2021 IEEE Conference on Control Technology and Applications (CCTA).
August 8-11, 2021. San Diego, California

2021 IEEE Conference on Control Technology and Applications (CCTA).
August 8-11, 2021. San

2021 IEEE Conference on Control Technology and Applications (CCTA) | 978-1-6654-3643-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCTA48906.2021.9658972

Machine learning-based steering control for automated vehicles utilizing V2X communication

Sergei S. Avedisov, Chaozhe R. He, Dénes Takács, and Gábor Orosz

***Abstract*— A neural network-based controller is trained on data collected from connected human-driven vehicles in order to steer a connected automated vehicle on multi-lane roads. The obtained controller is evaluated using model-based simulations and its performance is compared to that of a traditional nonlinear feedback controller. The comparison of the control laws obtained by the two different approaches provides information about the naturalistic nonlinearities in human steering, and this can benefit the controller development of automated vehicles. The effects of time delay emerging from vehicle-to-everything (V2X) communication, computation, and actuation are also highlighted.**

I. Introduction

Autonomous driving is one of the most intricate challenges for automotive engineering. Engineers are facing complex, safety critical tasks including localization of the vehicle and obstacles based on multiple sensors and their fusion, decision making, path planning, and motion control of the vehicle. Each of the above mentioned tasks has its own bottleneck which requires solid research efforts. For example, localisation of the vehicle and obstacles around based on optical sensors can be problematic in inclement weather conditions. Decision making and path planning often suffers from the lack of human intuition. Motion control has to consider passenger comfort and safety while having limited information about road conditions and need to take into account time delays in the control loops.

In this paper, we focus on scenarios where wireless vehicle-to-everything (V2X) communication can provide solutions for the above mentioned problems. We consider the case when a connected automated vehicle (CAV) follows a connected human-driven vehicle (CHV) along a highway while receiving V2X messages (containing GPS position, speed and heading angle [1]); see Fig. 1. In this case the CAV may utilize the trajectory of the CHV as a "target path" and such strategy can be particularly beneficial during adverse weather conditions when the performance of optical sensors deteriorate. Rather than constructing separate decision making, planning, and control algorithms, here we take an "end-to-end" approach [2], [3]. Namely, V2X data collected from human-driven vehicles is used to train a neural network and the corresponding controller is deployed on the CAV. This approach allows us to capture the naturalistic behavior of (attentive) human drivers and may enhance performance in terms of safety and passenger comfort.

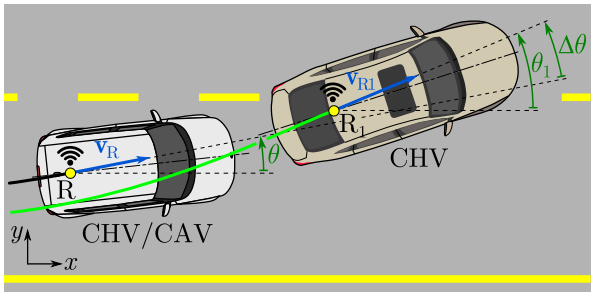

Fig. 1. The physical layout of the connected vehicle system. A connected human-driven (CHV) vehicle is driven along a straight road while executing lane change maneuvers and followed by another connected vehicle (CHV/CAV). Data is collected when the following vehicle is driven by a human driver and the collected data is used to train a neural network-based controller to steer the vehicle when it is automated.

While traditionally control design relies on a detailed vehicle model, supervised learning is emerging as a viable alternative. In particular, neural networks have the capability of fitting complex functions [4], which is beneficial when learning human behavior [5]. The performance of the controller can be still evaluated by using a first principle-based vehicle model, similar to the approach in [6], [7]. Comparing the neural network-based controller to a traditional nonlinear feedback controller allows us to gain some intuition of how the trained neural network steers the vehicle, which can be used to enhance the traditional nonlinear controllers. In this paper, beyond the comparison of the control laws, we also analyze the robustness of the controllers against time delay originated in communication, computation and actuation.

The rest of the paper is organized as follows. The data collection and the neural network architecture including training and hyperparameter tuning are discussed in Sec. II. A first principle-based vehicle handling model used for evaluation and the traditional nonlinear feedback controller design are presented in Sec. III. The two different control approaches are compared in Sec. IV where the advantages of the new approach are highlighted. We conclude the paper in Sec. V and lay out some future research directions.

S. S. Avedisov, C. R. He and G. Orosz are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA {avediska,hchaozhe,orosz}@umich.edu

D. Takács is with the Department of Applied Mechanics, Budapest University of Technology and Economics, H-1111, Hungary takacs@mm.bme.hu

S. S. Avedisov is also with the Toyota Motor North America R&D – Infotech Labs, Mountain View, CA 94043, USA

C. R. He is also with the Navistar Inc., Lisle, IL 60532, USA

D. Takács is also with the MTA-BME Research Group on Dynamics of Machines and Vehicles, Budapest, H-1111, Hungary

G. Orosz is also with the Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, MI 48109, USA

II. Neural Network-based Steering Controller

In order to obtain a steering controller that utilizes the V2X communication, we use experimental data to fit a two-layer neural network. In this section we lay out the data collection and processing procedure followed by the neural network design and performance evaluation.

A. Data collection and processing

The data used both for training and testing is collected in real driving scenarios. Two production passenger vehicles (shown in Fig. 2(a)) were driven by human drivers and both vehicles were equipped with V2X on-board units (see Fig. 2(b)) that recorded motion information with 10 Hz sampling frequency, This included the GPS positions of the rear axle center points R and R_1 (see Fig. 1), the heading angles $\theta = \sphericalangle \mathbf{v}_{\mathrm{R}}$, $\theta_1 = \sphericalangle \mathbf{v}_{\mathrm{R1}}$, and the speeds $v = |\mathbf{v}_{\mathrm{R}}|$, $v_1 = |\mathbf{v}_{\mathrm{R1}}|$ of the vehicles. The GPS positions were converted to the positions $(x_{\mathrm{R}}, y_{\mathrm{R}})$ and $(x_{\mathrm{R1}}, y_{\mathrm{R1}})$ in the ground-fixed coordinate system whose x-axis is aligned with the lanes on the test track; see Fig. 1. The steering angle γ of the following vehicle was also collected through the CAN bus and it was synchronized with the GPS data.

The vehicles were driven on the straight highway segment of the Mcity test track (see Fig. 2(c)). The preceding vehicle carried out lane change maneuvers (green trajectory) and the following vehicle "copied" the motion (black trajectory) imitating low-visibility scenarios where human drivers use nearby vehicle's motion as guidance. Data were collected from 15 lane changes, including 8 changes to the left lane and 7 changes to the right lane. We use 7 left lane changes and 7 right lane changes to train the network, and reserve a single left lane change for evaluation. This results in total 206.3 seconds of data ($N = 2063$ data points) for training and validation as well as 18.3 seconds of data (183 data points) for testing.

Because we desire the neural network to steer the following vehicle with respect to the leading vehicle, we train it based on the relative positions, angles, and velocities. The features and the output of our neural network are defined as

$$\mathbf{X} = \begin{bmatrix} x_{\mathrm{R1}} - x_{\mathrm{R}} \\ y_{\mathrm{R1}} - y_{\mathrm{R}} \\ \sin\theta_1 - \sin\theta \\ v_1 - v \end{bmatrix}, \quad Y = \gamma\,, \tag{1}$$

respectively. To improve convergence of the training and to minimize the chance of the backpropagation algorithm being stuck at a local optimum, we normalize the features and the output:

$$\widetilde{\mathbf{X}} = \mathbf{G}_X(\mathbf{X} - \mathbf{X}_{\min}) - 1_4\,, \quad \widetilde{Y} = G_Y(Y - Y_{\min}) - 1\,. \tag{2}$$

Here $\mathbf{X}_{\min}$ collects the minimum values of the features in the training data, $Y_{\min}$ is the minimum value of the output, and 1_4 is a vector of 1's in $\mathbb{R}^4$. The matrix $\mathbf{G}_X \in \mathbb{R}^{4\times 4}$ and scalar G_Y are defined as

$$\mathbf{G}_X = \mathrm{diag}(\ldots, \frac{2}{X^{(j)}_{\max} - X^{(j)}_{\min}}, \ldots)\,, \quad G_Y = \frac{2}{Y_{\max} - Y_{\min}}\,, \tag{3}$$

where $\mathbf{X}_{\max}$ and $Y_{\max}$ correspond to the maximum values of the features and the output. The matrix $\mathbf{G}_X$ and scalar G_Y map the range of the features and the output to $[-1, 1]$.

B. Neural network design and performance evaluation

To achieve the desired steering control we use a single hidden layer neural network with a finite number of neurons. Such networks can approximate arbitrary continuous nonlinear functions while having a simple architecture [8].

The input is the scaled feature vector $\widetilde{\mathbf{X}} \in \mathbb{R}^4$, and the hidden layer contains M nodes – a hyperparameter which will need to be tuned. The scalar variable z_j for the j-th neuron in the hidden layer is then given by

$$z_j = \tanh\left(\mathbf{w}_j^{\mathrm{T}}\widetilde{\mathbf{X}} + b_j\right), \tag{4}$$

for $j = 1, \ldots, M$, where $\tanh(\cdot)$ is used as the activation function, while $\mathbf{w}_j \in \mathbb{R}^4$ and b_j contain the weights and the bias for neuron j. The output of the network is given by

$$\hat{\widetilde{Y}} = \sum_{j=1}^{M} W_j z_j + B = \sum_{j=1}^{M} W_j \tanh\left(\mathbf{w}_j^{\mathrm{T}}\widetilde{\mathbf{X}} + b_j\right) + B\,, \tag{5}$$

where W_j is the weight for neuron j and B is the bias.

Let us denote the square prediction error

$$E_t(\vartheta) = \left(\widetilde{Y}_t - \hat{\widetilde{Y}}_t\right)^2, \tag{6}$$

where $t = 1, \ldots, N$ stand for the discrete time identifying the t-th data point while ϑ collects the weights and biases $(\mathbf{w}_j, b_j, W_j, B)$. Let us use the mean square error as the objective function and seek for the weights and biases that minimize the objective, i.e.,

$$\vartheta^* = \arg\min_{\vartheta} E_{\mathrm{MSE}}(\vartheta) = \arg\min_{\vartheta} \frac{1}{N}\sum_{t=1}^{N} E_t(\vartheta)\,. \tag{7}$$

To solve this optimization problem, we consider two backpropagation methods: the gradient descent and the Levenberg-Marquardt algorithms.

When using gradient descent we initialize the weights and biases using a uniform random distribution and backpropagate ϑ according to

$$\vartheta_{k+1} = \vartheta_k - \alpha_k \nabla_\vartheta E_{\mathrm{MSE}}(\vartheta)\,, \tag{8}$$

where ∇_ϑ denotes the gradient, k counts the epochs, and $\alpha_k = 0.05/(1 + 0.05k)$ is the applied learning rate. The process is repeated until the validation error stops decreasing over several iterations or we surpass 200 epochs.

We also use the Levenberg-Marquardt algorithm [9], [10] which is a second-order algorithm similar to Newton's method. We back propagate ϑ according to

$$\vartheta_{k+1} = \vartheta_k - \left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \mu\mathrm{I}\right)^{-1}\mathbf{J}^{\mathrm{T}}\left(\mathbf{Y} - \hat{\mathbf{Y}}\right), \tag{9}$$

where $\mathbf{J} \in \mathbb{R}^{N\times(6M+1)}$ denotes the Jacobian of $E_{\mathrm{MSE}}(\vartheta)$ with the t-th row being $J_t = \nabla_\vartheta E_t(\vartheta)$, μ is an adjustable damping parameter, and $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ are column vectors with the t-th entry being $\widetilde{Y}_t$ and $\hat{\widetilde{Y}}_t$, respectively. When implementing this method we initialize the weights and biases

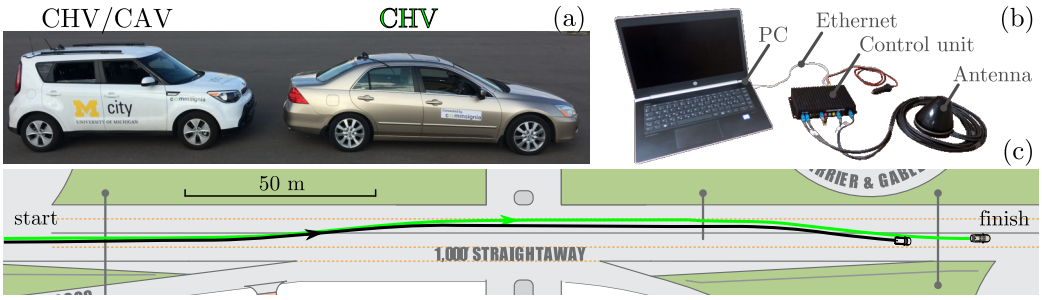

Fig. 2. Experimental setup. (a) Vehicles used in the experiments. The GPS antennas are mounted over the center of the rear axle of each vehicle. (b) V2X communication device used in the experiments. (c) Mcity test track with representative trajectories of experiments.

using the Nguyen-Widrow method [11]. This distributes the active regions of the M neurons more evenly over the feature space to yield faster convergence and smaller final mean square error.

A comparison between gradient descent and Levenberg-Marquardt algorithms is shown in Fig 3(a), for a network with $M = 20$. The Levenberg-Marquardt method achieves a root mean square error (RMSE) of 0.159 degrees for the steering angle in 40 epochs while gradient descent only achieves 0.478 degrees. In fact, the gradient descent still has an RMSE of over 0.350 degrees after 200 epochs of training. According to these results, we choose the Levenberg-Marquardt method in the MATLAB neural network toolbox to perform subsequent training, validation, and tuning.

In order to tune M we train and validate the network for various values of M between 1 and 100. In each case we randomly select 70% of the data for training and use the rest for validation. The results are shown in Fig. 3(b) where we applied the Levenberg-Marquardt backpropagation with the the Nguyen-Windrow initialization scheme. Both the training and the validation errors decrease with M but above $M = 25$ disparity between the training and validation errors becomes significant, indicating overfitting. Based on this we chose $M = 20$ and we retrain a network for 200 epochs yielding the final RMSE 0.152 degrees. This network is used for analysis and testing in the rest of the paper.

Traditionally, the performance of a neural network is evaluated by computing the RMSE for a randomly selected test run, in our case a lane change to the left. For this run we

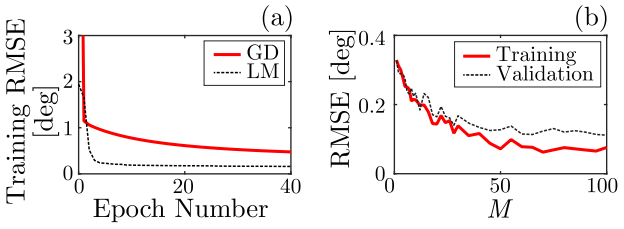

Fig. 3. (a) Comparison between the training errors of the gradient descent algorithm with random initilization and the Levenberg-Marquardt algorithm with Nguyen-Widrow initialization for $M = 20$ neurons in the hidden layer. (b) Training and validation errors for various M values using 70%-30% training-validation split in the Levenberg-Marquardt algorithm.

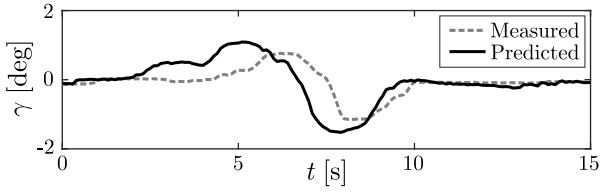

Fig. 4. Measured steering angle (gray dashed) vs steering angle predicted by (10) (black solid) for the test run.

predict the steering angle using the trained neural network:

$$\hat{\gamma} = Y_{\min} + \frac{B^* + 1}{G_Y} + \frac{1}{G_Y}\left(\sum_{j=1}^{M} W_j^* \tanh(\mathbf{w}_j^{*\mathrm{T}}(\mathbf{G}_X(\mathbf{X} - \mathbf{X}_{\min}) - 1_4) + b_j^*)\right), \tag{10}$$

cf. (2) and (5), where the asterisks indicate that we are using the optimized values of the weights and biases.

The time profiles of the measured and predicted steering angles are shown in Fig. 4 for the test run. The RMSE between the predicted and the measured signals is 0.424 degrees, which is significantly higher than the training RMSE for the final network. This is due to the human driver's varying reaction across different runs and that we did not use any points from the entire test run to train the network. More importantly, Fig. 4 shows that although at a given time the predicted and measured steering wheel angles have large deviations, the time profiles look qualitatively similar. This indicates that the RMSE may not be the best performance indicator for this application. Lastly, since the vehicle is a dynamic system, any difference between the measured and predicted steering angles will affect where the vehicle will be located subsequently. Thus, the actual features in the steering maneuver performed by the neural network will differ from the those measured during the test run.

In order to properly evaluate the performance of the neural network-based controller, we perform a dynamic simulations where the neural network steers the following vehicle. We construct a model-based simulation environment that uses first principle-based vehicle handling model. We also use this environment to construct a traditional nonlinear feedback controller that can serve as a basis of comparison for the neural network-based controller.

III. First Principle-based Feedback Design

Once using a neural network-based controller, it may be difficult to interpret the control law and to provide formal guarantees of stability and safety for the closed-loop system. To resolve this, we compare the controller designed above with a first principle-based nonlinear controller. In particular, we demonstrate that comparing the two different controllers can give information about the linear stability of the trained neural network-based controller.

A. Mechanical model

In order to describe the lateral dynamics of the vehicle involved in the experiments, we use the in-plane single-track vehicle model with front wheel drive shown in Fig. 5. The mass and the mass moment of inertia about the center of gravity G are denoted by m and J_{G}, respectively. The wheelbase of the vehicle is l while the distance of the center of gravity G from the rear axle is b; see Table I for the parameters used in this paper. The lateral forces at the front and rear tires are denoted by F_{F} and F_{R}, respectively, and we neglect the effects of the self-aligning moments of the tires. The motion of the vehicle is described by three generalized coordinates: the position coordinates x_{R} and y_{R} of the center point R of the rear axle and the yaw angle ψ. The steering angle γ is considered as time dependent parameter, an input that we can assign.

In our model we consider constant driving speed V, namely, the longitudinal speed of the front wheel is kept constant. This leads to the kinematic constraint:

$$\dot{x}_{\mathrm{R}}\cos(\psi+\gamma)+\dot{y}_{\mathrm{R}}\sin(\psi+\gamma)+l\dot{\psi}\sin\gamma=V\,. \tag{11}$$

To derive the equations of motion of the vehicle, we apply the Appell-Gibbs formalism (see [12], [13]), which requires us to choose two so-called pseudo velocities. Here we choose σ (the lateral velocity of the rear axle center point R) and ω (the yaw rate), that is,

$$\sigma=-\dot{x}_{\mathrm{R}}\sin\psi+\dot{y}_{\mathrm{R}}\cos\psi\,,\quad \omega=\dot{\psi}\,. \tag{12}$$

From (11,12) the generalized velocities can be expressed as

$$\begin{aligned}\dot{x}_{\mathrm{R}}&=V\frac{\cos\psi}{\cos\gamma}-\sigma\frac{\sin(\psi+\gamma)}{\cos\gamma}-l\,\omega\cos\psi\tan\gamma\,,\\ \dot{y}_{\mathrm{R}}&=V\frac{\sin\psi}{\cos\gamma}+\sigma\frac{\cos(\psi+\gamma)}{\cos\gamma}-l\,\omega\sin\psi\tan\gamma\,,\\ \dot{\psi}&=\omega\,.\end{aligned} \tag{13}$$

Without going into details, the Appell equations become

$$\begin{bmatrix} m/\cos^2\gamma & m(b+l\tan^2\gamma)\\ m(b+l\tan^2\gamma) & J_{\mathrm{G}}+m(b^2+l^2\tan^2\gamma)\end{bmatrix}\begin{bmatrix}\dot{\sigma}\\ \dot{\omega}\end{bmatrix}=\begin{bmatrix}f_1\\ f_2\end{bmatrix}, \tag{14}$$

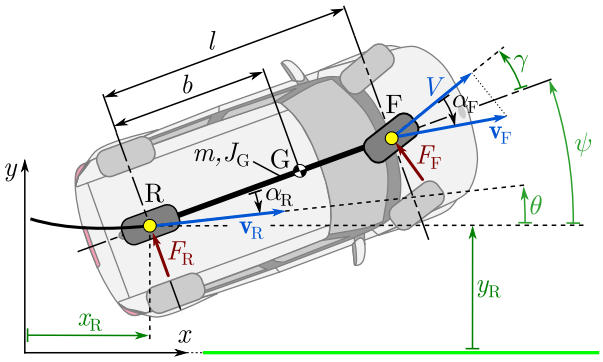

Fig. 5. Single track vehicle model.

where the right hand side can be expressed as

$$\begin{aligned}f_1&=\frac{F_{\mathrm{F}}}{\cos\gamma}+F_{\mathrm{R}}-\frac{m}{\cos\gamma}\Big(V-(l-b)\omega\sin\gamma\Big)\omega\\ &\quad+m\frac{\tan\gamma}{\cos^2\gamma}\Big(V\sin\gamma-\sigma-l\,\omega\Big)\dot{\gamma}\,,\\ f_2&=\frac{F_{\mathrm{F}}\,l}{\cos\gamma}-\frac{m}{\cos\gamma}\Big(bV+(l-b)\sigma\sin\gamma\Big)\omega\\ &\quad+m\,l\frac{\tan\gamma}{\cos^2\gamma}\Big(V\sin\gamma-\sigma-l\,\omega\Big)\dot{\gamma}\,.\end{aligned} \tag{15}$$

The lateral tire forces F_{F} and F_{R} are calculated via the brush tire model (see Appendix) as function of the side slip angles α_{F} and α_{R} of the front and rear tires, which can be determined based on vehicle kinematics:

$$\begin{aligned}\tan\alpha_{\mathrm{F}}&=-\frac{\sigma+l\,\omega}{V\cos\gamma}+\tan\gamma\,,\\ \tan\alpha_{\mathrm{R}}&=\frac{\sigma\cos\gamma}{-V+(\sigma+l\,\omega)\sin\gamma}\,.\end{aligned} \tag{16}$$

From the side slip angle of the rear wheel, the heading angle can be calculated as $\theta=\psi-\alpha_{\mathrm{R}}$; see Fig. 5.

The system (13,14) with expressions (15,16) results in 5 first order nonlinear differential equations for the state variables $(x_{\mathrm{R}},y_{\mathrm{R}},\psi,\sigma,\omega)$, controlled by the input γ that shall be assigned by the controller; see (10) or the traditional nonlinear controller designed below.

B. Traditional nonlinear controller

Here we construct a simple nonlinear feedback controller that only utilize the lateral positions and the heading angles:

$$u=k_y(y_{\mathrm{R1}}-y_{\mathrm{R}})+k_\theta\big(\sin\theta_1-\sin\theta\big)\,, \tag{17}$$

(cf. Fig. 1), and we assign the steering angle according to

$$\gamma(t)=\arctan u(t-\tau)\,. \tag{18}$$

TABLE I
VEHICLE PARAMETERS

	Parameter name	Value	Unit
l	wheelbase	2.57	m
b	center of gravity position	1.54	m
m	vehicle mass	1770	kg
J_{G}	vehicle mass moment of inertia	1343	kgm^2
V	longitudinal velocity	15	m/s
a	tire-ground contact half-length	0.1	m
k	lateral tire stiffness	2×10^6	N/m

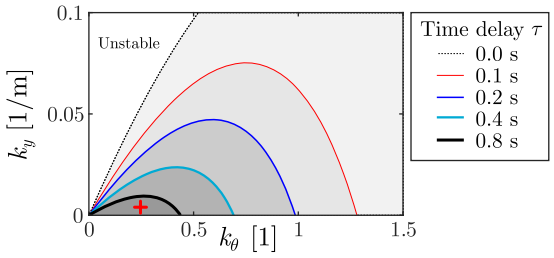

Fig. 6. Stability charts in the (k_θ, k_y) plane. The colored curves represent the theoretical stability boundaries for different values of the time delay. The red cross indicates the gain parameters used when testing the controller; see gray mesh in Fig. 7(a) and thin dotted curves in Fig. 7(b,c).

Here τ stands for the time delay that arises from communication, computation, state estimation, and actuation.

We remark that we derived the neural network-based controller from human-driving data without taking into account the time delay (as the actual value of the delay is not known), but when applying the controller to steer the CAV the delay still appears in the control loop, i.e.,

$$\gamma(t) = \hat{\gamma}(t-\tau)\,, \tag{19}$$

where $\hat{\gamma}$ is given by (10).

We investigate the linear stability of the traditional controller (17,18) in terms of the gain parameters k_y and k_θ. For simplicity, we focus on the scenario where the leading vehicle is driving along the x-axis, that is, we substitute $y_{\mathrm{R1}} \equiv 0$ and $\theta_1 \equiv 0$ into (17). The closed-loop dynamics of the following vehicle are described by (13,14,17,18) that can be written into the form of a delay differential equation of a neutral type. After the linearization around the equilibrium $(x_{\mathrm{R}} = Vt, y_{\mathrm{R}} \equiv 0, \psi \equiv 0, \sigma \equiv 0, \omega \equiv 0)$, the characteristic function $D(\lambda)$ of the resulting linear delay differential equation can be determined analytically. The characteristic equation $D(\lambda) = 0$ has infinitely many solutions for the characteristic roots $\lambda \in \mathbb{C}$. $D(0) = 0$ provides the stability boundary $k_y > 0$ for non-oscillatory stability loss. The boundary related to the oscillatory stability loss can be calculated from $D(\mathrm{i}\Omega) = 0$ via the D–subdivision method; see [14]. These boundaries encapsulate a domain in the parameter space where all characteristic roots have negative real parts. In Fig. 6 linear stability chart are constructed in the plane of the control gains k_θ and k_y for different values of the delay τ as indicated. Observe that the shaded linearly stable region shrinks as the time delay is increased.

IV. Testing and Comparison

In order to evaluate the performance of neural network-based controller, we compare it to the traditional nonlinear controller. We compare the "geometry" of the control laws as well as the simulation performance when applying the controllers to the model developed in Sec. III-A.

Fig. 7(a) compares the control law of neural network controller (10,19) (colored surface) to the traditional controller (17,18) (gray mesh) by plotting the steering angle as a function of the lateral distance $\Delta y = y_{\mathrm{R1}} - y_{\mathrm{R}}$ and

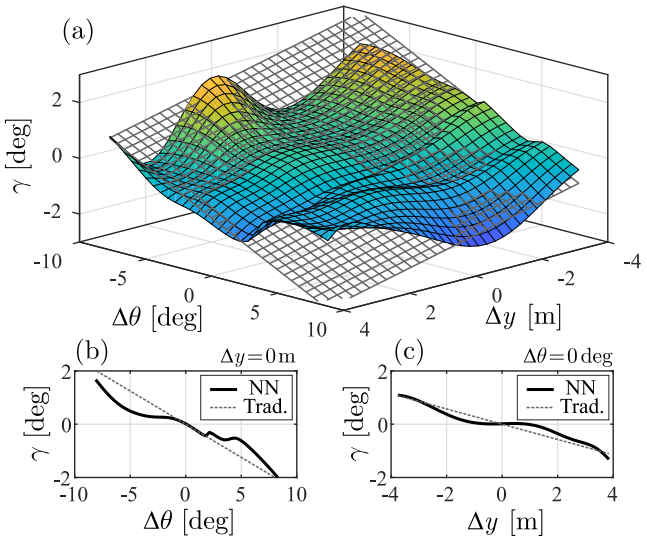

Fig. 7. Comparison between the control laws of the neural network-based controller (colored surface in panel (a) and thick solid curves in panels (b,c)) and the traditional nonlinear controller (gray mesh in panel (a) and thin dotted lines in panels (b,c)). The gains for the traditional controller are indicated by the red cross in Fig. 6.

the heading angle difference $\Delta\theta = \theta_1 - \theta$. We plot slices at $\Delta y = 0$ and at $\Delta\theta = 0$ in Fig. 7(b) and (c), respectively. For the other two features of the neural network, we set the longitudinal distance to $\Delta x = x_{\mathrm{R1}} - x_{\mathrm{R}} = 28$ m and the speed difference to $\Delta v = v_1 - v = 0$ (cf. (1)) that are close to the average values observed during data collection. For the traditional controller we use the gain parameters $k_\theta = 0.25$ and $k_y = 0.005$ 1/m that ensure linear stability even for $\tau = 0.8$ s; see red cross in Fig. 6. We remark we have $\sin\theta_1 - \sin\theta \approx \theta_1 - \theta$ in (1).

Panel (a) shows that the overall trend of the two surfaces is similar, indicating that the trained controller follows a similar logic as the traditional controller. Larger differences can be observed between the surfaces for larger values of Δy and $\Delta\theta$. Moreover, the curves in panels (b) and (c) reveal that the neural network-based controller is less sensitive when the lateral distance and heading angle difference are small and is more "aggressive" when errors are large. This indicates a strengths of the neural network-based approach: it is able to capture the naturalistic nonlinearities in human steering. In the meantime, the linear stability is ensured for the neural network-based controller since for small deviations it applies similar gains as the linearly stable traditional controller.

In order to illustrate how the controllers perform lane changes in dynamic situations we utilize the mechanical model (13,14) in numerical simulations with parameters given in Table I. We let the controllers to steer the model of the following vehicle. For the leading car's trajectory, we use data from a separate experiment. When applying the neural-network based controller (10,19), we consider the constant speeds $v \approx V = v_1 = 15$ m/s yielding $\Delta v = v_1 - v = 0$ and the constant longitudinal distance $\Delta x = x_{\mathrm{R1}} - x_{\mathrm{R}} = 28$ m. For the traditional nonlinear controller (17,18) we use the gains $k_\theta = 0.25$ and $k_y = 0.005$ 1/m; cf. red cross in Fig. 6.

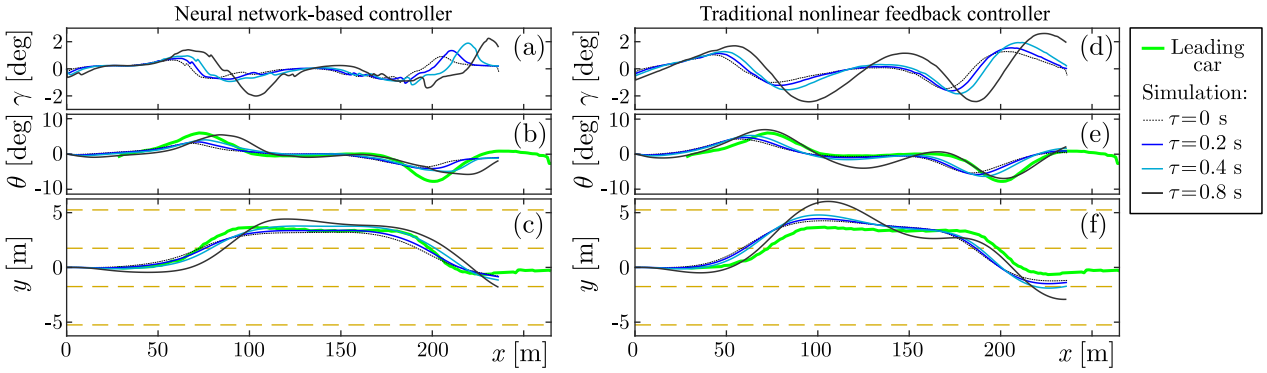

Fig. 8. Simulation results for the neural network and traditional controller with different time delays.

Indeed, we also include the time delay τ in the simulations for both controllers.

Figure 8 compares the simulation results. The trajectories are plotted for different time delays as indicated. In case of the neural network-based controller the vehicle follows the leader's trajectory with good accuracy while larger deviations are observed when applying the traditional controller, especially for larger delay values. That is, the neural network-based controller outperforms the traditional controller and also shows better robustness against the time delays. For the neural network-based controller one may observe that the heading angle is not well aligned with the lane direction at the end section (after the second lane change). This is likely caused by the fact that at this section the leading vehicle drops its speed (as it reaches the end of the track), that is, the constant speed and constant longitudinal distance assumptions used in the simulations do not hold any more.

V. Conclusion

In this paper we trained a neural network-based controller to control the motion of a connected automated vehicle based on V2X information received from nearby vehicles. We demonstrated that the trained controller can capture the naturalistic nonlinear behavior of human drivers and it outperform a traditional nonlinear controller in our simulations in terms accuracy and robustness against time delays in the control loop. Hence, controllers can be improved in the design stage by utilizing the main characteristics of control laws learned from human data.

Appendix

In case of the brush tire model [15], the lateral tire force can be given as functions of the tire slip angle α:

$$F(\alpha) = \begin{cases} \phi_1 \tan\alpha + \phi_2 \operatorname{sgn}\alpha \tan^2\alpha + \phi_3 \tan^3\alpha, & 0 \leq |\,\alpha\,| < \alpha_{\text{crit}}, \\ \mu F_z \operatorname{sgn}\alpha, & \alpha_{\text{crit}} < |\,\alpha\,|, \end{cases} \tag{20}$$

where the critical side-slip angle is $\alpha_{\text{crit}} = \frac{3\mu_0 F_z}{2a^2 k}$. Here a is the half-length of the tire-ground contact patch, k is the distributed lateral stiffness of the tire, F_z is the vertical load on the axle, μ and μ_0 are the coefficients of friction for sliding and rolling, respectively, and we have the coefficients

$$\phi_1 = 2a^2 k, \quad \phi_2 = -\frac{4a^4 k^2}{3\mu_0 F_z}\left(2 - \frac{\mu}{\mu_0}\right), \quad \phi_3 = \frac{8a^6 k^3}{9\mu_0^2 F_z^2}\left(1 - \frac{2\mu}{3\mu_0}\right). \tag{21}$$

Acknowledgement

This research was supported by the Mobility Transformation Center at the University of Michigan and by the National Research, Development, and Innovation Office of Hungary under grant no. NKFI-128422. Dénes Takács would like to thank the Rosztoczy Foundation for their generous support.

References

[1] G. Orosz, J. Ge, C. He, S. Avedisov, W. Qin, and L. Zhang, "Seeing beyond the line of sight - controlling connected automated vehicles," *Mechanical Engineering Magazine, Dynamic System & Control*, vol. 5, no. 4, pp. 8–12, December 2017.

[2] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.

[3] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *33rd AAAI Conference on Artificial Intelligence*, vol. 54, 2019, pp. 3387–3395.

[4] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang, "Learning polynomials with neural networks," in *International Conference on Machine Learning*, 2014, pp. 1908–1916.

[5] S. Bae, D. Saxena, A. Nakhaei, C. Choi, K. Fujimura, and S. Moura, "Cooperation-aware lane change maneuver in dense traffic based on model predictive control with recurrent neural network," in *2020 American Control Conference (ACC)*, 2020, pp. 1209–1216.

[6] M. A. Ashraf, J. Takedaa, and R. Toorisu, "Neural network based steering controller for vehicle navigation on sloping land," *Engineering in Agriculture, Environment, and Food*, vol. 3, no. 3, pp. 100–104, 2010.

[7] G. Han, W. Fu, W. Wang, and Z. Wu, "The lateral tracking control for the intelligent vehicle based on adaptive PID neural network," *Sensors*, vol. 17, no. 6, p. 10.3390/s17061244, 2017.

[8] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, no. 2, pp. 303–314, 1989.

[9] A. Rangathan, "The Levenberg-Marquardt algorithm," http://users-phys.au.dk/jensjh/numeric/project/10.1.1.135.865.pdf, Aarhus University, Tech. Rep., 2004.

[10] J. Moreé, "The Levenberg-Marquardt algorithm: implementation and theory," *Numerical Analysis*, pp. 105–116, 1978.

[11] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by chosing initial values of the adaptive weights," in *International Joint Conference on Neural Networks*, 1990, pp. 21–26.

[12] F. Gantmacher, *Lectures in Analytical Mechanics*. MIR Publishers, 1975.

[13] V. De Sapio, *Advanced Analytical Dynamics: Theory and Applications*. Cambridge University Press, 2017.

[14] G. Stépán, *Retarded Dynamical Systems: Stability and Characteristic Functions*. Longman, 1989.

[15] H. Pacejka, *Tire and Vehicle Dynamics*. Elsevier, 2012.