation to apply to the prior control signal. This process is repeated until the reference tracking performance becomes satisfactory.

Because iterative learning control works best when learning to follow the same reference trajectory under nearly constant ambient conditions, the most common applications of ILC are in the field of automated manufacturing. Notable examples include CNC machining [13], industrial robotics [14] [15], piezolectric stage positioning [16], motor control [17], and microdeposition [18]. However, the rise of automated systems outside factory environments has led to important applications of ILC for ground and air robotics. Chen and Moore [19] proposed a simple iterative learning scheme to improve path-following of a ground vehicle with omni-directional wheels. Purwin and Andrea synthesized an iterative controller using least-squares methods to aggressively maneuver a quadrotor unmanned aerial vehicle (UAV) from one state to another [20]. Sun et al. [21] proposed an iterative learning controller for speed regulation of high-speed trains. Other novel applications where ILC has been applied include Homogeneous Charge Compression Ignition (HCCI) [22] and wing vibration suppression [23].

2) Trajectory Modification Algorithms: Iterative learning control (ILC) algorithms can help an autonomous vehicle follow a desired trajectory more precisely over several laps of driving. However, ILC algorithms do not alter the desired trajectory itself, only the input signals that attempt to track the trajectory. This will not be sufficient at the limits of handling. Consider Fig. 2, where the desired speed profile was generated assuming a tire-road friction value of $\mu = 0.94$. While this is a reasonable assumption overall, there are several parts on the track where the vehicle exceeds the available friction and begins to understeer. Region $\textcircled{2}$ is one example. The tire slip norm ζ climbs above one, and as a result, the vehicle begins to slide off the track, resulting in the large negative tracking error spike in Fig. 2(b). The vehicle's stability algorithms must kick in and slow the vehicle down, which results in the speed tracking error seen in Fig. 2(a). In this situation, ILC will be unable to achieve better speed tracking and path tracking performance. The tires are saturated and the car is slowly beginning to careen off the track. Simply steering more on the next lap will not achieve better tracking performance due to the saturation of the steering actuator. In order to recover, the vehicle must deviate from the planned trajectory, either by slowing down or by taking a wider radius turn.

Some trajectory modification algorithms previously investigated in the literature have focused on modifying the *lateral* trajectory by altering the curvature profile as the vehicle understeers or oversteers. For example, Theodosis and Gerdes presented an algorithm to gradually widen the radius of a turn in response to a detected understeer [24]. The algorithm was validated experimentally, but assumed sufficient availability of road width. Klomp and Gordon also developed a strategy for recovering from vehicle understeer by solving for an optimal emergency braking profile to minimize deviation from the desired path [25]. On the other hand, Brown et al. [26] presented a model predictive control (MPC) approach that generally aimed to follow a planned vehicle trajectory at the limits. However, if the vehicle was at risk of understeering or oversteering, the MPC algorithm would deviate laterally from the planned trajectory in order to maintain stability of the vehicle without driving off the road.

Another approach to trajectory modification is Learning Model Predictive Control (LMPC), developed by Rosolia et al. [27] for the purpose of autonomous racing. In LMPC, data from every lap of racing is used to build invariant sets and approximate the value function for the generally non-convex lap time minimization problem. In practice, this enables the controller to explore different areas of the state space on each successive lap, modifying the desired trajectory on each lap in order to continuously improve the lap time.

C. Order of Contributions

Due to the repetitive nature of automotive racing, iterative learning control techniques are a promising method to gradually eliminate *trajectory tracking* errors . For this application of ILC, the repetitive trials are laps of racing, and the reference trajectory is the optimal speed profile and curvature developed in [4]. This paper will present adaptations of two established ILC methods (proportional-derivative and quadratically optimal) for use in the Audi TTS racing system. Section II-A presents both coupled and decoupled models of the vehicle lateral and longitudinal dynamics. These models are converted from state-space representations to *lifted domain* representations required for iterative learning control, and the proportional-derivative and quadratically optimal ILC algorithms are presented in §II-C and §II-D. Section II-E presents simulated results of the ILC algorithms for a sample vehicle trajectory, and finally, experimental results showing a gradual reduction of trajectory-following errors is presented in §II-F.

For situations where the vehicle underperforms due to excessive oversteer or understeer, an A* search algorithm is presented that iteratively modifies portions of the *velocity* profile to be more conservative if a stability violation is encountered on a prior lap. This is accomplished by generalizing the speed profile such that each part of the track can be driven with a learned value of friction μ and therefore a different maximum acceleration. The algorithm also modifies the velocity profile to be more aggressive if the tires are not being actuated at the limits. In this way, the algorithm is focused on learning the time-optimal friction profile $\mu^\star(s)$ by searching through datasets obtained over multiple laps.

II. Iterative Learning Control

A. Dynamic System Model

The ILC algorithms we consider require a set of closed-loop system dynamics that are (a) stable to any disturbance input, and (b) expressible as an *affine* discrete dynamical system. In our case, we have two subsystems: the steering controller and the longitudinal speed control. Stability of the steering controller under lanekeeping feedback was shown in the linear case by [28] and in the saturated case by [29]. Similar analyses can be considered to show the stability of a simple proportional speed-following controller.

The more difficult task is expressing the dynamics of the two subsystems using an affine model, given the tendency for the vehicle tires to saturate at the limits. This paper will model the dynamics with a bicycle model with single longitudinal input F_x and front steer input δ, shown in Fig.3.

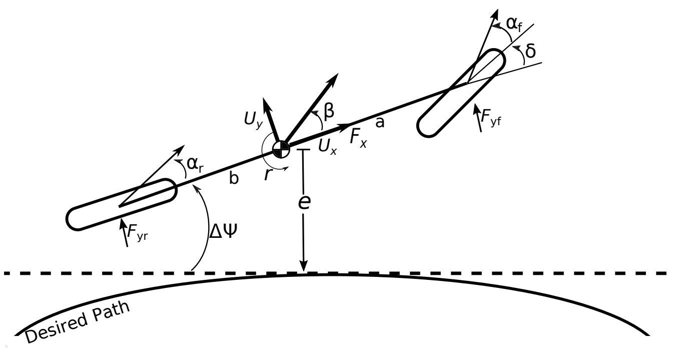

Fig. 3. Schematic of bicycle model with steer input $\boldsymbol{\delta}$ and longitudinal input $\boldsymbol{F_x}$.

Nonlinear, coupled equations of motion are provided in (2)-(6):

$$\frac{de}{dt} = \left(v + U_x^{\text{des}}(s)\right)(\beta + \Delta\Psi) \tag{2}$$
$$\frac{dv}{dt} = \left(v + U_x^{\text{des}}(s)\right)\beta r + \frac{F_x}{m} - \frac{F_{\text{yf}}(\alpha_{\text{f}}, F_x)\delta}{m} \tag{3}$$
$$\frac{d\beta}{dt} = \frac{F_{\text{yf}}(\alpha_{\text{f}}, F_x) + F_{\text{yr}}(\alpha_{\text{r}}, F_x)}{m\left(v + U_x^{\text{des}}(s)\right)} - r \tag{4}$$
$$\frac{dr}{dt} = \frac{aF_{\text{yf}}(\alpha_{\text{f}}, F_x) - bF_{\text{yr}}(\alpha_{\text{r}}, F_x)}{I_z} \tag{5}$$
$$\frac{d\Delta\Psi}{dt} = r \tag{6}$$

The system dynamics presented in (2) have five states, yaw rate r, vehicle sideslip β, lateral path tracking error e, heading error $\Delta\Psi$, and speed tracking error v, defined by:

$$v = U_x - U_x^{\text{des}} \tag{7}$$

The potential for coupling between the subsystems is apparent from (2)-(6), not only directly from the state equations but also due to the complex nature of tire force generation at the handling limits. As shown in Fig. 4, as longitudinal force F_x is distributed across the tires, the available lateral force decreases. At the limits of handling, this *derating* of the lateral force may become significant. As a result, we model lateral tire force generation F_y as a function of both lateral tire slip α and the applied longitudinal force F_x, using a modified form of the Fiala tire model [30]. The front and rear tire slip are themselves functions of the vehicle state and are given by:

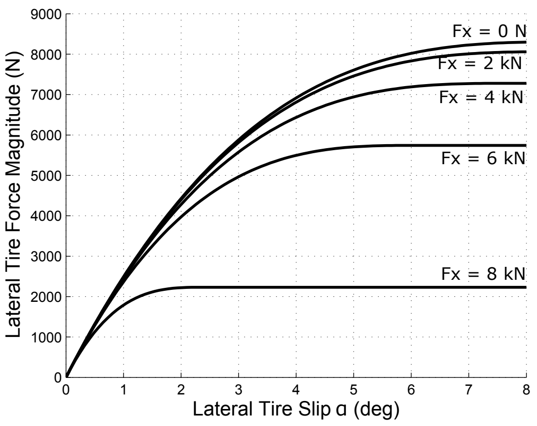

Fig. 4. Lateral tire force curve as a function of longitudinal force $\boldsymbol{F_x}$ and lateral tire slip $\boldsymbol{\alpha}$.

$$\alpha_{\text{f}} = \beta + \frac{ar}{U_x} - \delta \tag{8a}$$
$$\alpha_{\text{r}} = \beta - \frac{br}{U_x} \tag{8b}$$

The next step is to break up the control inputs into the closed-loop feedback term and the *learned* component that is modified on every lap by the ILC algorithm:

$$F_x = F_x^{\text{FB}} + F_x^L \tag{9}$$
$$= -K_x v + F_x^L \tag{10}$$
$$\delta = \delta_{\text{FB}} + \delta_L \tag{11}$$
$$\delta = -k_p(e + x_{\text{LA}}\Delta\Psi) + \delta_L \tag{12}$$

where K_x is the proportional speed tracking gain and k_p and x_{LA} are the lookahead gains presented in [10].

The closed loop system dynamics are now given by:

$$\frac{de}{dt} = \left(v + U_x^{\text{des}}(s)\right)(\beta + \Delta\Psi) \tag{13}$$
$$\frac{dv}{dt} = \left(v + U_x^{\text{des}}(s)\right)\beta r + \frac{-K_x v + F_x^L}{m} \tag{14}$$
$$- \frac{F_{\text{yf}}(\alpha_F, F_x)(k_p(e + x_{LA}\Delta\Psi) + \delta_L)}{m} \tag{15}$$
$$\frac{d\beta}{dt} = \frac{F_{\text{yf}}(\alpha_{\text{f}}, F_x) + F_{\text{yr}}(\alpha_{\text{r}}, F_x)}{m\left(v + U_x^{\text{des}}(s)\right)} - r \tag{16}$$
$$\frac{dr}{dt} = \frac{aF_{\text{yf}}(\alpha_{\text{f}}, F_x) - bF_{\text{yr}}(\alpha_{\text{r}}, F_x)}{I_z} \tag{17}$$
$$\frac{d\Delta\Psi}{dt} = r \tag{18}$$

The nonlinear closed-loop dynamics must be converted into an affine, discrete-time dynamical system to apply conventional iterative learning control algorithms. In our case, we have two system outputs ($y = [e \;\; v]^T$) that are measured and two input signals to learn ($u = [\delta_L \;\; F_x^L]^T$). Since we run the iterative learning control algorithm after seeing a trial of data, we can approximate the dynamics in (13)-(18) by linearizing about the observed states and inputs x_o, u_o

from the first lap. The affine model is therefore given by:

$$\begin{bmatrix} \dot{e} \\ \dot{\beta} \\ \dot{r} \\ \Delta\dot{\Psi} \\ \dot{v} \end{bmatrix} = [A(t)] \begin{bmatrix} e \\ \beta \\ r \\ \Delta\Psi \\ v \end{bmatrix} + [B(t)] \begin{bmatrix} \delta_L \\ F_x^L \end{bmatrix} + d(t) \tag{19}$$

$$\begin{bmatrix} \dot{e} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} x \tag{20}$$

The time-varying state space matrices $A(t)$ and $B(t)$ are given by Jacobian linearizations of the closed loop nonlinear dynamics (13) about the observed states and inputs $x_o(t), u_o(t)$ from the last trial:

$$[A(t)] = \frac{\partial f}{\partial x}\Big|_{x_o(t)} \tag{21}$$

$$[B(t)] = \frac{\partial f}{\partial u}\Big|_{u_o(t)} \tag{22}$$

While this multiple-input, multiple-output (MIMO) model captures the coupled behavior of the longitudinal and lateral inputs, it is tedious to compute, either numerically or analytically. If we make the assumption that the longitudinal and lateral inputs are decoupled and make small angle approximations, we obtain the following linear state equations for the lateral dynamics:

$$\dot{\beta} = \frac{F_{\text{yf}} + F_{\text{yr}}}{mU_x} - r \qquad \dot{r} = \frac{aF_{\text{yf}} - bF_{\text{yr}}}{I_z} \tag{23a}$$

$$\dot{e} = U_x(\beta + \Delta\Psi) \qquad \Delta\dot{\Psi} = r - U_x\kappa \tag{23b}$$

and the following first order equation for the longitudinal dynamics, assuming a simple point mass model with proportional speed tracking feedback:

$$\dot{v} = \frac{-K_x v + F_x^L}{m} \tag{24}$$

The resulting state matrices $A(t)$ and $B(t)$ for (19) are then simply block diagonal matrices consisting of the linearized lateral dynamics and the first order longitudinal dynamics:

$$A(t) = \tag{25}$$
$$\begin{bmatrix} 0 & U_x(t) & 0 & U_x(t) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \frac{-ak_{\text{p}}\tilde{C}_{\text{f}}(t)}{I_{\text{z}}} & \frac{-ak_{\text{p}}x_{\text{LA}}\tilde{C}_{\text{f}}(t)}{I_{\text{z}}} & \frac{-a^2\tilde{C}_{\text{f}}(t)-b^2\tilde{C}_{\text{r}}(t)}{U_x(t)I_{\text{z}}} & \frac{b\tilde{C}_{\text{r}}(t)-a\tilde{C}_{\text{f}}(t)}{I_{\text{z}}} & 0 \\ \frac{-k_{\text{p}}\tilde{C}_{\text{f}}(t)}{mU_x(t)} & \frac{-k_{\text{p}}x_{\text{LA}}\tilde{C}_{\text{f}}(t)}{mU_x(t)} & \frac{b\tilde{C}_{\text{r}}(t)-a\tilde{C}_{\text{f}}(t)}{mU_x(t)^2}-1 & \frac{-\tilde{C}_{\text{f}}(t)-\tilde{C}_{\text{r}}(t)}{mU_x(t)} & 0 \\ 0 & 0 & 0 & 0 & -K_x \end{bmatrix}$$

$$B(t) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{a\tilde{C}_{\text{f}}(t)}{I_{\text{z}}} & 0 \\ \frac{\tilde{C}_{\text{f}}(t)}{mU_x(t)} & 0 \\ 0 & 1 \end{bmatrix} \tag{26}$$

The affine term $d(t)$ is given by:

$$d(t) = \begin{bmatrix} 0 \\ -\kappa(t)U_x(t) \\ \frac{a\tilde{C}_{\text{f}}(t)\tilde{\alpha}_{\text{f}}(t)-b\tilde{C}_{\text{r}}(t)\tilde{\alpha}_{\text{r}}(t)+a\tilde{F}_{\text{yf}}(t)-b\tilde{F}_{\text{yr}}(t)}{I_z} \\ \frac{\tilde{C}_{\text{f}}(t)\tilde{\alpha}_{\text{f}}(t)+\tilde{C}_{\text{r}}(t)\tilde{\alpha}_{\text{r}}(t)+\tilde{F}_{\text{yf}}(t)+\tilde{F}_{\text{yr}}(t)}{mU_x(t)} \\ 0 \\ 0 \end{bmatrix} \tag{27}$$

Where $\tilde{F}_{\text{yf,r}}(t)$ are the steady-state front and rear tire forces from the first lap, $\tilde{C}_{\text{f,r}}(t)$ are the effective cornering stiffnesses at those tire forces, and $\tilde{\alpha}_{\text{f,r}}(t)$ are the corresponding front and rear tire slips. While (25) is written as a MIMO system for compactness, assuming decoupled lateral and longitudinal dynamics provides two single-input, single-output (SISO) systems.

B. Lifted Domain Representation and ILC Problem Statement

Whether the coupled (21) or decoupled (25) dynamics are assumed, the final modeling step is to apply standard discretization techniques to obtain dynamics in the following form:

$$x_{k+1} = A_k x_k + B_k u_k + d_k \tag{28}$$

$$y_k = Cx_k \tag{29}$$

For a given lap of racing j, sensor measurements provide N observations of both the lateral path deviation e and longitudinal speed tracking error v. These measurements can be stacked into a $2N \times 1$ array:

$$\mathbf{e}_j = \begin{bmatrix} e_1 & \dots & e_N & v_1 & \dots & v_N \end{bmatrix}^T \tag{30}$$

These measurement errors are related to the learned control inputs δ^L and F_x^L as follows:

$$\mathbf{e}_j = P\mathbf{u}_j^L + \mathbf{w} \tag{31}$$

$$\mathbf{u}_j^L = \begin{bmatrix} \delta_1^L & \dots & \delta_N^L & F_{x1}^L & \dots & F_{xN}^L \end{bmatrix}^T \tag{32}$$

The system dynamics modeled in the previous section are represented by the *lifted-domain* dynamics matrix P, which is $2N \times 2N$ and given by:

$$P = \left[\begin{array}{c|c} P_{e\delta} & P_{eF} \\ \hline P_{v\delta} & P_{vF} \end{array} \right] \tag{33}$$

Where each submatrix in (33) is $N \times N$ and represents the lifted-domain dynamics from a given input to a given output. Individual terms of the sub-matrices are given by:

$$p_{lk} = \begin{cases} 0 & \text{if } l < k \\ C_y B_u(k) & \text{if } l = k \\ C_y A(l)A(l-1)\cdots A(k)B_u(k) & \text{if } l > k \end{cases} \tag{34}$$

Where C_y is the row of C in (28) corresponding to the desired output and B_u the column of B in (28) corresponding to the desired input. Note that for the case of uncoupled lateral and longitudinal dynamics, the off-diagonal sub-matrices of P are $[0]$ since we have two SISO systems. The term $\mathbf{w}$ in (31) is the unknown disturbance. Iterative learning control relies on the assumption that this disturbance is the underlying cause of the observed errors $\mathbf{e}_j$, and that the disturbance, while unknown, is constant from lap to lap.

Given the error signal $\mathbf{e}_j$ for a given lap j, the iterative learning problem is to find the inputs $\mathbf{u}_{j+1}$ that will cancel out the tracking error on the next lap. The learned inputs are then applied, the observed error $\mathbf{e}_{j+1}$ is recorded, and the process is repeated until the tracking error falls to a desired level. There is a wide body of literature on methods to determine $\mathbf{u}_{j+1}$ given P and $\mathbf{e}_j$, but this paper will compute the ILC input for the next lap with the following formulation:

$$\mathbf{u}_{j+1}^L = Q(\mathbf{u}_j^L - L\mathbf{e}_j) \tag{35}$$

where Q is the $2N \times 2N$ *filter* matrix, and L is the $2N \times 2N$ *learning* matrix. In the following two sections, the matrices Q and L will be obtained by designing a proportional-derivative (PD) iterative learning controller as well as a quadratically optimal (Q-ILC) learning controller.

C. Proportional-Derivative Controller

The proportional-derivative ILC computes the steering δ^L and force F^L correction for the current lap j based on the error and error derivative at the same time index k from the previous lap:

$$\delta_j^L(k) = \delta_{j-1}^L(k) - k_{p\delta}e_{j-1}(k) - k_{d\delta}(e_{j-1}(k) - e_{j-1}(k-1)) \tag{36}$$

$$F_j^L(k) = F_{j-1}^L(k) - k_{pF}v_{j-1}(k) - k_{dF}(v_{j-1}(k) - v_{j-1}(k-1)) \tag{37}$$

where $k_{p\delta}$ and k_{pF} are proportional gains and $k_{d\delta}$ and k_{dF} are derivative gains. In the lifted domain representation from (35), the resulting learning matrix L is given by

$$L = \begin{bmatrix} -(k_{p\delta}+k_{d\delta}) & & 0 & 0 & \dots & 0 \\ k_{d\delta} & \ddots & & \vdots & \ddots & \vdots \\ 0 & k_{d\delta} & -(k_{p\delta}+k_{d\delta}) & 0 & \dots & 0 \\ 0 & \dots & 0 & -(k_{pF}+k_{dF}) & & 0 \\ \vdots & \ddots & \vdots & k_{dF} & \ddots & \\ 0 & \dots & 0 & 0 & k_{dF} & -(k_{pF}+k_{dF}) \end{bmatrix} \tag{38}$$

The PD equation (36) determines δ^L only using lateral path deviation e and F_x^L using only the speed tracking error v. The filter matrix Q is obtained by taking any filter transfer function and converting into the lifted domain via (34). An important design consideration in choosing the two k_p and k_d gains is avoiding a poor lap-to-lap "transient" response, where the path tracking error increases rapidly over the first several laps before eventually decreasing to a converged error response e_∞. This is a commonly encountered design requirement for ILC systems, and can be solved by ensuring the following *monotonic convergence* condition is met [12]:

$$\gamma \triangleq \bar{\sigma}(PQ(I - LP)P^{-1}) < 1 \tag{39}$$

where $\bar{\sigma}$ is the maximum singular value. In this case, the value of γ provides an upper bound on the change in the tracking error norm from lap to lap, i.e.

$$||\mathbf{e}_\infty - \mathbf{e}_{j+1}||_2 \leq \gamma||\mathbf{e}_\infty - \mathbf{e}_j||_2 \tag{40}$$

Fig. 5 shows values of γ for both an unfiltered PD controller ($Q = I$), and for a PD controller with a 2 Hz, first order low pass filter. The γ values are plotted as a contour map against the controller gains $k_{p\delta}$ and $k_{d\delta}$. Addition of the low-pass filter assists with monotonic stability by removing oscillations in the control input generated when trying to remove small reference tracking errors after several iterations. Since the filtering occurs when generating a control signal for the next lap, the filter Q can be zero-phase. The plot shown in Fig. 5 is for the steering ILC design only, but the same analysis is possible for the longitudinal ILC design as well. Because the P matrix in general can change from iteration to iteration and will also change depending on the trajectory, making a general assertion or proof about stability of the iterative learning controller is difficult.

D. Quadratically Optimal Controller

An alternate approach to determining the learned steering and longitudinal force input is to minimize a quadratic cost function for the next lap:

$$J_{j+1} = \mathbf{e}_{j+1}^T T\mathbf{e}_{j+1} + \mathbf{u}_{j+1}^T R\mathbf{u}_{j+1}^T + \Delta_{j+1}^T S\Delta_{j+1} \tag{41}$$

where $\Delta_{j+1} = \mathbf{u}_{j+1}^L - \mathbf{u}_j^L$ and the $2N \times 2N$ matrices T, R, and S are weighting matrices, each given by a scalar multiplied by the identity matrix for simplicity. This formulation allows the control designer to

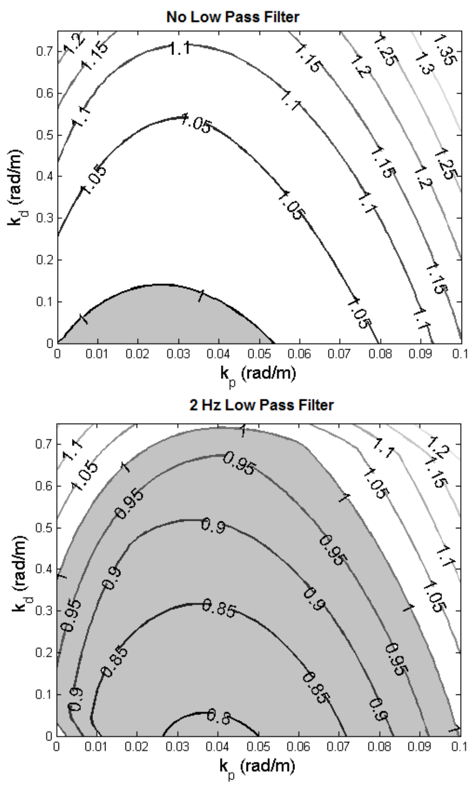

Fig. 5. Values of convergence bound γ vs. $\boldsymbol{k_{p\delta}}$ and $\boldsymbol{k_{d\delta}}$ for PD iterative learning controller with (top) no filtering and (bottom) with a 2 Hz low-pass filter. Lower values of γ correspond to faster convergence. Shaded regions correspond to gains that result in system monotonic stability. The stability analysis in Fig. 5 assumes the $\boldsymbol{P}$ matrix is constant for all iterations and is generated assuming decoupled vehicle dynamics for a straight-line trajectory at a constant speed of 20 m/s.

weight the competing objectives of minimizing the tracking errors e and v, control effort $|\delta^L|$ and $|F_x^L|$, and change in the control signal from lap to lap. While constraints can be added to the optimization problem, the unconstrained problem in (41) can be solved analytically [31] to obtain desired controller and filter matrices:

$$Q = (P^TTP + R + S)^{-1}(P^TTP + S) \tag{42a}$$
$$L = (P^TTP + S)^{-1}P^TTP(T^{1/2}P)^{-1}T^{1/2} \tag{42b}$$

An advantage of the quadratically optimal control design over the simple PD controller is that the controller matrices Q and L take the linearized, time-varying dynamics P into account. This allows the iterative learning algorithm to take into account changes in the steering dynamics due to changes in vehicle velocity. Furthermore, if the fully coupled dynamics (21) are used, the iterative learning algorithm also accounts for the second-order effect of steering on the longitudinal dynamics and longitudinal force application on the lateral dynamics. However, a disadvantage is that computing δ^L in (35) requires matrix multiplications with the typically dense matrices Q and L for every lap, which can be computationally expensive for fast sampling rates.

E. Simulated Results

To test the feasibility of the PD and Q-ILC learning algorithms, the vehicle tracking performance over multiple laps is simulated using the path curvature and speed profiles shown in Fig. 6. To test the performance of the controller at varying accelerations, four speed profiles are tested. Each profile is generated with a different level of peak combined longitudinal/lateral acceleration, ranging from $5\ \mathrm{m/s^2}$ (below the limits) up to $9.5\ \mathrm{m/s^2}$ (close to exceeding the limits). For accurate results, simulations were conducted using numerical integration with fully nonlinear equations of motion (2)-(6) and coupled lateral/longitudinal dynamics [30].

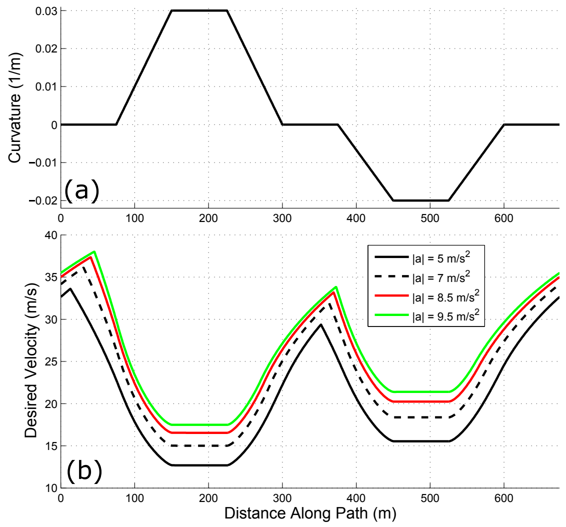

Fig. 6. (a) Curvature profile used for ILC simulation. (b) Velocity profiles generated for four different levels of longitudinal/lateral acceleration.

Simulated results for the root-mean-square (RMS) lateral path deviation is shown in Fig. 7. The results show the change in RMS error as the number of ILC iterations increase. Three different ILC controllers are tested. The first controller is the simple PD controller with low-pass filter (36), and the second controller is the quadratically optimal ILC algorithm (42) assuming fully coupled dynamics (21) in the plant matrix P (i.e. the full MIMO system). The third controller is also the quadratically optimal ILC algorithm, but the P matrix used in the optimization assumes decoupled dynamics (25) and therefore solves the lateral and longitudinal SISO problems separately.

Fig. 7 shows that both quadratically optimal ILC algorithms exponentially reduce the lateral path tracking error as the number of learning iterations is increased. Overall, the RMS lateral tracking performance is better at lower vehicle accelerations. This is unsurprising for two reasons. First, lateral path deviation in general increases for the lookahead steering feedback at higher accelerations. Second, our estimate of the vehicle dynamics contained in P is based on linearization, and the vehicle dynamics at lower accelerations are mostly linear.

Fig. 8 shows the same results as Fig. 7, but for the speed tracking performance. The overall trends are very similar. In both plots, there is very little difference between the coupled MIMO formulation and decoupled SISO formulation at low accelerations. This is expected, as the longitudinal and lateral dynamics are independent when the vehicle tires are not saturated. At higher accelerations, there are small differences, but the overall RMS errors are still quite similar. A reason for this is the nature of the speed profiles in Fig. 6. The vehicle spends the majority of time either fully braking/accelerating or turning at a constant velocity. There are only a few small transient regions where the vehicle needs significant amounts of both lateral and longitudinal acceleration. As a result, the need to account for the coupled dynamics may not be important in practice, especially

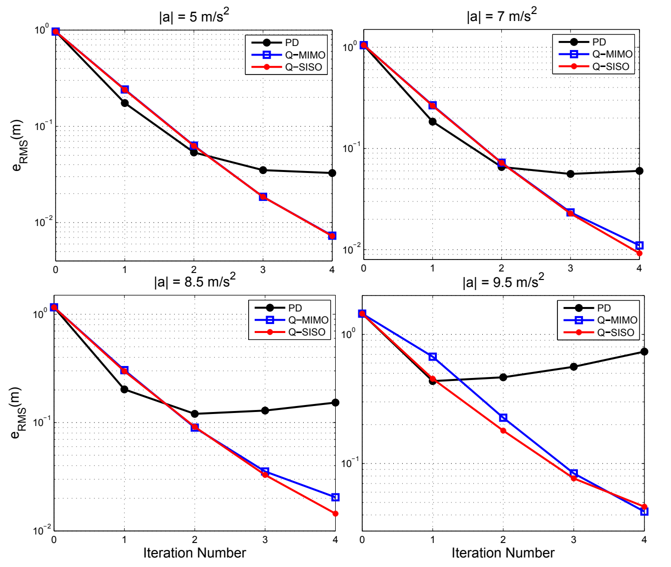

Fig. 7. Simulated results for root-mean-square path tracking error at several values of vehicle acceleration, with $\boldsymbol{T} = \boldsymbol{R} = \boldsymbol{I}$ and $\boldsymbol{S} = 100\boldsymbol{I}$. Results are plotted on a log scale.

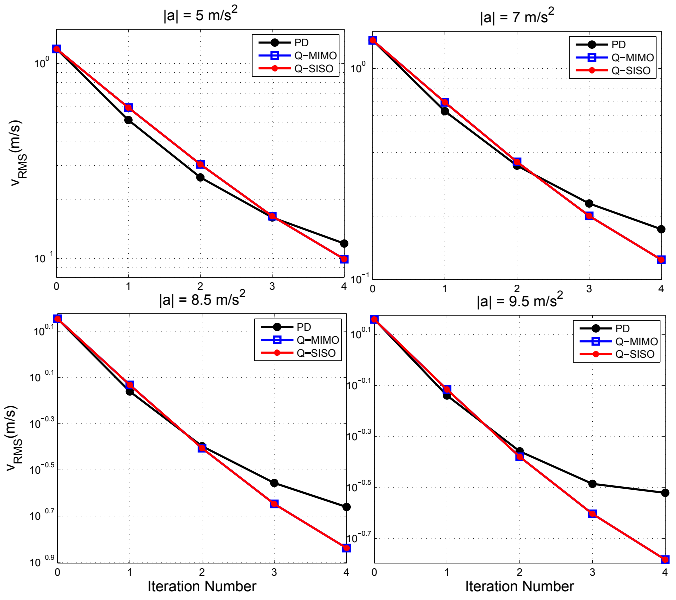

Fig. 8. Simulated results for root-mean-square speed tracking error $\boldsymbol{v}$ at several values of vehicle acceleration, with $\boldsymbol{T} = \boldsymbol{I}$, $\boldsymbol{R} = \boldsymbol{0}$, and $\boldsymbol{S} = \mathbf{1e} - 7\boldsymbol{I}$. Results are plotted on a log scale.

given the larger computation time needed when P is dense and not block-diagonal.

A final comment is that the proportional-derivative ILC algorithm performs relatively poorly. At low accelerations, the speed and path tracking performance both improve initially, but fail to improve after the second learning iteration. At high lateral and longitudinal accelerations, the tracking performance becomes even worse for the steering ILC in Fig. 7. This is unsurprising given that the linearized plant dynamics P are not explicitly accounted for in the selection of the PD gains. While the point mass model for the longitudinal dynamics is a relatively simple first order model, the lateral dynamics are fourth order and highly speed dependent. A simple PD approach for learning control is likely insufficient at the limits of handling.

TABLE I
VEHICLE PARAMETERS

Parameter	Symbol	Value	Units
Lookahead Distance	x_{LA}	15.2	m
Lanekeeping Gain	k_{LK}	0.053	rad m^{-1}
Lanekeeping Sample Time	t_s	0.005	s
ILC Sample Time	T_s	0.1	s
Speed Tracking Gain	K_x	2500	Nsm^{-1}
Q-ILC Matrix (Path)	T and R	I	-
Q-ILC Matrix (Path)	S	100 I	-
Q-ILC Matrix (Speed)	T	I	-
Q-ILC Matrix (Speed)	R	0	-
Q-ILC Matrix (Speed)	S	1e-7 I	-

F. Experimental Results

Experimental data for iterative learning control was collected over four laps at Thunderhill Raceway with the autonomous Audi TTS described in Fig. 1. The controller parameters are shown in Table I.

To save computation time, the learned controller inputs δ^L and F_x^L are calculated at a 10 Hz update rate at the end of every race lap and stored as lookup tables in the controller. Since the the real-time control occurs at 200 Hz, at every time step the controller interpolates the lookup table and applies the correct force and steer angle correction.

Fig. 9 shows the applied iterative learning signals and resulting path tracking error over four laps using the SISO quadratically optimal learning algorithm. The car is driven aggressively at peak lateral/longitudinal accelerations of 8 $\mathrm{m/s^2}$. On the first lap, despite the incorporation of a feedforward-feedback controller operating at a high sampling rate, several spikes in tracking error are visible due to transient vehicle dynamics.

However, the iterative learning algorithm is able to significantly attenuate these transient spikes after just two or three laps. One of the most important features of the time series plot is that the learned steering corrections are applied slightly before a lateral path deviation was observed the prior lap (i.e. the steering corrections *lead* the observed error). This is because the learning algorithm has knowledge of the system model and knows that a steering correction must be applied a few meters early to cancel a path deviation further down the road.

Fig. 10 shows the iterative learning signals for the longitudinal speed control at a slightly higher acceleration of 8.5 $\mathrm{m/s^2}$. Again, in just two to three iterations, significant lags in the speed tracking are attenuated from $s = 800 - 900$ meters and $s = 1200 - 1300$ meters. Additionally, the controller also acts to slow the car down when $v > 0$ and the vehicle exceeds the speed profile. This is also desirable from a racing perspective as it it prevents the vehicle from exceeding the friction limit. Oscillations in speed tracking performance are visible in Fig. 9(a) from 900 - 1100 meters and 1300 - 1400 meters. These are generally undesirable, and further tuning of the filter matrix Q is possible to remove rapid changes in the learned force input.

Notice that Fig. 10 has several regions where the speed error $v << 0$. These are straight regions of the track where is no true planned speed because the desired longitudinal action is to fully apply the throttle and go as fast as physically possible. For convenience, the ILC is programmed to saturate the longitudinal learning signal to 8000 Newtons, although a more elegant solution is to switch the ILC controller off on straight portions of the track.

In Fig. 11, root-mean-square tracking results are shown for a range of peak vehicle accelerations. The results show that at lower vehicle accelerations, the initial speed and lateral tracking errors (Iteration 0) are smaller, as the built-in feedback-feedforward controller performs better. However, as the speed profile becomes more aggressive, the path and speed tracking degrades in the presence of highly transient tire dynamics. Regardless of the initial error, application of iterative learning control reduces the trajectory tracking errors significantly over just 2 or 3 laps. At an acceleration of 8.5 $\mathrm{m/s^2}$, for example, the RMS lateral tracking error is around 3 cm, on the order of the expected RMS error from the GPS position sensor. On some tests, the RMS tracking error increases slightly from Lap 2 to Lap 3, and for the case where vehicle acceleration is 9 $\mathrm{m/s^2}$, the lateral tracking error is constant from Lap 1 to Lap 2 before decreasing further in Lap 3. While not predicted in simulation, this behavior likely occurs because the repeating disturbance from lap-to-lap is not exactly constant, especially as the vehicle approaches the handling limits. More refined tuning of the gain matrices may be able to prevent this RMS error increase, or the ILC algorithm can be stopped after several iterations once the tracking performance is acceptable.

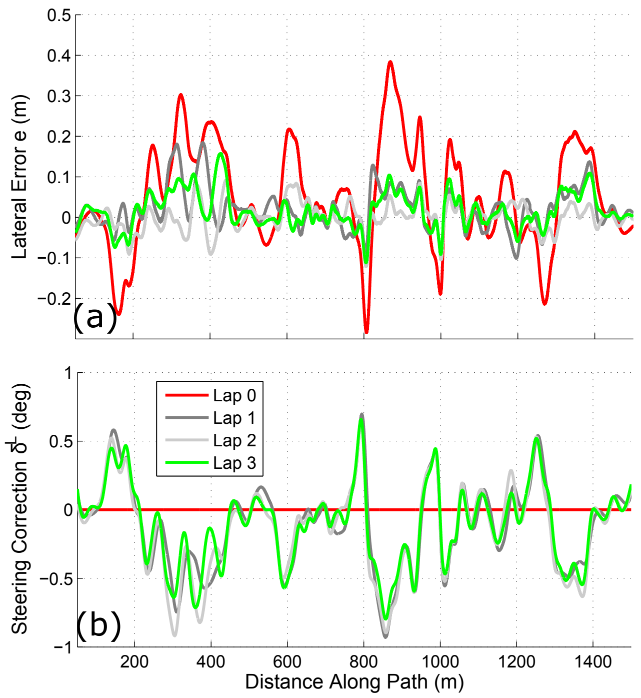

Fig. 9. Experimental results for path tracking error with Q-SISO learning controller, at peak lateral accelerations of 8 $\mathbf{m/s^2}$.

Experimental results in this section were only given for the quadratically optimal controller with decoupled (SISO) dynamics. The PD iterative learning controller was not tested due to the relatively worse simulation performance, and the quadratically optimal controller with coupled dynamics provided no clear benefit in simulation but a much longer computation time.

III. ITERATIVE TRAJECTORY MODIFICATION

A. Effect of Tire Slip Norm on Lap Time

Fig. 12 shows the complicated effect of driving at different levels of lateral and longitudinal acceleration for region $\textcircled{2}$ in Fig. 2. Recall that the speed profile is generated by assuming a global value of tire friction μ, which is directly related to the acceleration norm of the desired speed profile ($\sqrt{a_x^2 + a_y^2}$) by a factor of $g = 9.81\ \mathrm{m/s^2}$. Fig. 12(a) confirms that higher levels of μ result in strictly faster speed profiles for the same turn.

However, selecting a more aggressive value of μ does not necessarily entail a faster experimental lap time. Consider the actual

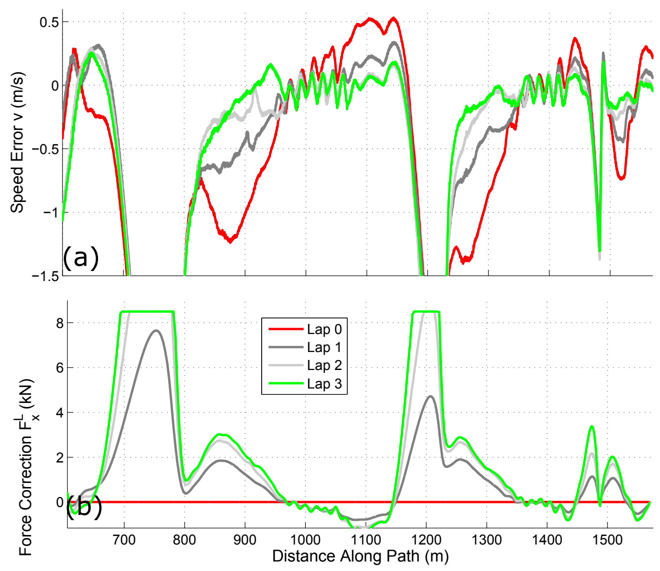

Fig. 10. Experimental results for speed tracking error with Q-SISO learning controller, at peak lateral accelerations of 8.5 $\mathbf{m/s^2}$.

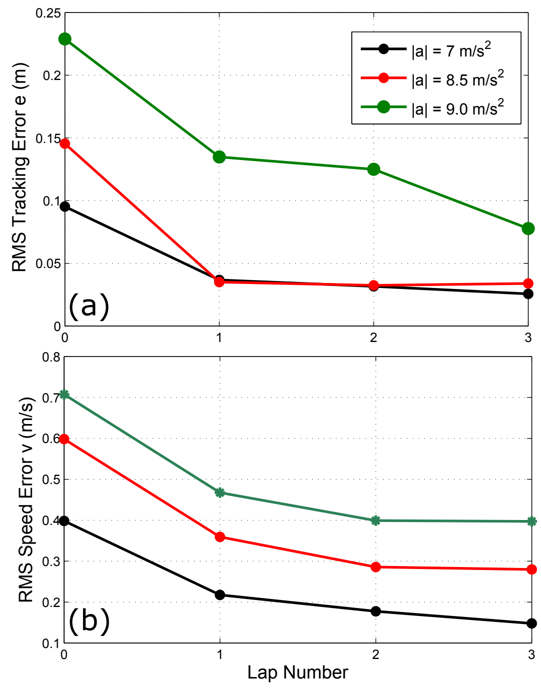

Fig. 11. Experimental RMS tracking error for the Q-SISO learning controller at several levels of lateral acceleration. (a) RMS lateral tracking error as a function of lap number/iteration number. (b) RMS speed tracking error as a function of lap/iteration number. Note that lap 0 corresponds to the baseline case where iterative learning control is not applied.

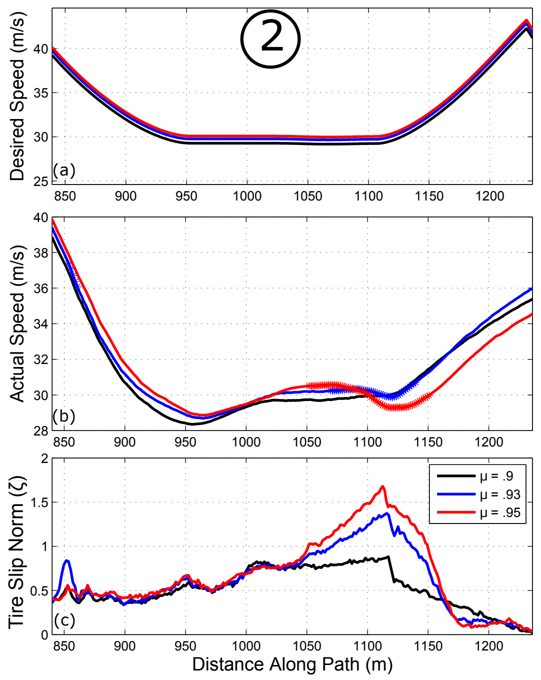

Fig. 12. (a) Desired speed for varying levels of $\boldsymbol{\mu}$ (b) Actual speed for varying levels of $\boldsymbol{\mu}$. Asterisks correspond to regions where $\boldsymbol{\zeta > 1}$. (c) Tire slip norm for varying levels of $\boldsymbol{\mu}$.

speeds of the vehicle in Fig. 12(b) when trying to experimentally follow three different speed profiles. For the case where $\mu = 0.9$, the vehicle completes the turn without fully utilizing the tire's capability and achieves relatively low velocities. For the extreme case where $\mu = 0.95$, the vehicle enters the turn at a high speed but then begins to slide as $\zeta = 1.6$. While not shown in the plot, the saturation is occurring primarily at the front tires, causing an understeer that can cause the car to skid off the track. Completing the lap therefore requires a stabilizing action from the stability controller to slow the car down and regain control of the vehicle. As a result, when the vehicle accelerates at the end of the turn, the actual vehicle speed at $\mu = 0.95$ is significantly slower than the case where $\mu = 0.9$! A final "just right" value of $\mu = 0.93$ was also tested experimentally for this turn, and while the car does slide a bit at this level of driving (peak $\zeta = 1.3$), the needed stabilizing action is significantly smaller and the vehicle exits the turn with the highest speed.

Unfortunately, the best value of μ is not constant throughout the track. Fig. 13 shows the same data plotted for region $\textcircled{3}$ of the Thunderhill Raceway. In this case, even with an atypically high value of $\mu = 0.97$, the slip norm of the vehicle tires is relatively low, and the value of μ that optimizes the completion time for this turn could be even larger.

B. Naive Method: Greedy Algorithm

Section III-A shows that to minimize the overall lap time, there is a need to generalize the speed profile such that different portions of the track can be driven with different values of μ. In other words, **the problem is to learn the "friction profile" $\mu^\star(s)$ along the path that minimizes the experimental vehicle lap time.**

The simplest approach to finding $\mu(s)$ is a greedy algorithm where a set of experimental data is collected for a variety of different speed profiles, each corresponding to a different value of μ and therefore a different acceleration level. The greedy approach is then to discretize the track into a number of small sections and pick the value of μ for

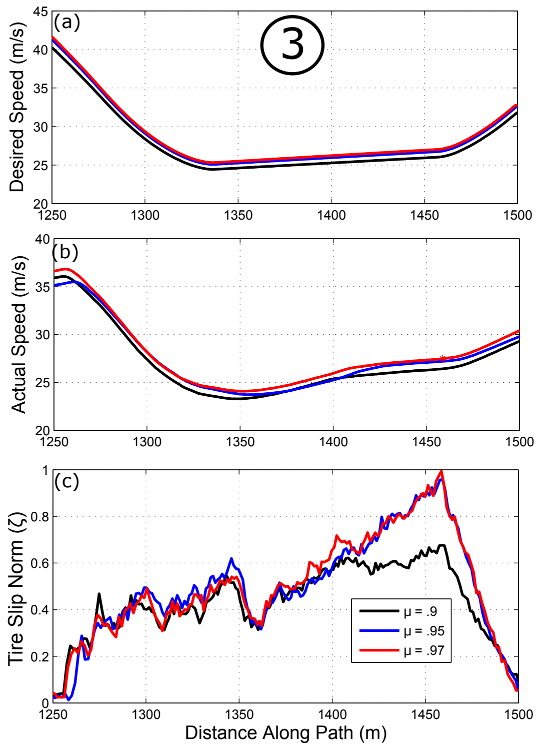

Fig. 13. (a) Desired speed for varying levels of $\boldsymbol{\mu}$. (b) Actual speed for varying levels of $\boldsymbol{\mu}$. Asterisks correspond to regions where $\boldsymbol{\zeta > 1}$. (c) Tire slip norm for varying levels of $\boldsymbol{\mu}$.

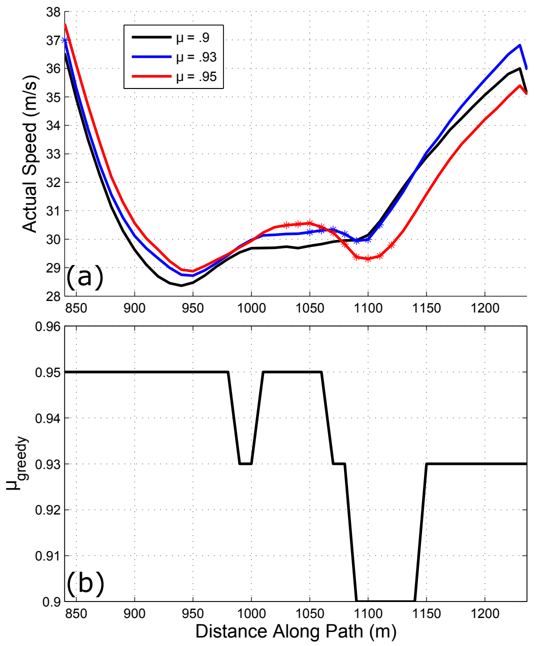

Fig. 14. (a) Desired speed for varying levels of $\boldsymbol{\mu}$ (b)"Greedy" value of $\boldsymbol{\mu}$ as a function of distance along track. Asterisks denote region of track where vehicle is understeering.

TABLE II
$\boldsymbol{U_k(\mu)}$ AND $\boldsymbol{Z_k(\mu)}$ FOR $\boldsymbol{k} = 191$

μ	$U_x(m/s)$	ζ
0.9	28.42	0.55
0.93	28.91	0.63
0.95	29.14	0.66

each section that corresponds to the highest observed experimental velocity. The final desired velocity profile is then generated using the numerical integration approach presented in [32]. This method does not require a single value of friction across the whole track, and generates a smooth velocity profile even when the peak acceleration limits vary from point to point.

A plot showing the results of applying the greedy algorithm for section ② is shown in Fig. 14. The flaw in simply selecting μ based on the highest speeds is apparent at $s = 1050$m. The greedy algorithm suggests the vehicle mostly drive at $\mu = 0.95$, but then switch to driving at $\mu = 0.93$ as soon as the vehicle begins to slide. Switching to a less aggressive velocity profile at this point is impossible to achieve in practice, because the vehicle is already fully sliding and has no control until the vehicle slows down. As a result, the greedy algorithm fails to capture the hidden cost of a large understeer, which results in a period of time where the speed must inevitably drop.

C. Framing Trajectory Learning as Search Problem

Given the inadequacy of the greedy algorithm, a more sophisticated approach is necessary to learn $\mu^\star(s)$ from experimentally observed data. This section frames the desire to find the minimum time $\mu^\star(s)$ as a tree search problem. Consider discretizing the racing path into N evenly spaced segments. For example, on the Thunderhill Raceway with $\Delta s = 5$m, $\mathbf{s} = [0 \;\; 5 \ldots s_k \ldots 4495 \;\; 4500]$ for $k = 0 \ldots N$, with $N = 901$. For each path distance s_k, there are M_k velocity and M_k tire slip observations from experimental data, each corresponding to a different μ. For example, looking at Fig. 12, if $k = 191$, $s_k = 950$m, $M_k = 3$, the velocity and slip norm ζ observations $U_k(\mu)$ and $Z_k(\mu)$ are as follows:

M_k is not necessarily the same for all k to account for experimental trials that do not cover the full lap. For safety and time reasons, some parts of the track will only have experimental data collected at two different friction values, while others may have five or six.

Nodes of the search tree are then defined as a two-element state tuple, with the first state element being s_k and the second element being the current friction coefficient μ_k. Since the car must start from the beginning of the path, the first state has $s_0 = 0$. The M_k edges from a given node correspond to actions that the car can take at $s = s_k$. In this case, the "action" is the next value of μ, and the successor states are $(s_{k+1}, \mu_1)...(s_{k+1}, \mu_{M_k})$. A diagram of the search tree is shown in Fig. 15.

Each edge is associated with a travel cost c_t. The travel cost for a given edge is the amount of time it takes to go from $s = s_k$ to $s = s_{k+1}$ driving at the friction coefficient μ associated with that edge. As illustrated in Fig. 16, the cost is estimated from the experimentally observed data assuming linear acceleration between points. The travel cost from node (s_k, μ_k) to (s_{k+1}, μ_{k+1}) can therefore be expressed mathematically with trapezoidal integration of the straight-line velocity profile:

$$c_t(s_k, s_{k+1}, \mu_k, \mu_{k+1}) = \ln\frac{a_x(\Delta s) + U_k(\mu_k)}{a_x} - \ln\frac{U_k(\mu_k)}{a_x} \tag{43}$$

$$a_x = \frac{U_{k+1}(\mu_{k+1}) - U_k(\mu_k)}{\Delta s} \tag{44}$$

In addition to the travel cost, there is also a *switching* cost c_s associated with switching to a different value of μ. These are determined by the observed tire slip norm measurements $Z_k(\mu)$. The switching cost is expressed mathematically as:

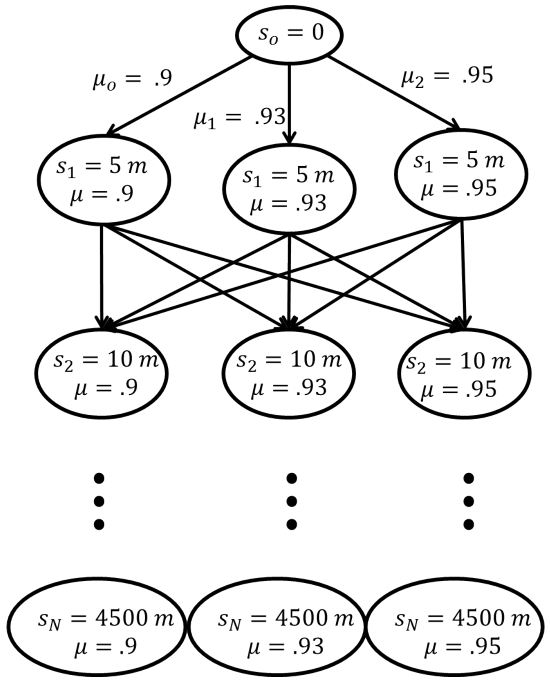

Fig. 15. Sample search tree for Thunderhill race track where there are only 3 experimentally observed full laps at $\boldsymbol{\mu} = 0.9, 0.93, 0.95$.

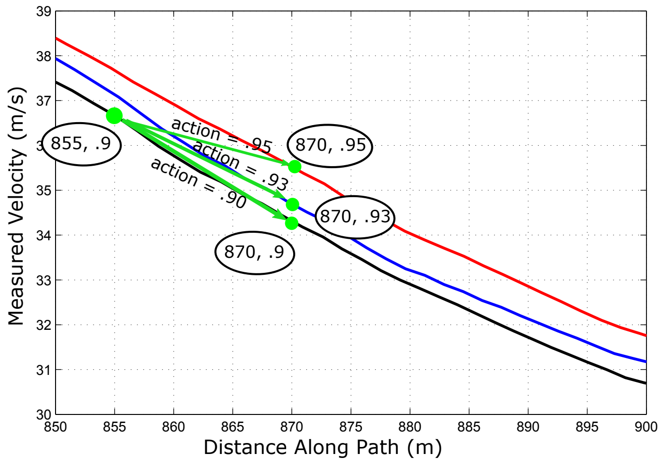

Fig. 16. Illustration of travel cost. Current state is (855, 0.9). Assuming $\boldsymbol{\Delta s}$ is 15 meters, cost is time to travel from this node to one of the three possible successor nodes, depending on the action taken.

$$c_s(s_k, s_{k+1}, \mu_k, \mu_{k+1}) = \tag{45}$$

$$\mathbb{1}\left(\mu_{k+1} \neq \mu_k\right)\left(\lambda + \infty \times \mathbb{1}\left(Z_k(\mu_k) > 1\right)\right) \tag{46}$$

Where $\mathbb{1}$ is the indicator function. The switching cost function (45) implies that an action will never incur a switching cost if the selected value of μ is unchanged from the previous selection. If the value of μ does change, there is a small switching penalty λ, chosen by trial and error to discourage the search algorithm from changing the friction profile to gain a trivial decrease in lap time. Additionally, there is a very large (infinite) switching penalty if the search algorithm attempts to change the friction profile while the vehicle's tires are saturated ($\zeta > 1$). This reflects the physical inability of the car to control its trajectory while sliding and is what separates the search algorithm

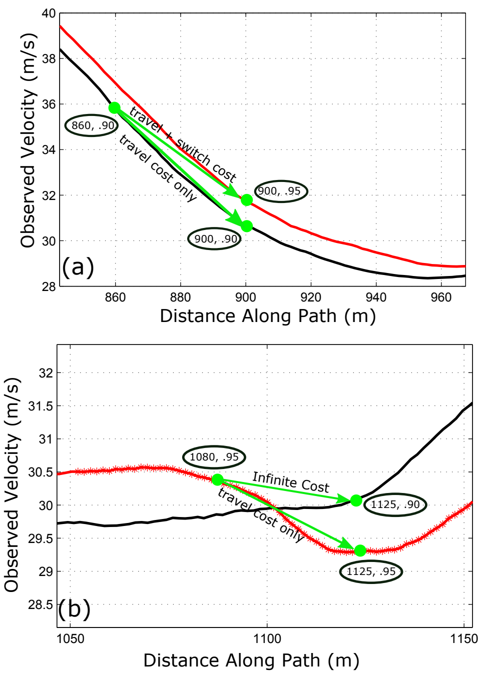

Fig. 17. (a) Costs when vehicle currently is not sliding. The vehicle can switch to a more aggressive velocity profile, paying a small switching penalty, or can continue on the current profile. (b) Costs when vehicle is currently sliding. Vehicle has no choice but to continue on current trajectory. Asterisks denote regions where $\boldsymbol{\zeta > 1}$ and the vehicle is sliding.

from the greedy algorithm. A diagram demonstrating (45) is shown in Fig. 17.

D. A Search Algorithm and Heuristic*

Once the tree is mathematically defined in terms of the start state, nodes, edges, and costs, the search problem is to find the sequence of actions from the start node to any terminal node that minimizes the total cost. In our case, the start state, nodes, edges and costs were defined in the previous section, and a terminal node is any node at the end of the path (i.e. $k = N$). The sequence of actions in our case is the friction profile $\mu = [\mu_1 \ldots \mu_k \ldots \mu_N]$, which determines how aggressively the vehicle will drive on every part of the track. The total cost is the sum of all individual travel costs c_t and switching costs c_s, and has intuitive units of time.

Minimum-cost tree spanning algorithms (e.g. breadth-first search, Dijkstra's algorithm, etc.) are a well known subject and a thorough description can be found in [33]. For the purpose of solving this search problem, the A* search algorithm is used. Like breadth-first search, the A* algorithm is guaranteed to find the lowest cost path from a start node to a goal node, but uses a priority queue data structure to more efficiently explore the search tree. Frontier leaf nodes n being considered for exploration are ranked according to the following modified cost:

$$f(n) = g(n) + h(n) \tag{47}$$

where $g(n)$ is the true cost to go from the start node to node n. The function $h(n)$ is a heuristic estimate of the cost to get from node n to any goal state (i.e. to the end of the path). For A* to be guaranteed to find the shortest path, $h(n)$ must be *admissible*, meaning that $h(n)$ must underestimate the true cost of getting to the end of the path.

In our case, we have a very intuitive heuristic function $h(n)$ for the A* implementation. In Sec. III-B, the greedy approach was discussed.

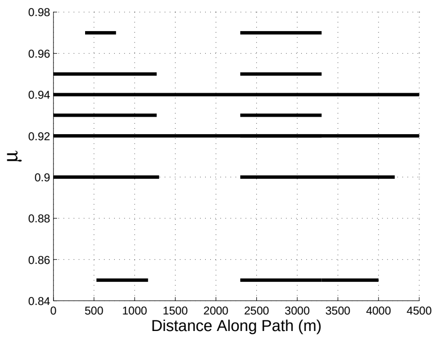

Fig. 18. Coverage of experimental data. For speed profiles corresponding to $\mu = 0.92$ and $\mu = 0.94$, experimental data of the autonomous race vehicle was observed over the whole track. For safety and time constraints, the other speed profiles were only tested on sections of the track.

Define $\mathbf{U}_{\rm g} = [U_{\rm g}(1) \dots U_{\rm g}(k) \dots U_{\rm g}(N)]$ to be the highest observed experimental speed for each index s_k. Because we know tracking this greedy profile is physically impossible due to vehicle sliding, time estimates from this profile will always underestimate the true cost. We therefore define $h(n)$ as follows:

$$h(n) = \int_{s_n}^{s_N} \frac{1}{U_g(s)} ds \tag{48}$$

where s_n is the value of s corresponding to node n and s_N is the total length of the track. While (48) is an integral equation, it can be solved for the discrete array $\mathbf{U}_{\rm g}$ via trapezoidal numerical integration.

E. A* Implementation and Results

Because the A* algorithm relies on experimental observations to learn $\mu^\star(s)$, experimental data was collected over several trials, with each trial consisting of a speed profile generated with a constant μ value over the track. The μ values chosen for experimental data collection were 0.85, 0.9, 0.92, 0.93, 0.94, 0.95, and 0.97. Ideally, each speed profile would be tested experimentally for a full lap at high speed. However, due to safety and time constraints associated with collecting high speed race data, only the speed profiles corresponding to $\mu = 0.92$ and $\mu = 0.94$ were observed over the whole track. The other speed profiles were only tested on sections of the track. Fig. 18 shows the experimental data coverage.

TABLE III
SEARCH ALGORITHM INFORMATION

Parameter	Symbol	Value	Units
Track Length	L	4500	m
Discretization Length	Δs	5	m
Number of Points	N	901	-
Switching Cost	λ	0.05	s
CPU Solution Time		70	s
Nodes Explored		6887	-

After collecting the experimental data, the A* algorithm was applied to learn the optimal friction profile $\mu^\star(s)$. Parameters for the algorithm are shown in Table III. To interface more efficiently with experimental data files, the search algorithm was implemented in MATLAB using custom code instead of a standard search algorithm library. The entire search process took approximately 70 seconds on a core i7 laptop machine, exploring 6887 nodes in the process. However, since MATLAB is not designed for computational efficiency in tree-based search algorithms, a C++ implementation would likely be several orders of magnitude faster.

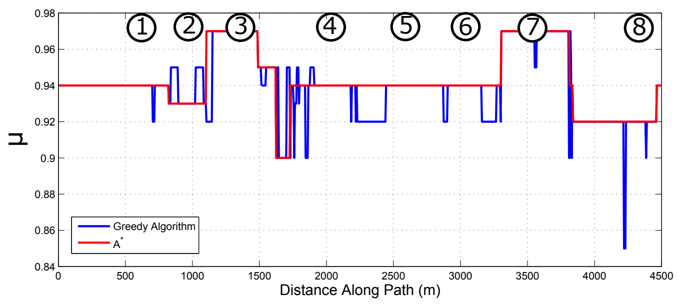

Fig. 19. Minimum time $\mu(s)$ profile for Thunderhill, for both the A* solution and greedy algorithm solution.

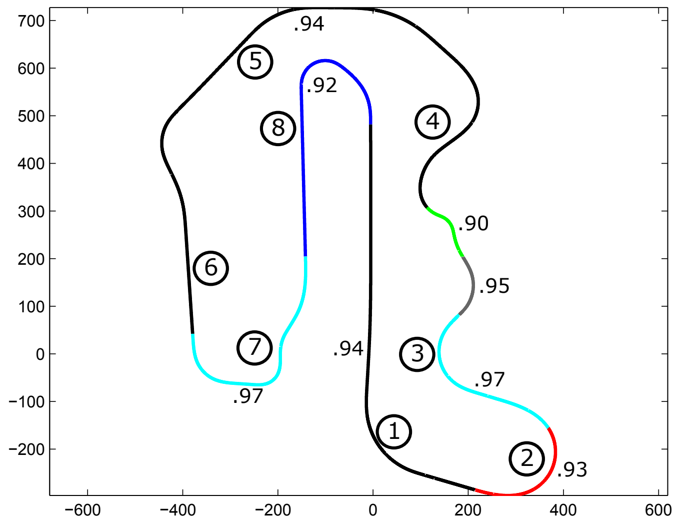

Fig. 20. Minimum time $\mu^\star(s)$ profile from A* algorithm plotted on map of Thunderhill.

The resulting $\mu^\star(s)$ profile associated with the minimum lap time on Thunderhill Raceway is shown in Fig. 19. For comparison, the A* solution is plotted against the greedy algorithm solution. Because of the incorporation of switching costs, the A* algorithm switches μ values only when necessary to achieve a nontrivial increase in lap time, and only switches μ when the vehicle is not sliding. The same results are plotted on a map of the track in Fig. 20. A satisfying observation is that the optimal profile reduces μ to 0.93 in section ② to be more conservative and increases μ to 0.97 in section ③ to be more aggressive. This matches our observations about tire slip norm ζ in Sec. III-A.

Predicted lap time results are shown in Table IV. Notice that the A* predicted lap time is slightly slower than the greedy algorithm prediction, which is expected given the physical infeasability of the greedy algorithm assumptions. The results also indicate that a significant lap time improvement can be expected over a velocity profile generated with a constant μ.

TABLE IV
LAP TIMES

Driver	Lap Time
Constant $\mu(s)$ = 0.94	139.2
Constant $\mu(s)$ = 0.92	139.4
Greedy $\mu_g(s)$ (Prediction)	136.4
A* $\mu^*(s)$ (Prediction)	136.8

Fig. 20 provides interesting insights about what the A* algorithm

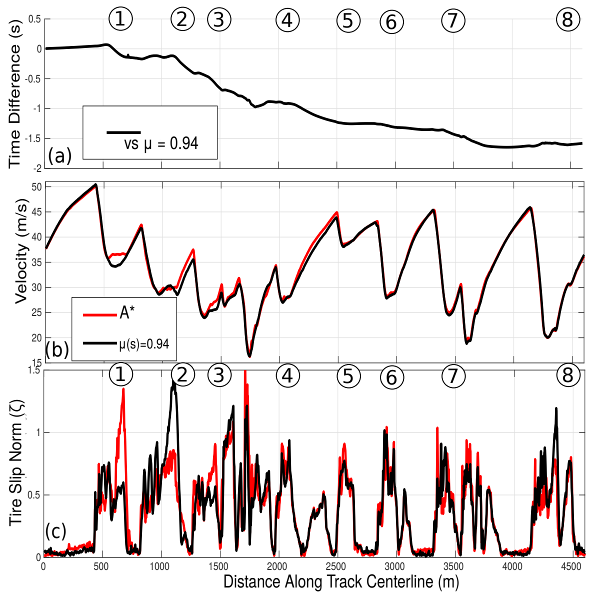

Fig. 21. Time difference between experimental dataset collected with A* result $\boldsymbol{\mu^*(s)}$ and constant friction velocity profile at $\boldsymbol{\mu = 0.94}$. Negative time distance corresponds to A* result being ahead. (b) Experimental velocities between both experimental datasets. (c) Tire slip norm measurements for both datasets.

is learning. Section ② is a long, sweeping turn with mostly steady-state cornering dynamics, so the optimal friction value of .93 is in the middle of the range of possible values and representative of the average friction between the tires and the track. Sections ③ and ⑦ represent short turns followed by a turn in the opposite direction. For these turns, the algorithm has learned it is better to drive a little faster than the true friction limit would dictate, because by the time the vehicle begins to understeer, the turn is already complete and the vehicle can reverse the steering quickly while following the desired path. Section ⑧ occurs before a long straight section of the track where recovering from an understeer would result in a significantly lower speed on the fastest part of the track. As a result, the algorithm's planned acceleration is more conservative. Finally, the section with the lowest $\mu(s)$ occurs on a part of the track with significant lateral weight transfer. In general, lateral weight transfer reduces the cornering forces available to the vehicle, but this effect was not captured in the trajectory planning phase, which assumed a planar model with coupled left and right tires. In summary, the A* algorithm allows the car to learn subtle but important driving behaviors that are not easily captured through simulation.

F. Experimental Validation

The best validation of the A* algorithm is to experimentally drive the optimal velocity profile $U_x^\star(s)$ generated from $\mu^\star(s)$. From Fig. 19, this velocity profile will have accelerations as low as 9.0 $\mathrm{m/s}^2$ on some turns, and as high as 9.7 $\mathrm{m/s}$ on others. Fig. 21 shows autonomous experimental data from driving $\mu^\star(s)$ compared to one other experiment[1]. The comparison experiment is a fully autonomous test with a constant $\mu = 0.94$.

The experimental results show solid performance of the learned friction profile $\mu^\star(s)$. From Fig. 21(a), the lap time using the learned friction profile is roughly 1.5 seconds faster than the lap time from the constant friction profile. A significant part of this improved performance comes through more efficient friction usage. For example, in sections ② and ⑧, learning to drive more cautiously enables the tire slip norm ζ to drop closer to 1, avoiding a costly understeer. In sections ③ and ⑦, the A* algorithm has learned that more aggressive driving is possible, increasing ζ closer to 1.[2]

IV. Conclusion

This paper presents two complementary approaches to gradually improve the driving performance of an autonomous race car over time. Iterative learning control (ILC) formulations gradually determine the proper steering and throttle inputs to precisely track the desired racing trajectory. Experimental results at combined vehicle accelerations of 9 $\mathrm{m/s}^2$ indicate that the proposed algorithm can rapidly attenuate trajectory following errors over just two or three laps of racing (§II-F). This paper also presents a tree-search algorithm to minimize experimental lap times by learning different acceleration limits for each turn on the track. An A* search algorithm is devised to search through experimental data and find the best value of μ for each portion of the track in order to globally minimize the resulting lap time. Key developments of this algorithm include designing an appropriate A* heuristic to minimize the needed computation time and designing the cost function to account for the physical difficulty of altering the vehicle's trajectory while understeering or oversteering.

Acknowledgment

This research is supported by the National Science Foun- dation Graduate Research Fellowship Program (GRFP). The authors would like to thank the members of the Dynamic Design Lab at Stanford University and the Audi Electronics Research Lab.

References

[1] D. Casanova, "On minimum time vehicle manoeuvring: The theoretical optimal lap," Ph.D. dissertation, Cranfield University, 2000.

[2] D. P. Kelly, "Lap time simulation with transient vehicle and tyre dynamics," Ph.D. dissertation, Cranfield University, 2008.

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

[4] N. R. Kapania, J. Subosits, and J. C. Gerdes, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," in *ASME 2015 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2015.

[5] K. Kritayakirana and J. C. Gerdes, "Using the center of percussion to design a steering control for an autonomous racecar," *Vehicle System Dynamics*, vol. 50, pp. 33–51, 2012.

[6] S. Muller, M. Uchanski, and K. Hedrick, "Estimation of the maximum tire-road friction coefficient," *Journal of Dynamic Systems, Measurement, and Control*, vol. 125, no. 4, pp. 607–617, 2003.

[7] K. Kritayakirana and J. C. Gerdes, "Autonomous vehicle control at the limits of handling," *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 4, pp. 271–296, 2012.

[8] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W. Zhang, D. H. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, "Automated vehicle control developments in the PATH program," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 114–130, 1991.

[9] M. Nagai, H. Mouri, and P. Raksincharoensak, "Vehicle lane-tracking control with steering torque input," *Vehicle System Dynamics Supplement*, vol. 37, pp. 267–278, 2002.

[1]There was a minor change between the optimal μ profile from Fig. 19 and what was tested experimentally. For section ②, the value of μ was set to 0.92 as opposed to 0.93. This was a safety measure taken based on preliminary testing.

[2]Note that due to time constraints, both datasets were all taken on different dates, meaning some variation in weather and tire conditions is possible for each test.

[10] N. R. Kapania and J. C. Gerdes, “Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling,” *Vehicle System Dynamics*, vol. 53, no. 12, pp. 1687–1704, 2015.

[11] K. Kritayakirana, “Autonomous vehicle control at the limits of handling,” Ph.D. dissertation, Stanford University, 2012.

[12] D. A. Bristow, M. Tharayil, and A. G. Alleyne, “A survey of iterative learning control,” *IEEE Control Systems*, vol. 26, no. 3, pp. 96–114, 2006.

[13] D.-I. Kim and S. Kim, “An iterative learning control method with application for CNC machine tools,” *IEEE Transactions on Industry Applications*, vol. 32, no. 1, pp. 66–72, 1996.

[14] C. T. Freeman and Y. Tan, “Iterative learning control with mixed constraints for point-to-point tracking,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 604–616, 2013.

[15] L. Hladowski, K. Galkowski, Z. Cai, E. Rogers, C. T. Freeman, and P. L. Lewin, “Experimentally supported 2D systems based iterative learning control law design for error convergence and performance,” *Control Engineering Practice*, vol. 18, no. 4, pp. 339–348, 2010.

[16] D. Huang, J.-X. Xu, V. Venkataramanan, and T. Huynh, “High-performance tracking of piezoelectric positioning stage using current-cycle iterative learning control with gain scheduling,” *IEEE Trans. on Industrial Electronics*, vol. 61, no. 2, pp. 1085–1098, 2014.

[17] A. Mohammadpour, S. Mishra, and L. Parsa, “Iterative learning control for fault-tolerance in multi-phase permanent-magnet machines,” in *American Control Conference (ACC)*, 2013, pp. 5929–5934.

[18] D. J. Hoelzle, A. G. Alleyne, and A. J. W. Johnson, “Basis task approach to iterative learning control with applications to micro-robotic deposition,” *IEEE Trans. on Control Systems Technology*, vol. 19, no. 5, pp. 1138–1148, 2011.

[19] C. Chen, Y. Jia, J. Du, and F. Yu, “Lane keeping control for autonomous 4WS4WD vehicles subject to wheel slip constraint,” in *American Control Conference (ACC)*, 2012, pp. 6515–6520.

[20] O. Purwin and R. DAndrea, “Performing and extending aggressive maneuvers using iterative learning control,” *Robotics and Autonomous Systems*, vol. 59, no. 1, pp. 1–11, 2011.

[21] H. Sun, Z. Hou, and D. Li, “Coordinated iterative learning control schemes for train trajectory tracking with overspeed protection,” *IEEE Trans. on Automation Science and Engineering*, vol. 10, no. 2, pp. 323–333, 2013.

[22] C. Slepicka and C. R. Koch, “Iterative learning on dual-fuel control of homogeneous charge compression ignition,” *IFAC-PapersOnLine*, vol. 49, no. 11, pp. 347–352, 2016.

[23] X.-z. Xu, Y.-k. Gao, and W.-g. Zhang, “Iterative learning control of a nonlinear aeroelastic system despite gust load,” *International Journal of Aerospace Engineering*, vol. 2015, 2015.

[24] P. A. Theodosis, “Path planning for an automated vehicle using professional racecar driving techniques,” Ph.D. dissertation, Stanford University, 2014.

[25] M. Klomp, M. Lidberg, and T. J. Gordon, “On optimal recovery from terminal understeer,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 2014.

[26] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, “Safe driving envelopes for path tracking in autonomous vehicles,” *Control Engineering Practice*, vol. 61, pp. 307–316, 2017.

[27] U. Rosolia, A. Carvalho, and F. Borrelli, “Autonomous racing using learning model predictive control,” in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 5115–5120.

[28] E. J. Rossetter and J. C. Gerdes, “A study of lateral vehicle control under a virtual force framework,” in *Proceedings of the International Symposium on Advanced Vehicle Control*, 2002.

[29] K. L. Talvala and J. C. Gerdes, “Lanekeeping at the limits of handling: Stability via Lyapunov functions and a comparison with stability control,” in *Dynamic Systems and Control Conference*, 2008, pp. 361–368.

[30] R. Y. Hindiyeh and J. C. Gerdes, “A controller framework for autonomous drifting: Design, stability, and experimental validation,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 136, no. 5, p. 051015, 2014.

[31] D. A. Bristow and B. Hencey, “A Q,L factorization of norm-optimal iterative learning control,” in *47th IEEE Conference on Decision and Control (CDC)*, 2008, pp. 2380–2384.

[32] J. Subosits and J. C. Gerdes, “Autonomous vehicle control for emergency maneuvers: The effect of topography,” in *American Control Conference (ACC), 2015*. IEEE, 2015, pp. 1405–1410.

[33] R. Stuart and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.