AUTONOMOUS VEHICLES

Neural network vehicle models for high-performance automated driving

Nathan A. Spielberg*, Matthew Brown, Nitin R. Kapania, John C. Kegelman, J. Christian Gerdes

Automated vehicles navigate through their environment by first planning and subsequently following a safe trajectory. To prove safer than human beings, they must ultimately perform these tasks as well or better than human drivers across a broad range of conditions and in critical situations. We show that a feedforward-feedback control structure incorporating a simple physics-based model can be used to track a path up to the friction limits of the vehicle with performance comparable with a champion amateur race car driver. The key is having the appropriate model. Although physics-based models are useful in their transparency and intuition, they require explicit characterization around a single operating point and fail to make use of the wealth of vehicle data generated by autonomous vehicles. To circumvent these limitations, we propose a neural network structure using a sequence of past states and inputs motivated by the physical model. The neural network achieved better performance than the physical model when implemented in the same feedforward-feedback control architecture on an experimental vehicle. More notably, when trained on a combination of data from dry roads and snow, the model was able to make appropriate predictions for the road surface on which the vehicle was traveling without the need for explicit road friction estimation. These findings suggest that the network structure merits further investigation as the basis for model-based control of automated vehicles over their full operating range.

INTRODUCTION

Autonomous vehicles promise to revolutionize human mobility and vehicle safety. This promise stems from eliminating the need for a human driver, markedly reducing the cost of mobility and, ideally, eliminating the 94% of crashes attributable to human recognition, decision, or performance error (*1*). This, in turn, requires automated vehicles capable of navigating safely through the world more skillfully than a human driver. Many automated vehicle designs incorporate a common architecture of generating a safe, collision-free trajectory through the environment with a high-level planning layer and subsequent tracking of this trajectory using lower-level controllers (*2*–*5*). Navigating safely, therefore, requires not only the planning of collision-free trajectories but also the ability to tightly track the desired path, ideally within tens of centimeters. Furthermore, as deployment of automated vehicles expands, these path-following controllers must handle a wide variety of conditions, including safely navigating icy roads when friction is reduced or emergency collision-avoidance maneuvers when the need arises. Each of these cases is an example of everyday driving in which operation at the vehicle's limits becomes critical.

Although there has been substantial work developing control techniques for automated vehicles, much of this effort has focused on controlling the vehicle under normal driving conditions with gentle maneuvers on high-friction, dry road surfaces (*6*, *7*). Research on controlling vehicles in more critical maneuvers near the friction limits has illuminated many associated challenges (*8*–*11*). Fundamentally, as the vehicle approaches the limits of tire-road friction, it becomes either unstable (if the rear tire reaches the limits) or uncontrollable (if the front tire reaches the limits). To track a path at the limits of the vehicle's capabilities, some estimate of the vehicle's road-tire friction coefficient is necessary for both trajectory design and determining the appropriate steering commands. Obtaining such estimates is challenging in general (*12*) and complicated further by the facts that tire-road friction often quickly changes and portions of the road may consist of different surfaces. Aside from difficulty in estimating this critical parameter, developing an accurate dynamic model that can be used for trajectory generation and following at the limits is a difficult task because the equations of motion become highly nonlinear. The designer must further choose the appropriate level of fidelity, deciding whether or not to include effects such as weight transfer across tires due to acceleration or the lag in tire force generation with rapid steering movement.

As challenging as the limits of handling are for the control system designer, they are even more challenging for the average driver and a substantial factor in many crashes. Yet, more experienced drivers, particularly those with racing experience even at an amateur level, have the ability to safely control the vehicle at the full limits of its capabilities (*13*). In racing, this is demonstrated by producing low and consistent lap times, but, in a critical maneuvering situation, this capability means the ability to harness all of the tire-road friction to avoid a crash. If we want automated vehicles that can maneuver better than an experienced driver in critical situations, the bar on the performance of a tracking controller must be set rather high.

Here, we show that a simple path-tracking architecture can enable an automated vehicle to track a path accurately while using the tire-road friction as fully as a champion amateur race car driver. The key is an appropriate model. Using a physics-based dynamics model for feedforward control, a simple linear feedback controller, and a trajectory designed from the vehicle's modeled friction limits, the car could drive with mean path-tracking errors below 40 cm at the modeled friction limits. Because the model represents only an estimate of what the true limits are, we benchmarked our automated vehicle performance against a champion amateur race driver and compared lap times over segments on the track. This novel comparison demonstrates that the controller's operation at the modeled friction limits is comparable in friction utilization with the real-world ability of an experienced race car driver.

To achieve this level of performance, of course, the simple physics-based model must be well characterized for the particular condition of

Department of Mechanical Engineering, Stanford University, Building 530, 440 Escondido Mall, Stanford, CA 94305, USA.
*Corresponding author. Email: nspielbe@stanford.edu

driving on this dry high-friction race course. The question of how comparable models could be developed for on-road automated vehicles then naturally arises. Although obtaining parameters for different vehicles would be feasible as part of a typical development process, several parameters vary strongly with the road condition. Although researchers have demonstrated online parameter estimation that can adapt to changing road conditions (*14*–*16*), such techniques have not matured to the point of commercial deployment in automobiles or the performance required of a safety-critical system. In addition, real-time estimation does not leverage the vast quantity of data that current vehicles generate, and future automated vehicles will conceivably be able to share. Nor does it address the question of model fidelity given that parameter estimation grows more challenging as the model complexity increases. Ideally, the model generation process should be capable of leveraging data about surfaces with varying levels of friction and reduce the number of a priori modeling decisions while still capturing the accuracy and performance of physics-based models tuned for specific conditions.

These requirements motivated investigating neural network models for vehicle control. Because of their universal function approximation properties, neural network models have shown numerous recent achievements, such as benchmarks in image recognition and mastering the game of Go (*17*–*19*). Early research in neural network models illustrated that these models are capable of vehicle control and dynamic model identification (*20*, *21*). Neural network vehicle models have shown success in numerous robotics applications from quadcopter control to control of scale rally racing vehicles (*22*, *23*). These models have been successfully used for vehicle dynamic model identification but have yet to be used to capture changing vehicle dynamics from driving at the limits on multiple friction surfaces (*24*, *25*). Additionally, neural network models can use history information to capture time-varying or higher-order effects, as shown in both model helicopter and robotics applications (*26*–*28*).

We present a feasibility study in which we used the states and inputs of the physics-based model as guidance to develop a two-layer feedforward neural network capable of learning vehicle dynamic behavior on a range of different surfaces. The network involves a combination of current measurements and history information from three previous time steps. The history information enabled the network to provide predictions of behavior at different friction levels without the need for an explicit friction estimation scheme. When trained on a combination of high- and low-friction data, the model made predictions appropriate to the surface described by the history information. By foregoing the step of friction estimation, the neural network with history information fused estimation and prediction capabilities, simplifying the task of vehicle control. This additional functionality did not come at the expense of performance. We show increased path tracking performance at the limits when compared with the tuned physics-based model. Furthermore, a simulation study demonstrates that the neural network model could capture a range of dynamic behaviors that were not included in the simple physics-based model.

RESULTS

To investigate the performance of a path tracking architecture at the limits of a vehicle's handling capabilities, we designed an experimental comparison with an experienced human race driver. An appropriate benchmark for our automated vehicle in this case is a proficient human driver who has substantial driving and amateur racing experience and extensive familiarity with the test course. In this experiment, we used a physics-based feedforward-feedback controller (as shown in Fig. 1) implemented on an automated 2009 Audi TTS (Shelley) (Fig. 2A) (*29*). The controller tracked a desired path while a separate controller matched the desired vehicle speed through brake, throttle, and gear shift commands. The path and speed profile were designed through optimization techniques to minimize the time required to drive the track based on the vehicle model (*30*).

The model used for feedback-feedforward control generates an appropriate steer angle to apply for a given path curvature and vehicle longitudinal speed. The accuracy of this input has a substantial effect on both the resulting path tracking error and the required feedback effort. The feedforward steering command here was derived from the equations of motion for a planar single-track or "bicycle" model, a commonly used model in the vehicle dynamics community derived from Newtonian physics. For this paper, mention of the "physics-based model" explicitly refers to the planar bicycle model. To calculate a feedforward steering input from these equations of motion, we used steady-state operating conditions to determine feedforward tire forces.

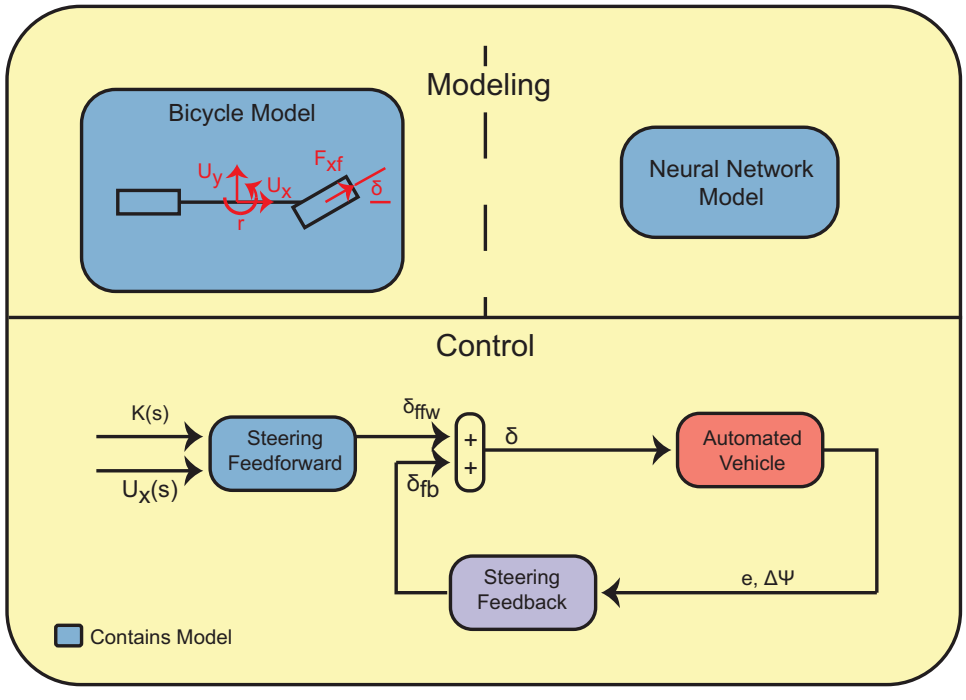

Fig. 1. Simple feedforward-feedback control structure used for path tracking on an automated vehicle. Possible models for use in generating feedforward steering commands consist of the physics-based model or a neural network model.

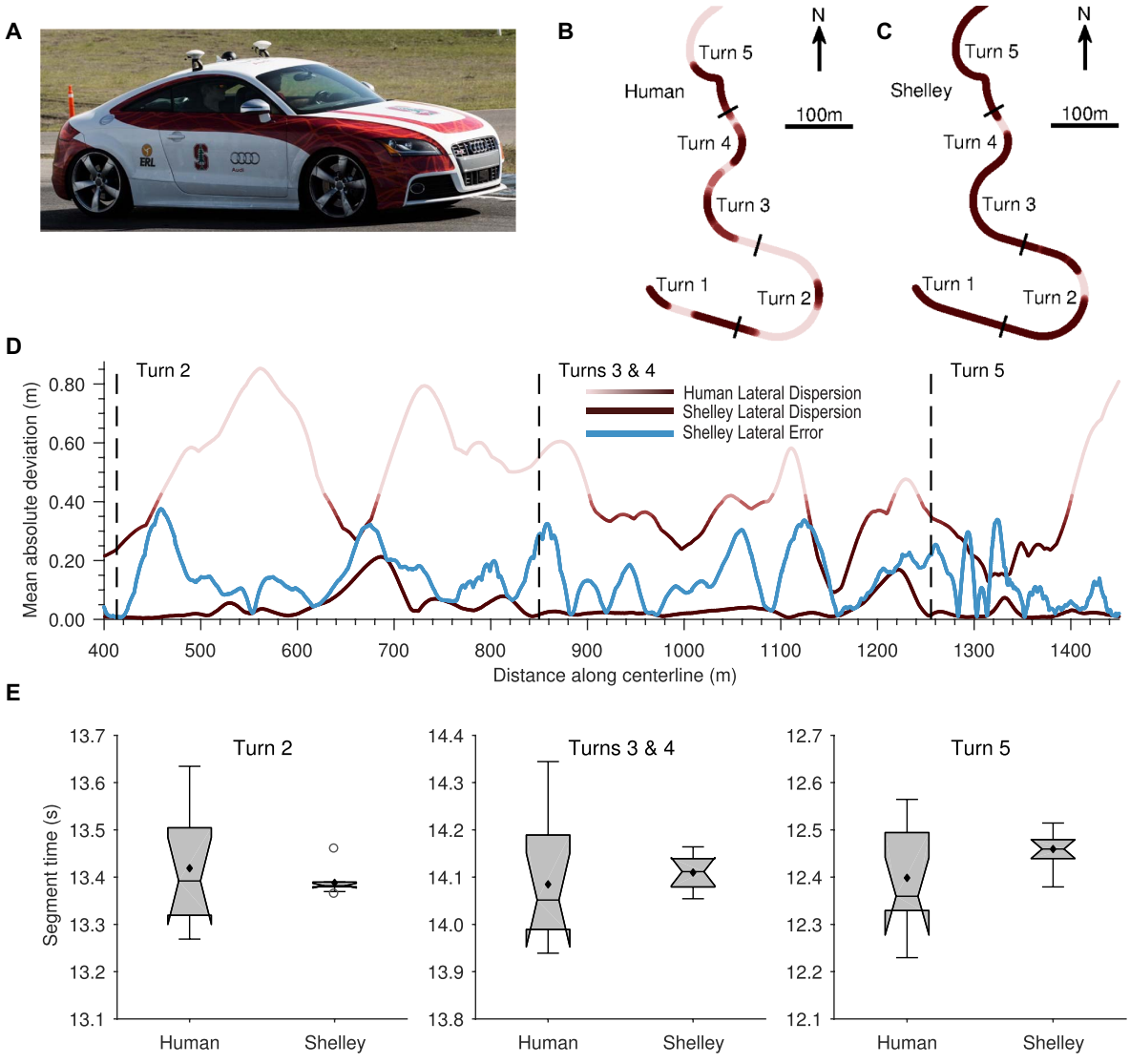

Fig. 2. Automated and human driving. (**A**) "Shelley," Stanford's autonomous Audi TTS designed to race at the limits of handling. (**B**) Human driver's MAD median path projected onto the first five turns of Thunderhill Raceway Park in Willows, California. (**C**) Shelley's MAD median path scaled by a factor of 4 to highlight relative differences. (**D**) MAD median path for both the human driver and Shelley (in red) along with Shelley's mean absolute deviation from the intended path (in blue). (**E**) Segment times from the champion amateur race driver benchmarked to Shelley.

Subsequently, these steady-state tire forces were converted to a desired steering input through the use of a physics-based tire model, which explicitly accounted for the effect of tire force generation and saturation. To compensate for inaccuracies in generated feedforward commands and disturbances, we used a simple path-based steering feedback controller to track a desired trajectory. This controller is based on e, the vehicle's lateral deviation from the desired trajectory, and $\Delta\Psi$, the vehicle's heading deviation from the desired trajectory, as shown in Fig. 1. The tire parameters of the physics-based model were fit using nonlinear least squares with experimental vehicle data.

To compare the automated approach with the experienced driver, we created a closed course study of racing performance consisting of the first five turns of Thunderhill Raceway Park in Willows, California. Both the automated vehicle and human participant attempted to complete the course in the minimum amount of time. This consisted of driving at accelerations nearing 0.95g while tracking a minimum time racing trajectory at the physical limits of tire adhesion. At this combined level of longitudinal and lateral acceleration, the vehicle was able to approach speeds of 95 miles per hour (mph) on portions of the track. Both automated vehicle and human participant participated in 10 trials of driving around the closed course. Tests were conducted under the same conditions, including ballasting the car to equate the mass of the vehicle during automated and human-driven testing. Even under these extreme driving conditions, the controller was able to consistently track the racing line with the mean path tracking error below 40 cm everywhere on the track (Fig. 2D).

To investigate the consistency of path following, we examined the mean absolute deviation from the median (MAD median) path dispersion, which is a robust measure of each driven trajectory's deviation from the track centerline. The experienced driver displayed a much greater path dispersion on average between laps than the automated vehicle (Fig. 2D). These data are also represented as a projection onto the

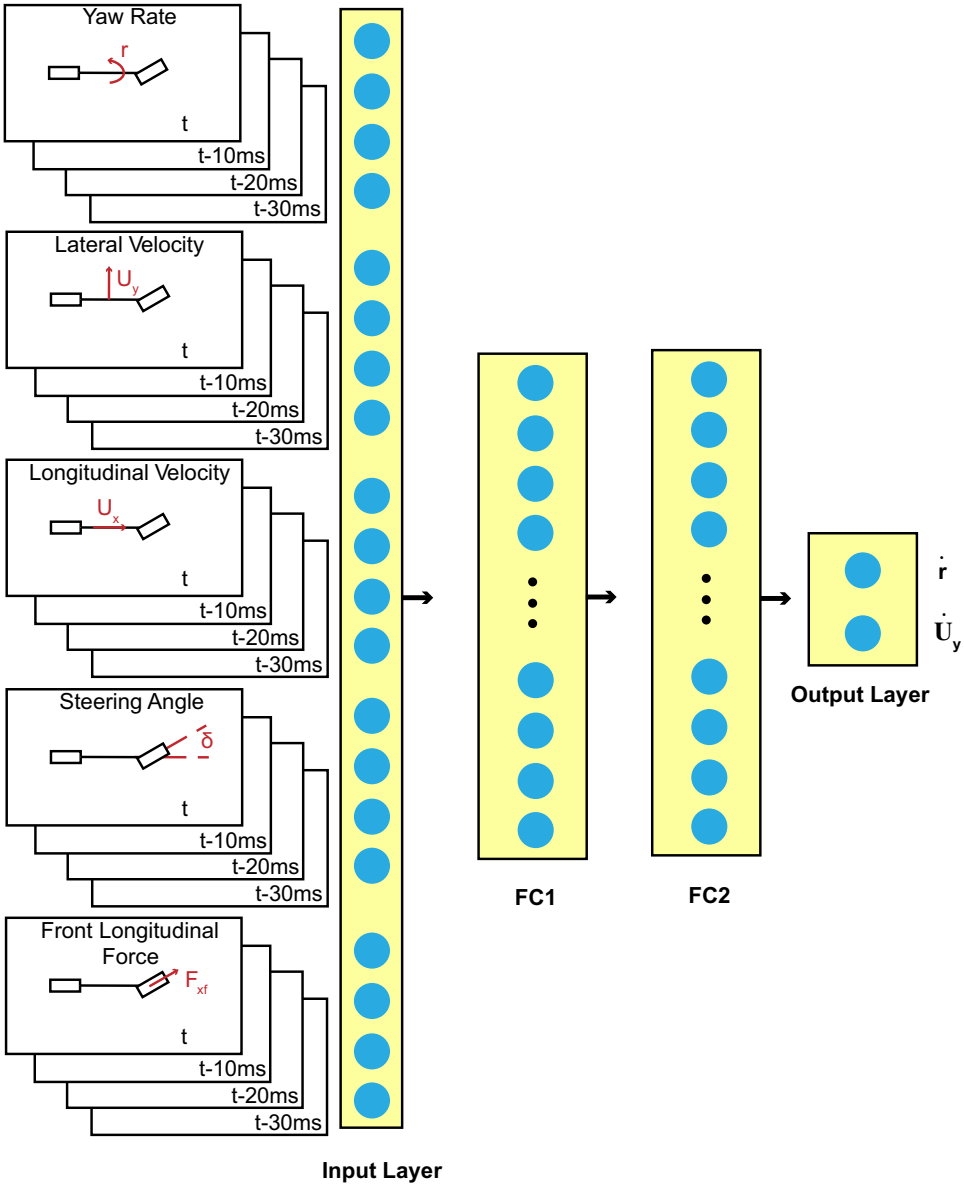

Fig. 3. Neural network dynamics model with input design based on the physics-based model. FC1 and FC2 denote fully connected layers in our two-layer feedforward neural network dynamics model.

track map in Fig. 2 (B and C), where N represents the north direction of the test course. The controller's consistency of path deviation demonstrates that the control approach was not only accurate but also precise. The low path dispersion of the automated vehicle can be explained by its objective to follow a precomputed trajectory using a highly accurate GPS-based localization system. As discussed later, the higher dispersion of the human-driven paths suggests that the human driver adopted a different strategy than the automated vehicle. Thus, the human and automated vehicle cannot be compared in terms of tracking accuracy or variability. They can, however, be compared in terms of time.

We used the metric of section time to compare both automated vehicle and human driver because this is the metric that both the racing driver and the automated vehicle's desired trajectory attempted to minimize. To compare section times on our closed racing course, we divided the course into three sections. Figure 2E shows section times recorded during the combined trials of both human participant and automated vehicle at Thunderhill Raceway Park. As the notched box and whiskers plots indicate, Shelley's times through each section of the track are well within the range of section times from the proficient human driver, which shows that the performance of the model-based controller is comparable with that of an experienced race driver at the limits of the vehicle's capabilities. On each box, the central line is the median, the point marked with a black diamond is the mean, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the adjacent sample no further than 1.5 times the interquartile range (IQR) beyond the edges, and possible "outliers" are plotted individually. The notched sections provide visual comparison intervals and were calculated as median $\pm 1.57 \times \mathrm{IQR}/\sqrt{n}$. The comparable lap times indicate comparable friction utilization by the simple feedforward-feedback controller with its physics-based model. The low path dispersion and comparable section times relative to a human driver resulted from a model tuned to a particular road surface. Having established a performance baseline relative to an experienced human driver, we could then use this controller performance as a baseline for the neural network model.

Motivated by the states and controls considered in the physics-based model, we chose to use a feedforward neural network with the inputs shown in Fig. 3. The neural network model consists of two hidden layers with 128 units in each layer and makes use of three delayed input states for each model state or control. Similar to the physics-based model, the network predicts the vehicle's yaw rate and lateral velocity derivatives. The network was initially trained in a supervised manner to replicate the physics-based model. After training on 200,000 trajectories over the full range of the physics-based model's input space, we updated the neural network with experimental vehicle data collected in high- and low-friction testing. High-friction testing was conducted at Thunderhill Raceway Park, and low-friction testing was conducted on a mixture of ice and snow at a test track near the Arctic Circle.

Although the neural network model can be used in a wide variety of control schemes, we wanted to compare it with the benchmark provided by the physics-based feedforward-feedback controller. Therefore, we used the learned neural network model to generate feedforward commands, making the same steady-state assumptions as the physics-based model. To generate feedforward steering commands, we found equilibrium points of the neural network dynamics model using a

second-order nonlinear optimization method. Measured speed and path curvature were used as inputs to the optimization to specify the correct feedforward command. This optimization was implemented online on the vehicle, and feedforward steering commands were calculated from the network at a rate of 20 Hz. To compensate for disturbances and model mismatch, we used the same simple path-based feedback controller structure in both cases for the comparison between controllers.

We compared both controllers by implementing them on an autonomous Volkswagen GTI (Fig. 4B), which we had the opportunity to use to obtain data on snow as it was prepared for automated driving. Figure 4A demonstrates the oval track on the skid pad at Thunderhill Raceway Park used to evaluate the two controllers. Both controller schemes used the same longitudinal speed profile and longitudinal controller and were tested at the limit of the vehicle's capability. When compared, we found that on turn entry, labeled "1" in Fig. 4C, the neural network controller learned to apply more steering than the physics-based model, resulting in lower tracking error in the middle of the turn. In the middle of the turn, the tracking error was influenced by the amount of road-tire friction available, with negative error showing the vehicle exceeding the grip limit. Furthermore, the neural network controller commanded less steering on turn exit ("3") because of its closer proximity to the desired path. The peaks shown on the exit and the straight sections were influenced by the steering feedback parameters, such as the controller gain and lookahead distance. We found that the neural network controller was able to visibly achieve a different distribution of lateral errors over the course of a lap at the limits (Fig. 4D). The count shown in this distribution is dependent on the bin size of 2.4 cm, where the number of bins chosen was 25. These results suggest that the neural network model is, in this case, able to achieve a higher degree of model fidelity than the physics-based model when implemented under the same set of steady-state assumptions and control architecture. That is, the controller satisfied the desired performance benchmark on this course.

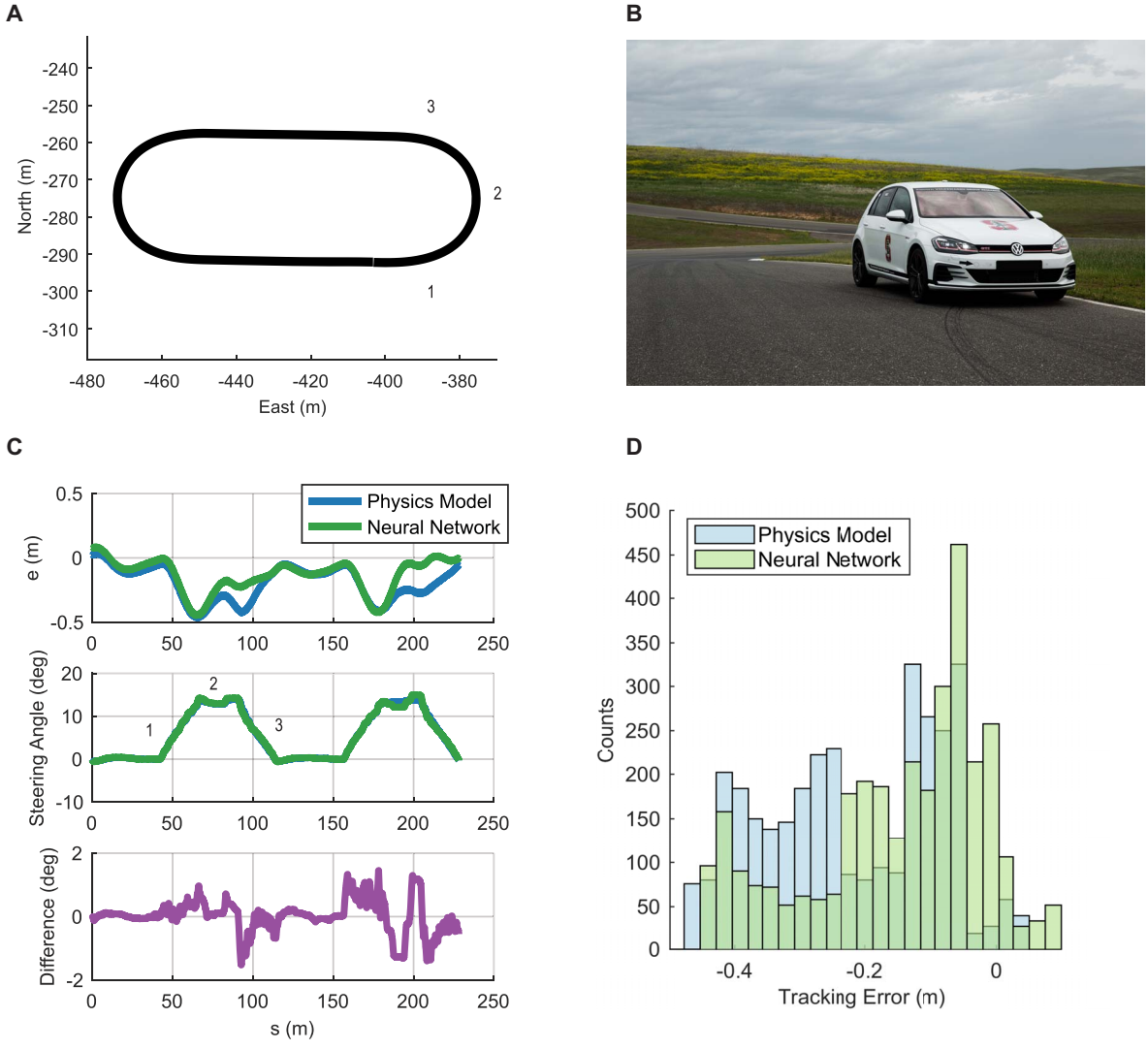

Fig. 4. Experimental model comparisons. (**A**) Experiment track map, showing corresponding sections 1, 2, and 3 in experimental data plots. (**B**) Picture of Volkswagen GTI experimental autonomous race vehicle. (**C**) Experimental comparison between physics-based controller and neural network controller showing lower tracking error at the limits on oval test track. (**D**) Histogram showing difference in lateral error distributions of neural network and physics-based controllers on oval test track.

The true power of the data-driven model, however, is not just to provide comparable performance to the physics-based approach. The neural network model also has the potential to incorporate higher-order dynamic effects and learn vehicle behavior on different road surfaces. To determine whether our learned model (Fig. 3) displays these characteristics, we examined predictions incorporating higher-fidelity vehicle dynamics modeling and multiple surface friction values in a pair of additional studies.

To demonstrate the modeling capability of the network relative to the simplified physics-based model, we used different fidelity dynamic models to generate training data based on a uniform random control policy. These data were used to both train the network and identify the best-fit parameters for the physics-based model, enabling their predictive capability to be compared. In the first comparison, the physics-based model itself generated the data, so there was no model mismatch between the physics-based model that generated the simulation data and the physics-based model that was learned. In this case, "No Mismatch" (Fig. 5A), the physics-based model substantially outperformed the neural network model and recovered the parameter set used for simulation. This is understandable, because the physics-based model

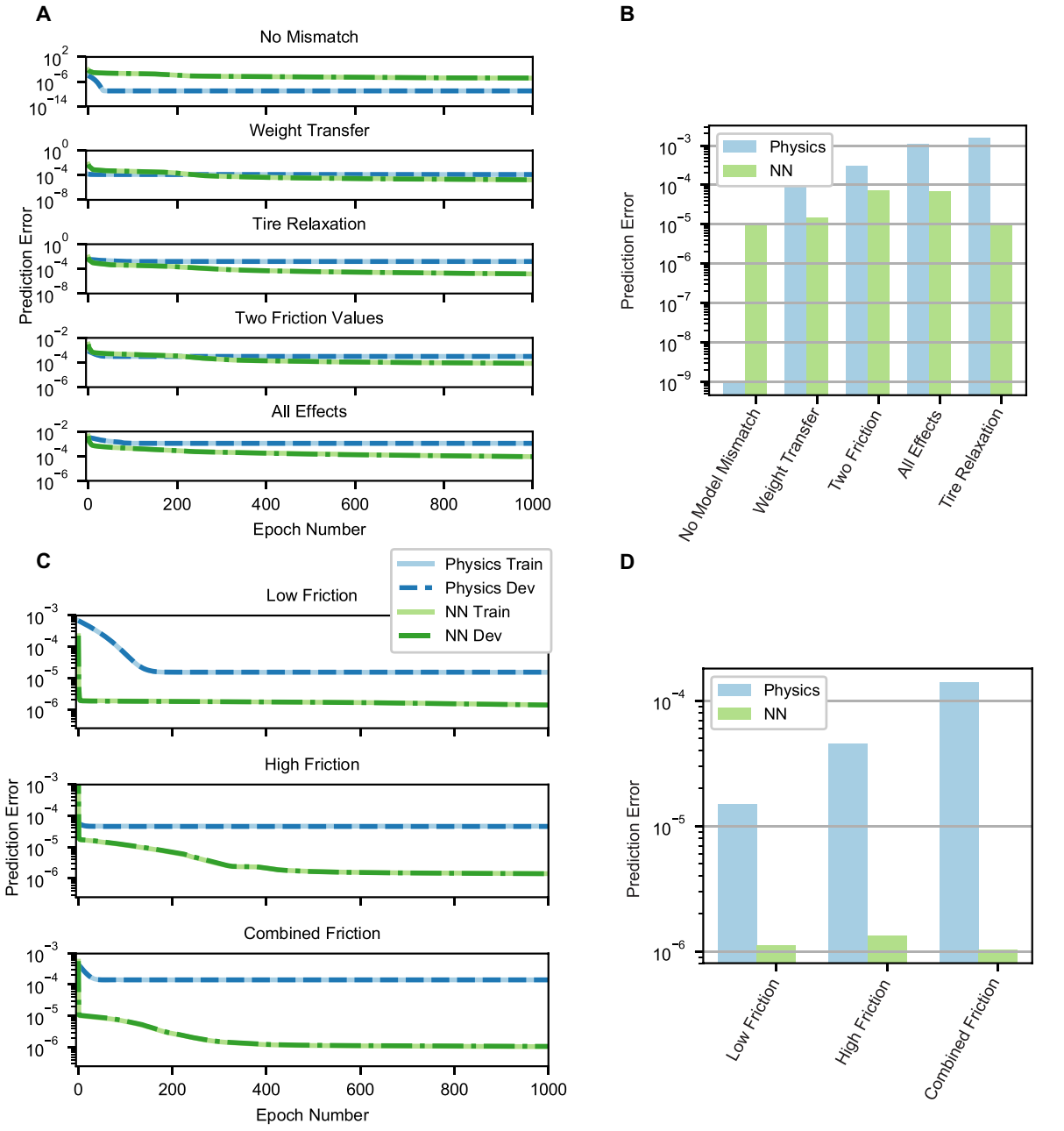

Fig. 5. Training and testing. (**A**) Training process for simulated data including multiple effects of model mismatch between data generation and optimization models. (**B**) Testing process for simulated data, showing generalization capability of learned models. (**C**) Training process for real collected vehicle data under various friction conditions. (**D**) Testing process for real vehicle data showing generalization capabilities of learned models.

represents the true model form behind the data, whereas the neural network was attempting to learn an approximate model.

The picture changed, however, when models of different fidelity generated the training data. We generated simulated data using physics-based models that were augmented to include the effects of longitudinal weight transfer, tire relaxation length, and multiple road surface friction values (Fig. 5A). When the data were fit to the simple physics-based model, these additional effects of model mismatch resulted in biased parameter values. We found that in all of these cases of model mismatch, the neural network model outperformed the physics-based model in prediction (Fig. 5A). Furthermore, we found that these results extended to held-out simulation data (Fig. 5B). These results are consistent with the insights from physics that were used to design the neural network predictive model. For example, in learning the effects of tire relaxation, the network was able to capture changing slip angle dynamics by including multiple delayed stages of states and inputs, whereas the physics-based model only used the current input and states to predict the vehicle's dynamics.

Motivated by the neural network's ability to capture a rich array of dynamics in simulation, we designed an additional study to evaluate the model's ability to make predictions on different road surfaces in real-world conditions. To do this, we collected both manually driven and automated data using the Volkswagen GTI platform (Fig. 4B). Additionally, we collected data from both high-friction driving on dry asphalt and low-friction driving on snow and ice. To illustrate the capability of the neural network to learn dynamics models for both low- and high-friction conditions, we separately trained and verified models for each condition individually (Fig. 5C). The results indicate an advantage for the neural network structure over the physics-based model in both the high- and low-friction cases. The data from both conditions could further be combined and used to train a single neural network and physics-based model. We found that this results in the highest training and testing errors for the physics-based model because of its inability to capture two different friction conditions as shown in Fig. 5C. The identified physics-based model characteristics represented approximately the average road surface condition, whereas the hidden nodes of the neural network model were able to implicitly represent and apply different conditions. As a result, the neural network outperformed the physics-based model by over an order of magnitude in both training and testing. These results indicate that the neural network model had greater predictive performance on both mixed and isolated road surface data, a characteristic that also generalizes to held-out test data as shown in Fig. 5D.

DISCUSSION

The results demonstrate that, with an appropriate model, a simple feedforward-feedback controller can provide path-tracking performance at the limits of a vehicle's friction capabilities, with friction utilization comparable with an experienced human race car driver. In addition, our feasibility study demonstrates that a neural network can provide the necessary model for such an approach, achieving performance that is better than a simple, but carefully tuned, static physics-based model. Most notably, such a model could predict performance on different friction surfaces without having to explicitly identify friction and demonstrated robustness when higher-fidelity vehicle dynamics characteristics were considered. The testing presented here shows that such neural network structures are viable candidates for dynamic models of automated vehicles and merit further investigation.

Benchmarking a tracking controller such as the one presented here against human performance is challenging. As demonstrated by the level of path dispersion, the human driver did not formulate the problem in terms of exact path tracking. Rather, the human driver tended to anchor the desired path at particular points, such as the apex of a turn, and focused on pushing the car to the friction limits (*13*). Because the approach of the human driver was fundamentally different from the typical automated vehicle architecture, section time seemed to be the fairest comparison of the two approaches. Both the human and the desired trajectory were operating with the intent to minimize time. Given the extreme sensitivity that section time displays to friction utilization, we can infer comparable friction utilization from comparable time.

Additionally, although our champion amateur driver is fast, professional drivers are faster still, suggesting an even greater capability to use friction. Thus, we have demonstrated capability comparable with an expert human, but we have not demonstrated the capability to exceed the high end of human performance. Accomplishing this will, in all likelihood, involve adopting some of the human driver's willingness to deviate from the path to more fully use friction and reduce time.

The results comparing the control performance of the neural network model and the physics-based model indicate that the controller using the neural network model had better path tracking performance on the path chosen for testing. The controller using the physics-based model attained a larger magnitude of lateral error, operating near 50 cm during cornering. However, on the basis of typical lane widths between 2.7 and 3.6 m and a typical vehicle width of 2 m, both controllers would keep a vehicle within a lane boundary even at the friction limits (*31*). The cornering speeds on this path did not exceed 26 mph, so this experiment reflects a reasonable model of an emergency maneuver for urban or suburban driving. Although further verification with other maneuvers is necessary before deployment, these results demonstrate the feasibility of a neural network approach to vehicle control at the limits.

When using the neural network model, the controller's feedforward computation only uses a small portion of the model's state space (where the vehicle is at steady state), whereas the neural network model has the capability to learn transient dynamics effects, as shown by its one-step prediction error. Therefore, the true potential of the neural network for control has not been realized in this particular control architecture. In addition, by making steady-state assumptions to generate commands from the neural network model, the state history of the network must be constrained. The feedforward controller did not make full use of the neural network's capabilities to simultaneously estimate and predict for variable friction surfaces. Other control structures, such as model predictive control, could make use of the estimation capabilities of the network, providing a simple way to tie together simultaneous estimation and control. Similarly, it would be possible to use a more complex physics model or estimate parameters online. However, this sequence of comparisons seemed to be the clearest way to establish a benchmark for the quality of the neural network model relative to more conventional approaches and human performance.

When learning the neural network model of the vehicle's dynamics, the learning process was efficient and only involved 35-min worth of data from the physical vehicle. Gathering additional data for other road surfaces, conditions, and tires is therefore possible without much additional cost. Additional investigation is required to establish the extent to which different conditions can be properly encoded in this neural network structure.

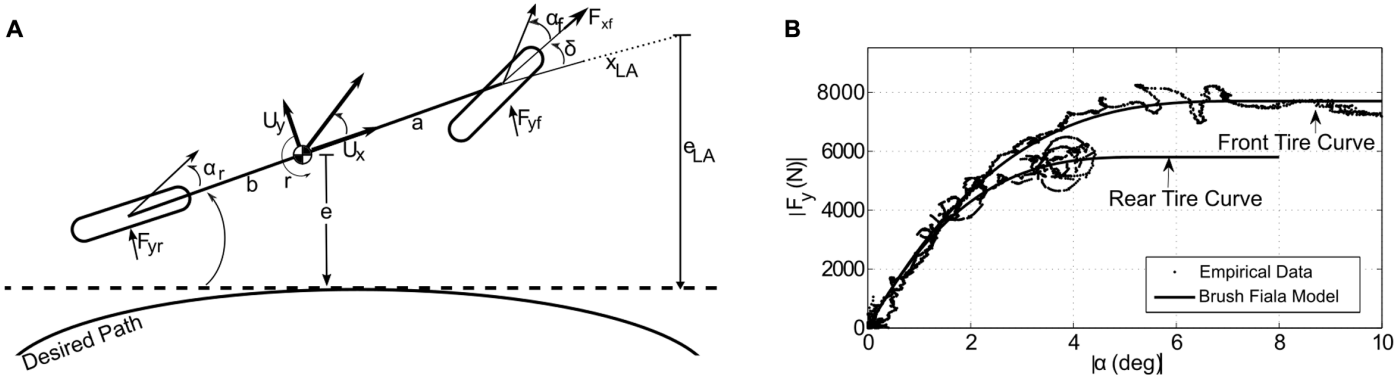

Fig. 6. Physics-based model and tire model. (**A**) Schematic of planar bicycle model including error states, referred to in this paper as the physics-based model. (**B**) Front and rear tire curves, with brush Fiala model fit to empirical tire data.

MATERIALS AND METHODS

Physics-based model

The proposed structure of the physics-based control design begins with the assumption that the vehicle dynamics are given by the planar single-track model, also referred to as the bicycle model. A schematic of the vehicle states is shown in Fig. 6A, and relevant parameters are described in Table 1. The planar bicycle model makes the key assumption that the left and right tires act to produce a single combined lateral force, resulting in just two lateral forces, F_{yf} and F_{yr}, acting at the front and rear axles. Actuation of steer angle δ at the front tire results in generation of the lateral tire forces through the slip angles α_f and α_r. The two resulting states that evolve are vehicle yaw rate r, which describes the vehicle angular rotation, and sideslip β, which is the ratio of the lateral velocity U_y to longitudinal velocity U_x. Because we are primarily interested in the lateral control of the vehicle, the longitudinal velocity U_x is considered a time-varying parameter and not a vehicle state.

In addition to the two vehicle states β and r, two additional states are required to show the vehicle's position relative to the desired path. These are also shown in Fig. 6A. The lateral path deviation, or lateral error e, is the distance from the vehicle center of gravity to the closest point on the desired path. The vehicle heading error $\Delta\Psi$ is defined as the angle between the vehicle's centerline and the tangent line drawn on the desired path at the closest point.

The equations of motion for the states shown in Fig. 6A are given by

$$\dot{U}_y = \frac{F_{yr} + F_{yf}\cos\delta + F_{xf}\sin\delta}{m} - rU_x \tag{1A}$$

$$\dot{r} = \frac{aF_{yf}\cos\delta + aF_{xf}\sin\delta - bF_{yr}}{I_z} \tag{1B}$$

$$\beta = \arctan\left(\frac{U_y}{U_x}\right) \tag{1C}$$

$$\dot{e} = U_x\sin\Delta\Psi + U_y\cos\Delta\Psi \tag{1D}$$

$$\Delta\dot{\Psi} = r - \dot{s}\kappa \tag{1e}$$

Table 1. Physics model definitions.

Parameter	Symbol	Units
Front axle to CG	a	m
Rear axle to CG	b	m
Front lateral force	F_{yf}	N
Front longitudinal force	F_{xf}	N
Front tire slip	α_f	rad
Rear lateral force	F_{yr}	N
Rear tire slip	α_r	rad
Steer angle input	δ	rad
Yaw rate	r	rad/s
Sideslip	β	rad
Lateral path deviation	e	m
Heading deviation	$\Delta\Psi$	rad
Longitudinal velocity	U_x	m/s
Lateral velocity	U_y	m/s

To obtain equations of motion for feedforward controller design in terms of derivatives of the error states, we can take time derivatives of $\dot{e}$ and $\Delta\dot{\Psi}$, set $F_{xf} = 0$, and substitute from Eqs. 1A and 1B, yielding

$$\ddot{e} = \frac{F_{yf} + F_{yr}}{m} - U_x\kappa\dot{s} \tag{2A}$$

$$\Delta\ddot{\Psi} = \frac{aF_{yf} - bF_{yr}}{I_z} - \kappa\ddot{s} - \dot{\kappa}\dot{s} \tag{2B}$$

The bicycle model dynamics are abbreviated as shown in Eq. 3B as f_{Bike} for describing the vector-valued learned bicycle model. The tire parameters (C_f, C_r, μ) in this model are learned to predict the bicycle

model dynamics from measured vehicle data. The inputs to the lateral bicycle model at a time t are denoted by x_t as shown in Eq. 3A.

$$x_t = (r, U_y, U_x, \delta, F_{xf})_t \tag{3A}$$

$$\begin{bmatrix} \dot{r} \\ \dot{U}_y \end{bmatrix} = f_{\text{Bike}}(x_t, C_f, C_r, \mu) \tag{3B}$$

Lookahead controller

The physics-based controller used as a benchmark for control is based on a feedforward-feedback architecture, a block diagram of which is shown in Fig. 1. Inputs to the feedforward steering angle δ_{FFW} are current path curvature κ and the forward velocity U_x. Inputs to the feedback steering angle δ_{FB} are the error states e and $\Delta\Psi$. The total steering command δ is the sum of the feedback and feedforward inputs.

The objective of the steering feedforward is to provide an estimate of the steer angle required to traverse a path with a known path curvature and velocity profile. This minimizes the level of compensation required by the steering feedback, reducing tracking errors and allowing for less overall control effort. The feedforward steering angle should depend only on the desired trajectory and should be independent of the actual vehicle states.

To design the feedforward steering controller from the physics-based model, we made the simplifying assumption that the vehicle operates under steady-state cornering conditions. This assumption has been shown (*32*) to reduce oscillation of the controller's yaw rate response. Setting $\dot{s} = U_x$ and $\ddot{s} = \dot{\kappa} = 0$ in Eq. 2 yields the following steady-state front and rear tire forces:

$$F_{yf}^{FFW} = \frac{mb}{L} U_x^2 \kappa \tag{4A}$$

$$F_{yr}^{FFW} = \frac{ma}{L} U_x^2 \kappa \tag{4B}$$

Under steady-state conditions and assuming small angles, the feedforward steering angle of the vehicle relates to the front and rear lateral tire slip α_f and α_r and path curvature κ by vehicle kinematics

$$\delta_{FFW} = L\kappa - \alpha_f^{FFW} + \alpha_r^{FFW} \tag{5}$$

where α_f^{FFW} and α_r^{FFW} are the lumped front and rear feedforward tire slip angles.

The choice of feedforward tire slip angles is related to the tire forces in Eq. 4 via a tire model $F_y = f(\alpha)$. To account for saturation of tire force with increasing tire slip magnitude, a single friction coefficient brush Fiala model (*33*) maps lateral tire slip angles into tire force as follows

$$F_{y\diamond} = \begin{cases} -C_\diamond \tan\alpha_\diamond + \dfrac{C_\diamond^2}{3\mu F_{z\diamond}} |\tan\alpha_\diamond| \tan\alpha_\diamond \\ \quad - \dfrac{C_\diamond^3}{27\mu^2 F_{z\diamond}^2} \tan^3\alpha_\diamond, & |\alpha_\diamond| < \arctan\left(\dfrac{3\mu F_{z\diamond}}{C_\diamond}\right) \\ -\mu F_{z\diamond} \operatorname{sgn} \alpha_\diamond, & \text{otherwise} \end{cases} \tag{6}$$

where the symbol $\diamond \in [f, r]$ denotes the lumped front or rear tire, μ is the surface coefficient of friction, and $C_\diamond$ and $F_{z\diamond}$ are the corresponding cornering stiffness and normal load parameters. The cornering stiffnesses and friction coefficient were determined by using nonlinear least squares to fit experimental data, as shown in Fig. 6B.

With the feedforward design complete, the remaining step is to design the feedback controller. The goal of the feedback controller is to minimize a lookahead error e_{LA}, which is the vehicle tracking error projected a distance x_{LA} in front of the vehicle (Fig. 6A).

The lookahead error and resulting feedback control law are given by

$$\delta_{FB} = -k_p(e + x_{LA}\sin(\Delta\Psi + \beta_{ss})) \tag{7}$$

$$\beta_{ss} = \alpha_r^{FFW} + b\kappa \tag{8}$$

with proportional gain k_p. Note that a key feature of this feedback controller is incorporation of steady-state sideslip information β_{ss}. In (*32*), accounting for sideslip in the feedback control law eliminated steady-state path tracking errors, assuming no plant-model mismatch. Furthermore, a linear systems analysis shows that using steady-state sideslip instead of measured vehicle sideslip allows the controller to maintain robust stability margins.

Comparison with human driver

To provide a comparison for benchmarking the physics-based controller, we compared Shelley's performance with that of a proficient driver. The proficient driver has years of amateur racing experience, including working with the research team. In addition, this driver has extensive experience with the course. The physics-based lookahead controller was implemented on Shelley, a 2009 Audi TTS. Shelley is equipped with an active brake booster, throttle-by-wire, and electronic power steering for full automated control. In addition, Shelley used an integrated navigation system aided by differential global navigation satellite system signals to obtain multicentimeter accuracy of position measurements. The system included a dSPACE MicroAutoBox II, which was used to record data from the vehicle as well as execute control commands at 200 Hz.

The objective of comparing differences in racing performance between the physics-based lookahead controller and the proficient driver is shown through comparisons of section times and path dispersion. Experimental tests were performed at Thunderhill Raceway Park, located in Willows, California. Both the human participant and Shelley drove 10 consecutive trials of turns 2 to 6 on the East track. In each trial, the vehicle's total mass was consistent. GPS markers for each section were used to segment each trial into three portions for analysis. For both participants, the driven path recorded from GPS was used to calculate the lateral deviation and distance along the track centerline. To characterize the dispersion of each participant's trajectories, we chose the MAD median.

Learning a global neural network model

As opposed to learning parameters in our physics-based model, we learned a neural network model capable of foregoing the step of modeling and identifying latent states, such as vehicle road friction interaction. In modeling the unknown or changing dynamics, the designer must condense all unknown or unmodeled effects into a given parameter set of predefined dimension. By using a neural network model using both control and state history denoted with a vector h_t,

we imposed less structure on the system identification task, allowing the network model to identify its own internal representations of time-varying dynamics. In the expanded network input space, a given point $P_t = h_t$ can be used to fully construct the system's latent state, given that the number of delayed stages T is long enough. This is demonstrated by Takens's theorem and further demonstrated in learning complex helicopter dynamics models (*27*, *34*).

We learn a neural network dynamics model of the form shown in Eq. 9, where θ represents the learned network weight parameters, x_t represents each stage of delayed state and control inputs, h_t denotes the history of state and control inputs, and f_{NN} is the abbreviation for the vector-valued two–hidden layer neural network dynamics model using softplus activation functions. In the network equations, a represents layer activations and z represents the weighted input to a given layer.

$$x_t = (r, U_y, U_x, \delta, F_{xf})_t \tag{9A}$$

$$h_t = [x_t, \ldots, x_{t-T}] \tag{9B}$$

$$z_1 = W_1^T h_t + b_1 \tag{9C}$$

$$a_1 = \log(1 + e^{z_1}) \tag{9D}$$

$$z_2 = W_2^T a_1 + b_2 \tag{9E}$$

$$a_2 = \log(1 + e^{z_2}) \tag{9F}$$

$$\begin{bmatrix} \dot{r} \\ \dot{U}_y \end{bmatrix} = W_3^T a_2 + b_3 \tag{9G}$$

$$\theta = [W_1, b_1, W_2, b_2, W_3, b_3] \tag{9H}$$

$$\begin{bmatrix} \dot{r} \\ \dot{U}_y \end{bmatrix} = f_{NN}(h_t, \theta) \tag{9I}$$

The measured targets for the network in training consist of the next measured yaw rate (r) and lateral velocity (U_y) states. Measured inputs to the network were delayed 10 ms per stage x_t. To predict target states, we learned the state derivatives with the network and then used Euler integration with a 10-ms time step (Δt) as shown below. The predicted targets for the physics-based model are also similarly calculated using Euler integration with a 10-ms time step, where $_{t+1}$ denotes the next sampled time step in Eq. 10.

$$\begin{bmatrix} \hat{r} \\ \hat{U}_y \end{bmatrix}_{t+1} = \begin{bmatrix} r \\ U_y \end{bmatrix}_t + \Delta t \cdot f_{NN}(h_t, \theta) \tag{10}$$

Simulated data

To study the neural network's capability to learn representations for these effects, we designed a simulation study using the physics-based model with varying degrees of additional model complexity. To demonstrate the capability of a neural network model for modeling vehicle dynamics, we used the single-track vehicle model and the Fiala tire model as previously described. To generate a dataset using the physics-based model as shown in Eq. 1, we first sampled the initial starting conditions (r, U_y, U_x) uniformly at random in the space of stable initial states. We also sampled the initial controls (δ, F_{xf}) uniformly at random in the space of possible inputs. To create a single trajectory of length T used as an input to the network for training, we used a uniform random control policy for both δ and F_{xf} with physics-based model dynamics to determine next states for the rest of the control trajectory. In addition to using the high-friction physics-based model to generate training data, we also modeled the following additional dynamics effects.

Weight transfer

During high-performance driving, one common effect that influences the vehicle's dynamics is longitudinal weight transfer. This effect impacts the vehicle's dynamics by increasing or decreasing the amount of normal force experienced on each tire as a result of acceleration or braking. This is shown in Eq. 11, where h is the height to the center of gravity of the vehicle, m is the vehicle mass, and g is acceleration due to gravity. When coupled with the Fiala tire model, weight transfer acts to increase or decrease the force capability of a given tire. We generated a simulated dataset with a dynamics model consisting of the physics-based model plus longitudinal weight transfer.

$$F_{zf} = \frac{b}{L} mg - \frac{h}{L} ma_x \tag{11A}$$

$$F_{zr} = \frac{a}{L} mg + \frac{h}{L} ma_x \tag{11B}$$

Tire relaxation

During low-speed driving, another dominant effect to model is tire relaxation length. Tire relaxation length can be modeled as a delay in the amount of lateral force experienced by each tire as shown below in Eq. 12 (*35*). The amount of delay is characterized by the tire relaxation length σ, a property of the tire, as well as the magnitude of the velocity V of the vehicle. One simulated dataset was generated with a physics-based model including the effects of tire relaxation length.

$$\dot{\alpha}_f = \frac{V}{\sigma_f}\left(\arctan\left(\frac{U_y + ar}{U_x}\right) - \delta - \alpha_f\right) \tag{12A}$$

$$\dot{\alpha}_r = \frac{V}{\sigma_r}\left(\arctan\left(\frac{U_y - br}{U_x}\right) - \alpha_r\right) \tag{12B}$$

Varying tire friction

To verify that the network model is capable of high predictive performance in multiple environmental conditions, we generated one dataset that consists of simulated high- and low-friction data. We did this by modeling both low- and high-friction surfaces explicitly as the friction parameter in the Fiala tire curve, which have the following values as shown in Eq. 13. We generated a simulated dataset for training consisting of 200,000 sampled trajectories, where half of the dataset was collected at high friction and half was collected at low simulated friction.

$$\mu_{low} = 0.3 \tag{13A}$$

$$\mu_{high} = 1.0 \tag{13B}$$

Last, one simulated dataset was generated where all of these effects were added to the physics-based model, and 200,000 samples were generated for training.

Experimental data

To demonstrate the capability of network models in real prediction tests, we collected vehicle data under both high- and low-friction driving conditions. Collected real vehicle data encompassed each of the modeled effects shown in simulation, including both high and low friction, plus additional difficult-to-model effects such as suspension geometry or internal control loops. Collected data consist of 214,629 trajectory samples or about 35 min of driving data split roughly equally between high- and low-friction driving. Data were collected using an automated Volkswagen GTI with a similar architecture to Shelley, as previously described. Low-friction data were collected in a testing track near the Arctic Circle, consisting of a variety of speeds up to the limits of the vehicle's capabilities on a low-friction test track. High-friction data were collected from a variety of maneuvers at the handling limits, with data collected at Thunderhill Raceway Park.

An integrated navigation system was used to measure U_x, U_y, r, and a_x, where a_x denotes the longitudinal acceleration of the vehicle. Measurements of the vehicle's steering angle were acquired from the vehicle's controller area network (CAN). Measurements were recorded at 100 Hz on a dSPACE MicroAutoBox. For the learning task, recorded data were filtered using a second-order Butterworth low-pass filter with a 6-Hz cutoff to only learn relevant vehicle dynamics and not higher-frequency effects such as suspension vibration.

Optimization and training

In preparing data for learning, each experimental dataset was divided 70% train, 15% held out for development, and 15% held out for model testing. To break temporal correlations within the dataset, we randomized the data so that each sample consisted of a time-correlated trajectory but any given two samples were not correlated. To compare capabilities of both the physics-based model and the neural network model, we optimized both models to fit the observed dynamics training data. We used the mean squared error objective for training as shown below, where $\hat{U}_y$ and $\hat{r}$ denote a predicted quantity from a model; otherwise, U_y and r are measured target quantities, and N denotes the number of training examples. By training the network, we are solving the optimization problem shown in Eq. 14, where θ denotes the weights of the network.

$$\underset{\theta}{\text{minimize}} \frac{1}{N}\sum_{i=1}^{N}\left\| \begin{bmatrix} r \\ U_y \end{bmatrix}_{t+1} - \begin{bmatrix} \hat{r} \\ \hat{U}_y \end{bmatrix}_{t+1} \right\|_2$$

$$\text{subject to} \begin{bmatrix} \hat{r} \\ \hat{U}_y \end{bmatrix}_{t+1} = \begin{bmatrix} r \\ U_y \end{bmatrix}_{t} + \Delta t \cdot f_{\text{NN}}(h_t, \theta) \qquad (14)$$

To optimize the physics-based model to observed data, we similarly formed an optimization model to train the model parameters. The parameters learned for the physics-based model consist of the tire friction (μ) and the tire front and rear cornering stiffnesses (C_f and C_r). Because of model mismatch in reality, any observed model mismatch will result in learned parameter variation of the model. Similarly, in the data generated from physics-based models including additional unmodeled effects, model mismatch also resulted in parameter variation. This led to solving the optimization problem (Eq. 15) for training the physics-based model of recorded data.

$$\underset{C_f, C_r, \mu}{\text{minimize}} \frac{1}{N}\sum_{i=1}^{N}\left\| \begin{bmatrix} r \\ U_y \end{bmatrix}_{t+1} - \begin{bmatrix} \hat{r} \\ \hat{U}_y \end{bmatrix}_{t+1} \right\|_2$$

$$\text{subject to} \begin{bmatrix} \hat{r} \\ \hat{U}_y \end{bmatrix}_{t+1} = \begin{bmatrix} r \\ U_y \end{bmatrix}_{t} + \Delta t \cdot f_{\text{Bike}}(x_t, C_f, C_r, \mu) \qquad (15)$$

Parameters for the physics-based model were initialized from a random Gaussian distribution for training, and parameters for the neural network model were initialized using Xavier uniform initialization. We used the Adam optimization method with default parameter initializations to perform a first-order optimization of the learned parameters in both the physics-based model and the neural network model (*36*). Training was performed using mini batches of 1000 samples for each update. The learning framework was implemented in TensorFlow using graphics processing unit parallelization in Python and trained with a computing cluster with an Intel i7 processor and Nvidia 1080 graphics card (*37*). For a single training dataset, the learning process took about 25 min.

Control using learned models

To control the vehicle using a learned neural network model, we developed a control method that leverages the lookahead feedforward-feedback control architecture. In this case, we used the learned neural network model to develop the approximate feedforward steering and sideslip commands. The neural network model was first trained on high-friction simulated data from a bike model simulation sampled with 200,000 trajectories over the state space of the model. Once the initial learning process converged, we used real experimental data in the next step of the training process inspired by the technique of subsequently increasing model fidelity simulators (*38*). We supplemented our datasets with both high- and low-friction experimental test data. Additional experimental data were obtained from tracking our oval test track at a range of lateral accelerations with the baseline feedback-feedforward controller to ensure that there was similarity in the test distribution of our controller. Once these data were combined, the neural network vehicle model was retrained to fit the newly collected real data as shown in Fig. 7. Once the model was optimized on real collected data, the model was used to generate feedforward steering and sideslip commands.

From the kinematics, which are not learned in the neural network dynamics model, we found that in steady state, we arrived at

$$r_{ss} = \kappa \dot{s} \qquad (16A)$$

$$F_{x,ss} = -m r_{ss} U_y \qquad (16B)$$

These steady-state values were used as inputs to the network in an optimization problem to find a stationary point in the model for feedforward control. Conditions for finding a stationary point are illustrated in the network model by finding a point where the state derivatives are zero for a given set of control and state inputs. For the

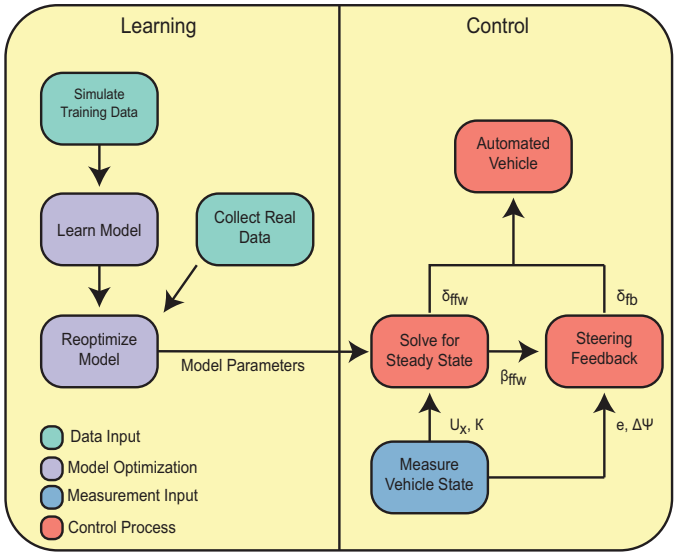

Fig. 7. Schematic of control process using a learned neural network model.

neural network with history, this consists of each of the delayed state and control inputs being constrained to be the same as shown in Eq. 17B.

$$\begin{bmatrix} \dot{r} \\ \dot{U}_y \end{bmatrix} = f_{\mathrm{NN}}(r, U_y, U_x, \delta, F_x, \ldots, r, U_y, U_x, \delta, F_x) \tag{17A}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = f_{\mathrm{NN}}(r_{ss}, U_{y,ss}, U_{x_{\mathrm{measured}}}, \delta_{ss}, F_{x,ss}, \ldots, r_{ss}, U_{y,ss}, U_{x_{\mathrm{measured}}}, \delta_{ss}, F_{x,ss}) \tag{17B}$$

To compute the feedforward values as in the physics-based controller, both a speed and path curvature must be provided. Speed was measured from the vehicle as an input to the network, and curvature was provided from the precomputed trajectory to follow. During the control process, curvature was calculated using a map-matching algorithm online on the vehicle.

Last, the controller computes the feedforward commands δ_{ffw} and $U_{y,\mathrm{ffw}}$ that best achieve equilibrium within the actuation limits (δ_u,δ_l). The solution was optimized with a constrained second-order interior point optimization method. This process used CasADi and IPOPT to solve the following nonlinear optimization problem shown in Eq. 18 (*39*, *40*).

$$\begin{aligned} \underset{\delta_{\mathrm{ffw}}, U_{y,\mathrm{ffw}}}{\text{minimize}} \quad & \dot{r}^2 + \dot{U}_y^2 \\ \text{subject to} \quad & \delta \le \delta_u \\ & \delta \ge \delta_l \\ & \begin{bmatrix} \dot{r} \\ \dot{U}_y \end{bmatrix} = f_{\mathrm{NN}}(\delta_{\mathrm{ffw}}, U_{y,\mathrm{ffw}}) \end{aligned} \tag{18}$$

This optimization problem aims to find the closest conditions and inputs for steady state of the learned model in a least-squares sense. In practice, the optimization finds solutions that are numerically close to steady state, attaining a maximum value of the cost function of 1.0072×10^{-18} during testing on the vehicle. When used for control, this optimization problem was solved every 50 ms on a computer with an Intel i7 processor. The kinematic steering angle was used as an initialization to warm start the nonlinear optimization. Although we initialized the optimization with the kinematic steering angle, the problem tends to not be very sensitive to the initial guess.

To obtain the steering values that are used for control, we used the steady-state lateral velocity along with the current speed to find the feedforward sideslip angle β_{ffw} as shown in Eq. 19A. Once the feedforward sideslip command was computed, it was used in the lanekeeping feedback control scheme, along with the computed δ_{ffw} as shown in Eq. 19B. The final steering command consists of the feedforward steering command found from solving for the network's stationary point, plus an added term for path-based steering feedback. The steering feedback, similar to the physics-based controller, includes a feedforward sideslip term that was calculated from solving for the optimal steady-state lateral velocity from the network.

$$\beta_{\mathrm{ffw}} = \arctan(U_{y,\mathrm{ffw}}/U_x) \tag{19A}$$

$$\delta = \delta_{\mathrm{ffw}} - k_p(e + x_{\mathrm{LA}}\sin(\Delta\Psi + \beta_{\mathrm{ffw}})) \tag{19B}$$

The resulting steering command was calculated and sent to the vehicle over its CAN interface. This steering command was tracked via a low-level steering controller, which applies a torque on the vehicle's steering system to obtain a desired road-wheel angle.

Comparison of both the lookahead physics-based controller and neural network feedforward controller was implemented on an automated Volkswagen GTI. Both controllers were tested at Thunderhill Raceway Park, on an oval map on the asphalt high-friction skid pad. Comparison of tracking errors occurred when both controllers were tracking a 0.9*g* longitudinal speed profile. A feedforward-feedback–based longitudinal controller was used in both implementations on the vehicle. Analysis of data was completed in MATLAB 2016b.

SUPPLEMENTARY MATERIALS

robotics.sciencemag.org/cgi/content/full/4/28/eaaw1975/DC1
Fig. S1. Measured and predicted sideslip from experimental control comparison.
Fig. S2. Measured longitudinal speeds from experimental control comparison.
Table S1. Test dataset results from simulation learning study.
Table S2. Test dataset results from experimental data learning study.
Movie S1. Neural network controller implemented on automated GTI, tested on oval track at Thunderhill Raceway Park.

REFERENCES AND NOTES

1. National Highway Traffic Safety Administration, National motor vehicle crash causation survey: Report to Congress (National Highway Traffic Safety Administration Technical Report DOT HS, U.S. Department of Transportation, 2008).
2. D. Dolgov, S. Thrun, M. Montemerlo, J. Diebel, Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robot. Res.* **29**, 485–501 (2010).
3. J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Zico Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, S. Thrun, Towards fully autonomous driving: Systems and algorithms, in *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium* (IEEE, 2011).
4. M. Maurer, J. C. Gerdes, B. Lenz, H. Winner, *Autonomous Driving* (Springer, 2016).
5. C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumer, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar,

Autonomous driving in urban environments: Boss and the Urban Challenge. *J. Field Robot.* **25**, 425–466 (2008).
6. P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, D. Hrovat, Predictive active steering control for autonomous vehicle systems. *IEEE Trans. Contr. Syst. Technol.* **15**, 566–580 (2007).
7. G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, S. Thrun, Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing, in *Proceedings of the 2007 American Control Conference* (IEEE, 2007).
8. M. Gerdts, S. Karrenberg, B. Müller-Beßler, G. Stock, Generating locally optimal trajectories for an automatically driven car. *Optim. Eng.* **10**, 439–463 (2009).
9. E. Velenis, E. Frazzoli, P. Tsiotras, J. Lu, On steady-state cornering equilibria for wheeled vehicles with drift, in *Proceedings of the IEEE Conference on Decision and Control* (IEEE, 2009).
10. Y. Gao, T. Lin, F. Borrelli, E. Tseng, D. Hrovat, Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads, in *ASME Dynamic Systems and Control Conference* (American Society of Mechanical Engineers, 2010).
11. A. Liniger, A. Domahidi, M. Morari, Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Appl. Methods* **36**, 628–647 (2015).
12. V. A. Laurense, J. Goh, J. C. Gerdes, Path-tracking for autonomous vehicles at the limit of friction, in *Proceedings of the American Control Conference* (IEEE, 2017), pp. 5586–5591.
13. J. Kegelman, L. Harbott, J. C. Gerdes, Insights into vehicle trajectories at the handling limits: Analysing open data from race car drivers. *Vehicle Syst. Dyn.* **55**, 191–207 (2017).
14. M. C. Best, Identifying tyre models directly from vehicle test data using an extended Kalman filter. *Vehicle Syst. Dyn.* **48**, 171–187 (2010).
15. L. Ray, Nonlinear state and tire force estimation for advanced vehicle control. *IEEE Trans. Control Syst. Technol.* **3**, 117–124 (1995).
16. J.-O. Hahn, R. Rajamani, L. Alexander, GPS-based real-time identification of tire-road friction coefficient. *IEEE Trans. Control Syst. Technol.* **10**, 331–343 (2002).
17. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. *Nature* **521**, 436–444 (2015).
18. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dielman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepal, D. Hassabis, Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
19. K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**, 359–366 (1989).
20. D. A. Pomerleau, Knowledge-based training of artificial neural networks for autonomous robot driving, in *Robot Learning* (Kluwer Academic Publishing, 1993), pp. 305–313.
21. K. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Netw.* **1**, 4–27 (1990).
22. S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, C. J. Tomlin, Learning quadrotor dynamics using neural network for flight control. arXiv:1610.05863 [cs.SY] (19 October 2016).
23. G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, E. A. Theodorou, Information theoretic MPC using neural network dynamics, in *30th Conference on Neural Information Processing Systems* (NIPS, 2016).
24. S. J. Rutherford, D. J. Cole, Modelling nonlinear vehicle dynamics with neural networks. *Int. J. Vehicle Design* **53**, 260–287 (2010).
25. A. Ghazizadeh, A. Fahim, M. El-Gindy, Neural networks representation of a vehicle model: 'Neuro–Vehicle (NV)'. *Int. J. Vehicle Design* **17**, 55–75 (1996).
26. E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, P. A. Ioannou, High-order neural network structures for identification of dynamical systems. *IEEE Trans. Neural Netw.* **6**, 422–431 (1995).
27. A. Punjani, P. Abbeel, Deep learning helicopter dynamics models, in *IEEE International Conference on Robotics and Automation* (ICRA, 2015), pp. 3223–3230.
28. I. Lenz, R. Knepper, A. Saxena, DeepMPC: Learning deep latent features for model predictive control, in *Proceedings of the Robotics: Science and Systems Conference* (Sapienza University of Rome, 2015), vol. 9.
29. J. Funke, P. Theodosis, R. Hiniyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Müller-Beßler, B. Huhnke, Up to the limits: Autonomous Audi TTS, in *Intelligent Vehicles Symposium (IV)* (IEEE, 2012), pp. 541–547.
30. K. Kritayakirana, J. C. Gerdes, Autonomous vehicle control at the limits of handling. *Int. J. Vehicle Autonomous Syst.* **10**, 271–296 (2012).
31. M. W. Hancock, B. Wright, *A Policy on Geometric Design of Highways and Streets* (AASHTO, 2013).
32. N. R. Kapania, J. C. Gerdes, Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling. *Vehicle Syst. Dynamics* **53**, 1687–1704 (2015).
33. H. B. Pacejka, *Tire and Vehicle Dynamics* (Elsevier, 2012).
34. F. Takens, Detecting strange attractors in turbulence, in *Dynamical Systems and Turbulence, Warwick 1980* (Springer, 1981).
35. S. M. Laws, C. D. Gadda, J. C. Gerdes, Frequency characteristics of vehicle handling: Modeling and experimental validation of yaw, sideslip, and roll modes to 8Hz, in *Proceedings of the International Symposium on Advanced Vehicle Control* (National Tsing Hua University, 2006).
36. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. arXiv:1412.6980 [cs.LG] (22 December 2014).
37. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, TensorFlow: A system for large-scale machine learning, in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (OSDI, 2016).
38. M. Cutler, T. J. Walsh, J. P. How, Reinforcement learning with multi-fidelity simulators, in *IEEE International Conference on Robotics and Automation* (ICRA, 2014), pp. 3888–3895.
39. J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, M. Diehl, CasADi: A software framework for nonlinear optimization and optimal control, *Math. Program. Comput.* (2018), pp. 1–36.
40. A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* **106**, 25–57 (2006).

Acknowledgments: We thank Volkswagen Group Research including P. Hochrein, B. Mennenga, and T. Drescher for their support by providing research vehicles and test facilities, as well as fostering research collaboration and discussion. In addition, we thank the Volkswagen Electronics Research Lab for providing vehicle support, J. Subosits and V. Laurense for support with automated GTI, J. Goh for video editing, and Thunderhill Raceway Park for providing test facilities. We would like to thank the editor and our reviewers for their comments and help in improving the clarity of this paper. **Funding:** N.A.S. is supported by the NSF Graduate Research Fellowship and William R. and Sara Hart Kimball Stanford Graduate Fellowship. This material is based on work supported by the NSF Graduate Research Fellowship Program under grant no. DGE-1147470. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. **Author contributions:** N.A.S. conceived and designed data-based framework for automated vehicle control as well as coordinated and contributed to manuscript. M.B. aided in development of optimization-based control and helped in manuscript development. N.R.K. assisted with development and integration with physics-based control and aided in manuscript preparation. J.C.K. designed and analyzed the comparison between automated vehicle and human participant. J.C.G. aided in experimental design and manuscript preparation. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** We have supplied the relevant data and code (at https://purl.stanford.edu/zb950hd3384) to recreate the results with the exception of the experimental low-friction vehicle data. These data were obtained at a proprietary test facility, and at this moment, we are not free to post them. The relevant models trained on these data and results generated from these data are included so that experimental results can be validated.

Submitted 4 December 2018
Accepted 5 March 2019
Published 27 March 2019
10.1126/scirobotics.aaw1975

Citation: N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, J. C. Gerdes, Neural network vehicle models for high-performance automated driving. *Sci. Robot.* **4**, eaaw1975 (2019).

Science Robotics

Neural network vehicle models for high-performance automated driving

Nathan A. Spielberg, Matthew Brown, Nitin R. Kapania, John C. Kegelman, and J. Christian Gerdes

Sci. Robot. **4** (28), eaaw1975. DOI: 10.1126/scirobotics.aaw1975

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.