2025 IEEE Intelligent Vehicles Symposium (IV)
June 22-25, 2025. Cluj-Napoca, Romania

Learning Teleoperated Driving Behavior from Limited Trajectory Data

Xunbi A. Ji, Sergei S. Avedisov, Illés Vörös, Mohammad Irfan Khan, Onur Altintas, and Gábor Orosz

***Abstract*— In this paper, we propose models with explicit trainable delays for learning teleoperated driving (ToD) behavior from limited vehicle trajectory data. The data-driven model is integrated with physics-based nonlinear vehicle dynamics and formulated as a neural delay differential equation (NDDE). The model can be analyzed using the same tools as developed for classical delay differential equations. The physics-based nonlinearity built into the data-driven model reduces the model complexity, enables training with limited data, and provides good generalizations. An overall latency in the loop is learned and a generic steering controller that characterizes the remote operator is identified at the same time through the training process. This information could be used to evaluate the performance of ToD in the presence of communication latency. We provide examples of learning from simulation data generated by a kinematic vehicle model and from experimental data generated by a human operator driving in a high-fidelity simulation environment. The same data-driven model and training algorithm is used in both cases, which demonstrates the generalizability of the proposed approach.**

***Index Terms*— Teleoperated driving, delayed lateral dynamics, data-driven models**

I. Introduction

Latency is an important component in vehicle dynamics and control [1], [2]. The reaction time of the driver and the actuation time of the powertrain systems both contribute to the overall latency in the closed-loop vehicle dynamics [3]. These latencies may lead to unstable behaviors, which in turn compromise safety, energy efficiency, and passenger comfort [4]. It was conjectured that vehicle automation will reduce latency, but recent experimental tests suggest that highly automated vehicles in fact respond slower than attentive human drivers [5], [6]. This is because on-board sensors and computation units introduce new sources of latency [7], [8]. Teleoperated driving (ToD) can act as a supplement to automated driving in a variety of scenarios. In teleoperated driving, a communication network is used to transmit sensor data from the vehicle to a remote center and to relay control commands back to the vehicle [9], [10]. This introduces communication latency into the control system, in addition to the vehicle actuation latency and the remote operator processing time, as illustrated in Fig. 1(a).

Xunbi A. Ji, Illés Vörös and Gábor Orosz are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA. {xunbij, illesvoe, orosz}@umich.edu.

Sergei S. Avedisov, Mohammad Irfan Khan, and Onur Altintas are with Toyota Motor North America R&D – InfoTech Labs, Mountain View, CA 94043, USA. {sergei.avedisov, mohammad.irfan.khan, onur.altintas}@toyota.com.

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

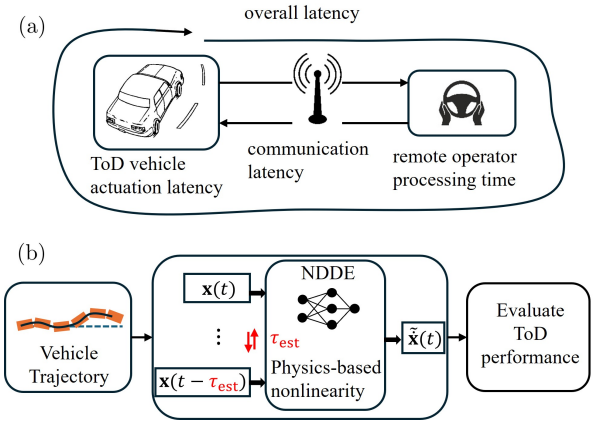

Fig. 1. (a) Latency components in the teleoperated control loop. (b) A flow chart of learning and evaluating the teleoperated driving behaviors. The physics-enhanced data-driven model is in form of neural delay differential equations (NDDE) with trainable delay.

These inevitable latencies take place in the ToD control loop in a sequential manner, therefore, one may model the cumulative effect using the overall latency, i.e., the sum of the three latency components. Since the overall latency impacts the control performance [4], it is important to estimate its value. Additionally, the latency and the behavior of the remote operator can change over time and the ToD model should adapt to such changes while relying on limited data.

Latencies can be estimated using least squares method [11] or using Bayesian inference [12]. Sparse identification of nonlinear dynamics (SINDy) can also be extended for time delay systems [13], [14] to estimate the model parameters including the delay. With the knowledge of the delay, many mitigation strategies can be applied to improve control performance [15]–[17]. The time delays are also considered implicitly in recurrent neural network models, as long as the model provides good predictions [18]. Typically, models without prior-knowledge are more flexible but they require more training data and the simulations are time-consuming.

We implement a learning framework which features a model with an explicit representation of the overall latency. The overall latency and model parameters are learned simultaneously from limited trajectory data. The learning process is illustrated in Fig. 1(b), where an explicit latency is incorporated in a neural delay differential equation (NDDE), which is a data-driven model in the form of a delay differential equation [19], [20]. The inputs to the NDDE are the current and delayed states of the ToD system while the output is the prediction of the state derivative at the current time. We format the NDDE as a neural network integrated with the physics-based nonlinear vehicle dynamics. Having trainable

2025 IEEE Intelligent Vehicles Symposium (IV) | 979-8-3315-3803-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/IV64158.2025.11097642

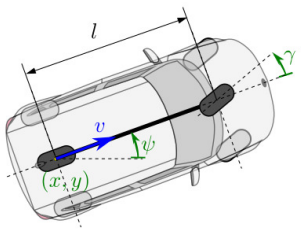

Fig. 2. Kinematic bicycle model utilized to describe the vehicle dynamics.

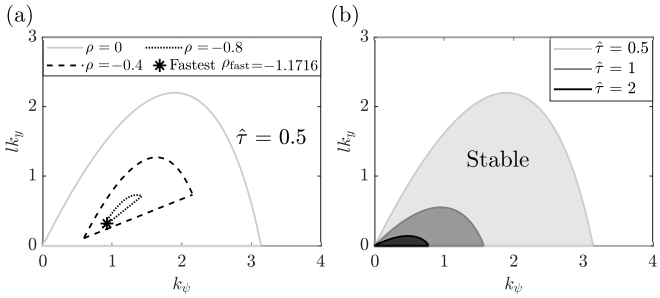

Fig. 3. (a) Contours for different convergence rates ρ when $\hat{\tau} = 0.5$. (b) Stability boundaries for different scaled delays $\hat{\tau}$.

delay instead of discretizing history simplifies the structure of the neural networks; provides physical meaning to the delay and allows detailed analysis of the model.

Integrating physics-based nonlinearity into a data-driven model reduces the model complexity, and thus, the delayed dynamics can be learned from limited data while the model generalizes well. The learned model characterizes the dynamics of the vehicle and the remote operator, and can be potentially viewed as a digital twin of the teleoperated vehicle-operator system [21], [22]. While the goal of digital twins typically is to create high-fidelity simulation environments, our goal is to establish the simplest nontrivial model for delayed ToD dynamics with the remote operator in the loop. This model shall enable the evaluation of the ToD performance (without further simulations) after the model is trained. This provides insights into ToD performance and may facilitate real-time adjustments for the communication network or the remote operator.

The rest of the paper is organized as follows. In Section II, we introduce the physics-based nonlinear vehicle model for ToD with a generic steering angle controller and establish the stability and performance analysis. In Section III we design the data-driven model and introduce the training algorithm for learning the overall latency and the control law. We provide examples of learning the delayed dynamics from limited vehicle trajectory data in Section IV, including experimental data with a human operator driving in the TELECARLA simulation environment. In the end, we summarize the results and provide future directions in Section V.

II. Teleoperated Driving System

In this section, we model the teleoperated driving (ToD) system in the presence of latency and provide stability and performance analysis while utilizing a simple nonlinear vehicle model [23]. This will help the construction of the data-driven model in Section III and lays the foundation of quantifying ToD performance with a human operator in the loop in Section IV.

Consider the single track model of a vehicle depicted in Fig. 2, where v is the longitudinal velocity and l is the wheelbase. The coordinates of the rear axle center are denoted by (x, y), while ψ is the yaw angle of the vehicle, and γ is the steering angle. The dynamics of the vehicle are

$$\begin{aligned} \dot{x}(t) &= v\cos(\psi(t)), \\ \dot{y}(t) &= v\sin(\psi(t)), \\ \dot{\psi}(t) &= \frac{v}{l}\tan\big(\gamma(t)\big). \end{aligned} \tag{1}$$

For the scenario of following a straight path $y^* \equiv 0$, $\psi^* \equiv 0$, a generic steering controller is given by

$$\gamma(t) = \mathrm{sat}\big(-k_y y(t-\tau) - k_\psi \psi(t-\tau)\big), \tag{2}$$

where the control gain k_y represents the sensitivity to the lateral error and the control gain k_ψ represents the sensitivity to the yaw error. The delay τ is the overall latency in the loop and the saturation function $\mathrm{sat}(\cdot) = \frac{1}{\pi}\arctan(\pi\cdot)$ limits the steering angle between ± 28.65 degrees, same as in [24].

One can nondimensionalize time and space in (1)-(2) using $\hat{t} = tv/l$, $\hat{\tau} = \tau v/l$, $\hat{y} = y/l$, $\hat{x} = x/l$, and then linearize around the steady state $\hat{x}^* \equiv \hat{t} + \hat{x}(0)$, $\hat{y}^* \equiv 0$, $\psi^* \equiv 0$, see [15] for details. The stability of lateral dynamics is given by the characteristic equation

$$\lambda^2 + k_\psi \mathrm{e}^{-\lambda\hat{\tau}}\lambda + l\,k_y \mathrm{e}^{-\lambda\hat{\tau}} = 0, \tag{3}$$

which has infinitely many solutions for the characteristic roots λ. The real part ρ of the rightmost roots $\lambda = \rho \pm \mathrm{j}\omega$ are used to determine stability and the convergence rate. When $\rho < 0$, the steady state motion is stable, that is, the vehicle converges to straight-running motion. The more negative value the ρ is, the faster the vehicle converges.

Substituting $\lambda = \rho$ (i.e., $\omega = 0$) into equation (3) yields

$$l\,k_y = -\rho \mathrm{e}^{\rho\hat{\tau}}(\rho + k_\psi \mathrm{e}^{-\rho\hat{\tau}}), \tag{4}$$

which is a straight line in the $(k_\psi, l\,k_y)$ plane. When $\omega > 0$, we obtain

$$\begin{aligned} k_\psi &= -\mathrm{e}^{\rho\hat{\tau}}\big((\rho^2-\omega^2)\sin(\omega\hat{\tau}) + 2\rho\omega\cos(\omega\hat{\tau})\big)/\omega, \\ l\,k_y &= -\mathrm{e}^{\rho\hat{\tau}}\big((\rho^2-\omega^2)\cos(\omega\hat{\tau}) - 2\rho\omega\sin(\omega\hat{\tau})\big) - k_\psi\rho, \end{aligned} \tag{5}$$

which describes a curve in the $(k_\psi, l\,k_y)$ plane that is parameterized by the angular frequency ω. If the control gains are selected such that $(k_\psi, l\,k_y)$ is on the curves, then the system will converge to the equilibrium with corresponding convergence rates ρ. These curves are plotted for different ρ values in Fig. 3(a) when using the scaled latency $\hat{\tau} = 0.5$. The region enclosed by the curves shrinks as ρ decreases and it disappears when the slopes of the two curves coincide at the bottom left corner. Correspondingly, there exists a fastest convergence rate

$$\rho_{\mathrm{fast}} = \frac{\sqrt{2}-2}{\hat{\tau}}, \tag{6}$$

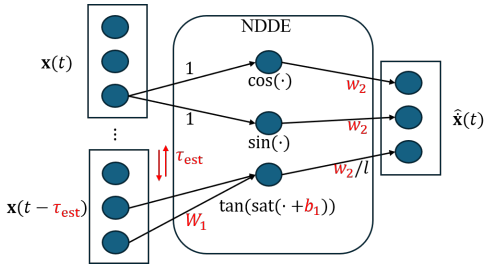

Fig. 4. Structure of the NDDE. The trainable parameters are indicated in red. As the trainable delay in the NDDE updates, the delayed input to the NDDE changes accordingly.

which depends only on the scaled latency $\hat{\tau} = \tau v/l$. The corresponding control gains are given by

$$\begin{aligned} k_{\psi,\text{fast}} &= \frac{\mathrm{e}^{\sqrt{2}-2}\left(2\sqrt{2}-2\right)}{\hat{\tau}}, \\ l\,k_{y,\text{fast}} &= \frac{\mathrm{e}^{\sqrt{2}-2}\left(10\sqrt{2}-14\right)}{\hat{\tau}^2}, \end{aligned} \tag{7}$$

indicated by the black star in Fig. 3(a).

For the special case $\rho = 0$, the curves correspond to the stability boundaries

$$l\,k_y = 0, \tag{8}$$

and

$$\begin{aligned} k_\psi &= \omega \sin \omega\hat{\tau}, \\ l\,k_y &= \omega^2 \cos \omega\hat{\tau}. \end{aligned} \tag{9}$$

The system is stable only if the control gains are inside the region enclosed by stability boundaries. The stable region shrinks as $\hat{\tau}$ increases, as shown in Fig. 3(b).

Therefore, the performance of ToD can be characterized by the scaled latency and the steering control gains. The scaled latency determines the stability boundaries and the fastest convergence rate of the system. The location of the actual control gains with respect to the stability boundaries and to the fastest-convergence gains characterizes the actual performance in convergence.

The data-driven model introduced in Section III is constructed in the form (1)-(2), so that the neural network parameters learned from data are interpretable. We will show that if the training data is generated by (1)-(2), the equivalent data-driven model can exactly recover all the parameters, including the latency. Then we will train the same model using experimental data generated by a human operator driving in a high-fidelity vehicle simulation environment to demonstrate that such interpretability can be retained. This also allows us to evaluate the performance of the ToD system, including the remote operator.

III. Data-driven Model for Delayed Dynamics

In this section, we utilize the physics-based form (1)-(2) to construct a neural delay differential equation (NDDE). Then we apply the algorithm for learning the dynamics of the vehicle and the remote operator, including the overall latency. The learned weights and biases carry physical meanings, like the longitudinal velocity and the steering control gains.

Consider the following neural delay differential equation

$$\dot{\mathbf{x}}(t) = \text{NDDE}\big(\mathbf{x}(t), \mathbf{x}(t-\tau_{\text{est}})\big), \tag{10}$$

with $\mathbf{x} = [x, y, \psi]^\top$ and

$$\text{NDDE} := \begin{bmatrix} w_2 \cos\left(\mathbf{x}_3(t)\right) \\ w_2 \sin\left(\mathbf{x}_3(t)\right) \\ \frac{w_2}{l} \tan\Big(\text{sat}\big(W_1 \mathbf{x}_{\mathbf{2:3}}(t-\tau_{\text{est}}) + b_1\big)\Big) \end{bmatrix}, \tag{11}$$

where the wheelbase l is assumed to be known and the rest of the NDDE parameters are the weights $w_2 \in \mathbb{R}$, $W_1 = [w_{11}, w_{12}] \in \mathbb{R}^{1\times 2}$, the bias $b_1 \in \mathbb{R}$, and the overall latency $\tau_{\text{est}} \in \mathbb{R}$. This model is built based on the kinematic model (1)-(2) and thus the NDDE parameters have physical meanings. For example, w_2 represents the longitudinal velocity, $-W_1$ represents the control gains in the steering controller and τ_{est} represents the overall latency in the loop. The bias b_1 represents the lateral offset of the straight path $y^* = -b_1/w_{11}$, which may not be zero when the remote operator is a human (who may follow the straight path with some constant lateral deviation). The structure of NDDE (11) is visualized in Fig. 4. The input data is the state trajectory and the output is the time derivative of the state, which can be estimated by numerical differentiation. The learning algorithms developed for NDDEs [20], [25] can be adopted.

Here the derivative loss function

$$\mathcal{L} = \frac{1}{Nn} \sum_{j=1}^{N} \sum_{i=1}^{n} \Big(\tilde{\dot{\mathbf{x}}}_i(t_j) - \dot{\mathbf{x}}_i(t_j)\Big)^2, \tag{12}$$

is applied in training where N is batchsize (the number of samples used for one update) and n is the dimension of the NDDE state $\mathbf{x}(t)$, in our case $n = 3$. The predicted state derivative

$$\tilde{\dot{\mathbf{x}}}(t_j) = \text{NDDE}\big(\mathbf{x}(t_j), \mathbf{x}(t_j - \tau_{\text{est}})\big), \tag{13}$$

is directly calculated from data. The gradients of the loss with respect to the parameters can be used for parameter update, see [25] for detailed derivations. One may also simulate the NDDE in the training process and use simulation error in the loss function [20]. Here, for the sake of computation efficiency, we use the derivative loss (12).

We follow a parallel-training algorithm developed in [25] and use the adaptive moment estimation `adamupdate` from MATLAB Deep Learning Toolbox for the parameter update. Multiple initializations balance the exploration and exploitation of parameter space, which is helpful when there exist multiple local minima. (This can happen when the data is not rich enough in terms of dynamics for training the model). All parameters, including the delay, are continuous, although the data is only provided at discrete time instances. Therefore, linear interpolation is used to obtain the delayed state at these time instances. Moreover, positivity and maximum value constraints are imposed on the delay parameter τ_{est}.

IV. Learning the Delayed ToD Dynamics from a Single Trajectory

In this section, we provide two examples of using NDDE to learn the ToD behavior from a single trajectory. In each example, the data-driven model has the same form (11), while the trajectory data are generated differently.

A. Learning from simulation data of the kinematic model

We generate a 10-second-long training trajectory using the kinematic model (1)-(2) with $l = 2.9$ m, $k_y = 0.2$ m^{-1}, $k_\psi = 1$ and constant velocity $v = 2$ m/s. The overall latency is $\tau = 1$ s and the vehicle aims to follow the straight path $y^* \equiv 0$, $\psi^* \equiv 0$. The sampling time is 0.1 second and thus only 100 data points are available for training, see Fig. 5. In the training algorithm, the number of attempts is 6, the learning rate for delay is $\eta_\tau = 0.1$, the learning rate for other parameters is $\eta = 0.01$, the batch size is $N = 30$, the maximum iteration number is $q_{\max} = 3000$, and the early stop iteration number is $s_{\max} = 500$.

The iterations of the loss and the NDDE parameters are shown in Fig. 6 for the attempt which gives the best training loss. For this attempt, the training process terminates before the maximum iteration because the loss stops decreasing for a while. The minimum loss appears around 1000 iteration and the corresponding parameters are recorded and used in the final trained NDDE. As we can see from panels (b), (c) and (d), the NDDE parameters indeed approach the ground truth in the simulation data. The learned latency is $\tau_{\text{est}} = 1.01$ s and the corresponding weights are $W_1 = [-0.1964, -0.9977]$. The bias $b_1 = 0.0002$ is also learned and it corresponds to the offset $-b_1/w_{11} = -0.0012$ m. These are all within 1% of the true values. This example shows that the proposed NDDE model and algorithm are capable of learning the overall latency and the ToD parameters simultaneously, from very limited trajectory data.

B. Learning from human-operator-in-the-loop experiments

We utilize experimental driving data where a human operator drives in a high-fidelity simulation environment. In particular, we establish a remote driving simulator using TELECARLA [26], in which we can introduce an additional delay as the communication latency. As shown in Fig. 7, the experimental setup consists of three parts: the client computer, the Ethernet wire connection, and the server computer. The CARLA simulations are running on the server computer and the data is transmitted to the client computer via the Ethernet wire with a network emulation Web-GUI. The network emulation introduces a constant delay $\Delta\tau$ to represent the communication latency. The client computer shows a driver's view to the remote operator and the steering commands generated by the operator are recorded and sent back to the server for execution.

In the experiments, the human operator performs a lane change maneuver after the vehicle reaches an equilibrium speed. The speed is maintained at a constant value around 4 m/s by a controller during the lane-changing maneuver and

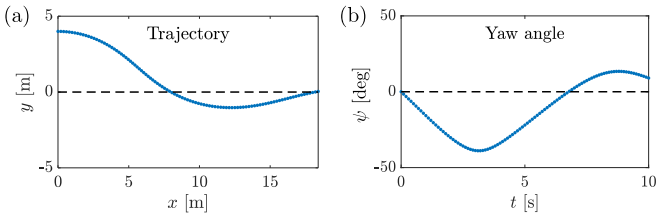

Fig. 5. Training data obtained from simulating the kinematic model.

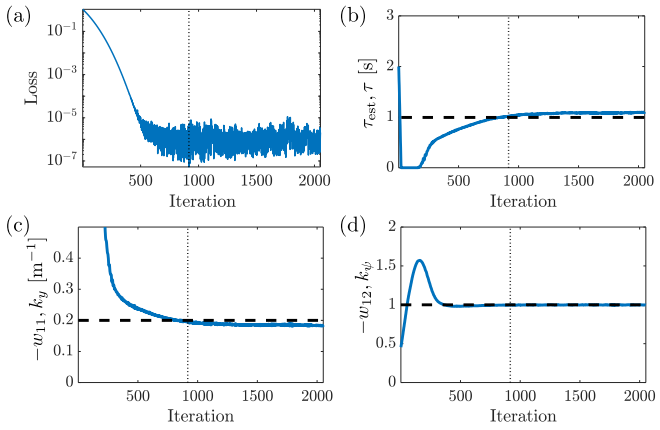

Fig. 6. Evolution of the loss and the equivalent parameters along the training iterations when learning from simulation data.

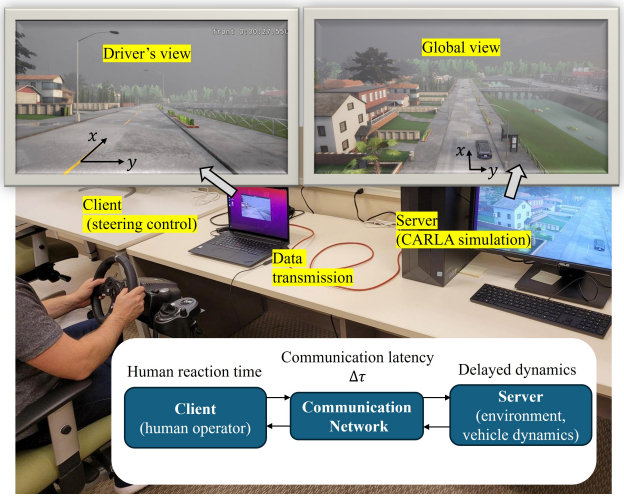

Fig. 7. TELECARLA setup used for collecting experimental data.

the human operator is only controlling the steering wheel. Multiple experiments are conducted under different values of the communication latency $\Delta\tau = 0, 0.15, 0.3, 0.4$ s. Each set of experiments contains three runs. The trajectory data are presented in Fig. 8 under different introduced latency $\Delta\tau$. It can be observed that when $\Delta\tau \geq 0.4$ s, the human operator starts losing control of the vehicle.

When training the neural networks, we do not combine the data from multiple runs since human behavior may differ even under the same driving conditions. Fig. 9(a) and (c) depict runs under $\Delta\tau = 0$ and $\Delta\tau = 0.4$ s, respectively. The NDDE (11) is utilized to learn from individual trajectories, each containing 24-seconds of data with sampling time 0.05

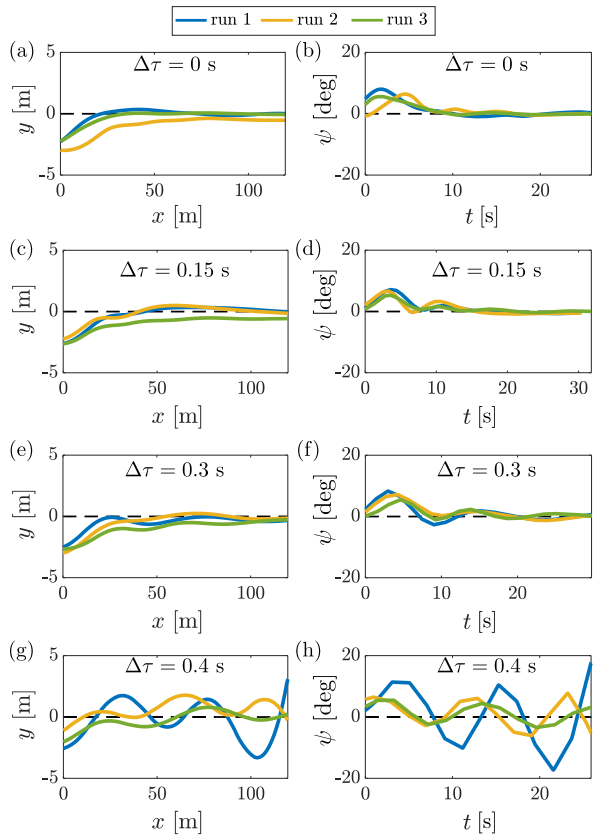

Fig. 8. Experimental data of a lane-change maneuver with a human remote operator in the loop. Different network latencies $\Delta\tau$ were introduced in the experiments and three runs of data were collected under each condition.

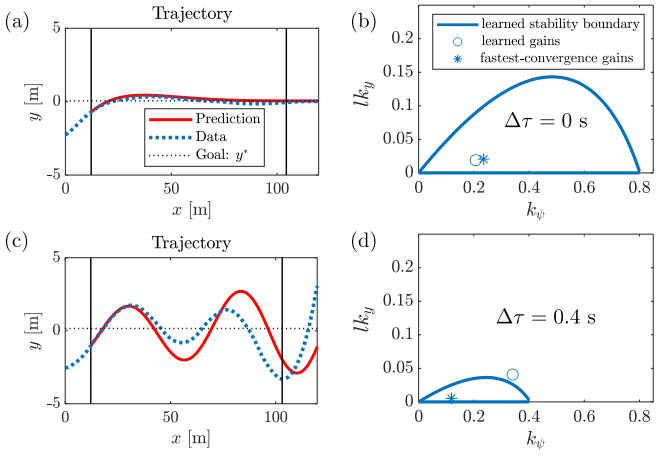

Fig. 9. Simulations of the trained NDDE are compared to the experimental data on the left. Stability charts for the learned parameters of the human operator are shown on the right. (a)-(b) a run of data collected under $\Delta\tau = 0$ s. (c)-(d) a run of data collected under $\Delta\tau = 0.4$ s

second (480 data points), see the segment until second vertical black line. The maximum allowed latency is set to be 3 seconds, so the training data until the first vertical black line is not included in the loss (12).

After training, the NDDE is simulated to predict the trajectory from the given history, beyond the training data. This allows us to examine how this simplest nontrivial model

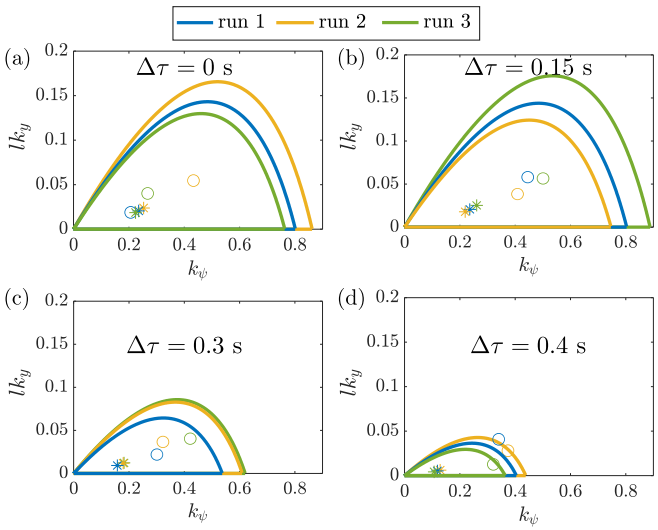

Fig. 10. Stability charts extracted from the TELECARLA experiments.

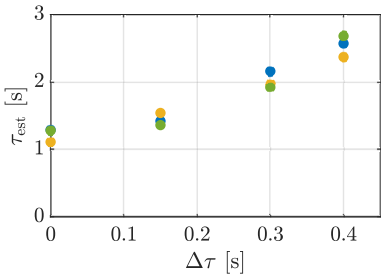

Fig. 11. The overall latency in the loop estimated from individual trajectories for different values of the communication latency $\Delta\tau$.

captures the complex dynamics of the human-operator-in-the-loop TELECARLA experiments, under different latencies. Since the simulation error is not considered in the loss function during training, it is necessary to examine the simulations even on the training data. When the latency is small, as in Fig. 9(a), the NDDE aligns well with the data. When the latency is larger and the trajectory is unstable, as shown in Fig. 9(b), the NDDE still captures the unstable behavior qualitatively, while the prediction accuracy decreases.

Stability and performance analysis can be done for the trained NDDE in the same way as for classical delay differential equations, since the data-driven model is in the form (1)-(2). Fig. 9(b) and (d) show the stability boundaries and the fastest-convergence gains (blue stars) under the learned scaled latency $\hat{\tau} = \tau_{\text{est}} w_2 / l$ where τ_{est} is the learned overall latency and w_2 is the learned speed. The equivalent control parameters k_y, k_ψ can also be extracted from the learned weight W_1, which are indicated as blue circles. When the human gains are close to the fastest-convergence gains, the trajectory of the teleoperated vehicle converges to the straight line quickly without oscillations. When the learned human gains are located outside the stability boundary, the ToD trajectory is indeed oscillatory and unstable.

We plot the stability charts and the learned human gains of all runs in Fig. 10. When the introduced latency is 0, the

overall latency in the loop is small. Then the stable region is large and the human can perform close to the fastest-convergence gains. However, when the introduced latency is large, the stable area is small, and the human gains are located outside or near the stability boundaries. We remark that similar behavior is observed in balancing problems where human reaction delay also plays an important role [27].

We show the correlation between the learned latency and the communication latency for all experiments in Fig. 11. The learned delay τ_{est} increases with the introduced communication latency $\Delta\tau$ and it varies under same $\Delta\tau$. This indicates that the human reaction time may differ in individual experiments, under the same network conditions.

V. Conclusion

In this paper, we built a bridge between physics-based and data-driven modeling of teleoperated driving (ToD) with time delays in the control loop by using neural delay differential equations (NDDEs). We designed a NDDE based on a kinematic vehicle model, so that the embedded neural networks can be trained with limited data and the parameters are interpretable. The learning algorithms developed for NDDEs enabled us to estimate the overall latency and to characterize the remote operator at the same time. Stability and performance analysis was carried out on the trained NDDE. We demonstrated the benefits of learning the ToD dynamics with a simple NDDE using simulation data as well as experimental data collected when a human operator was driving in a high-fidelity simulation environment.

As future directions, we will implement the training algorithm online and provide the performance evaluation in real-time. Different NDDE models will be designed and evaluated based on physics-based models incorporating steering and tire dynamics in a large variety of maneuvers. We also plan to study the behaviors of different remote operators under varying communication latency. Adding time-varying characteristics, such as dynamic weather and road surface conditions, is also an interesting future direction.

Acknowledgment

We genuinely thank Mariam Nour for setting up the TELECARLA simulation platform.

References

[1] I. Vörös, D. Takács, and G. Orosz, "Safe lane-keeping with feedback delay: bifurcation analysis and experiments," *IEEE Control Systems Letters*, vol. 7, pp. 1147–1152, 2023.

[2] S. Giacintucci, F. Della Rossa, G. Mastinu, G. Previati, and M. Gobbi, "Time delay effects on vehicle-and-driver stability," *IFAC-PapersOnLine*, vol. 58, no. 27, pp. 73–77, 2024.

[3] D. Yanakiev, J. Eyre, and I. Kanellakopoulos, "Analysis, design, and evaluation of AVCS for heavy-duty vehicles with actuator delays," UC Berkeley, California PATH, Tech. Rep., 1998. [Online]. Available: https://escholarship.org/uc/item/931877r2

[4] X. A. Ji, S. S. Avedisov, M. I. Khan, M. C. Lucas-Estan, B. Coll-Perales, I. Vörös, O. Altintas, and G. Orosz, "Fundamental rules of teleoperated driving with network latency on curvy roads," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2024, pp. 1841–1846.

[5] R. Donà, K. Mattas, G. Albano, S. Váss, and B. Ciuffo, "Stability issues in adaptive cruise control systems and traffic implication," in *Advanced Vehicle Control Symposium*. Springer, 2024, pp. 961–967.

[6] R. Donà, K. Mattas, G. Albano, S. Vass, and B. Ciuffo, "Towards automated driving: findings and comparison with ADAS," in *Advanced Vehicle Control Symposium*. Springer, 2024, pp. 954–960.

[7] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, pp. 1–54, 2017.

[8] J. Fayyad, M. A. Jaradat, D. Gruyer, and H. Najjaran, "Deep learning sensor fusion for autonomous vehicle perception and localization: A review," *Sensors*, vol. 20, no. 15, p. 4220, 2020.

[9] 5GAA Automotive Association and others, "Tele-operated driving (ToD): Use cases and technical requirements," *5GAA Automotive Association, Tech. Rep*, 2021.

[10] M. C. Lucas-Estañ, B. Coll-Perales, I. K. Khan, S. S. Avedisov, O. Altintas, J. Gozalvez, and M. Sepulcre, "Support of teleoperated driving with 5G networks," in *98th IEEE Vehicular Technology Conference (VTC2023-Fall)*. IEEE, 2023, pp. 1–6.

[11] J. I. Ge and G. Orosz, "Connected cruise control among human-driven vehicles: Experiment-based parameter estimation and optimal control design," *Transportation Research Part C*, vol. 95, pp. 445–459, 2018.

[12] X. A. Ji, T. G. Molnár, A. A. Gorodetsky, and G. Orosz, "Bayesian inference for time delay systems with application to connected automated vehicles," in *IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 3259–3264.

[13] E. Bozzo, D. Breda, and M. Tanveer, "Sparse identification of time delay systems via pseudospectral collocation," *IFAC-PapersOnLine*, vol. 58, no. 27, pp. 108–113, 2024.

[14] A. Köpeczi-Bócz, Tamás, H. Sykora, and D. Takács, "Data-driven delay identification with sindy," in *International Conference on Nonlinear Dynamics and Applications*. Springer, 2023, pp. 481–491.

[15] X. A. Ji, K. van den Boom, N. van de Wouw, and G. Orosz, "Act-and-wait strategy for mitigating the effect of latency in remote driving," *IFAC-PapersOnLine*, vol. 58, no. 27, pp. 61–66, 2024.

[16] T. G. Molnar, A. K. Kiss, A. D. Ames, and G. Orosz, "Safety-critical control with input delay in dynamic environment," *IEEE transactions on control systems technology*, vol. 31, no. 4, pp. 1507–1520, 2022.

[17] B. Chen, M. Xu, L. Li, and D. Zhao, "Delay-aware model-based reinforcement learning for continuous control," *Neurocomputing*, vol. 450, pp. 119–128, 2021.

[18] F. Naseer, M. N. Khan, A. Rasool, and N. Ayub, "A novel approach to compensate delay in communication by predicting teleoperator behaviour using deep learning and reinforcement learning to control telepresence robot," *Electronics Letters*, vol. 59, no. 9, p. e12806, 2023.

[19] Q. Zhu, Y. Guo, and W. Lin, "Neural delay differential equations," in *International Conference on Learning Representations*, 2021.

[20] X. A. Ji and G. Orosz, "Learn from one and predict all: single trajectory learning for time delay systems," *Nonlinear Dynamics*, vol. 112, no. 5, pp. 3505–3518, 2024.

[21] E. VanDerHorn and S. Mahadevan, "Digital twin: Generalization, characterization and implementation," *Decision support systems*, vol. 145, p. 113524, 2021.

[22] Z. Hu, S. Lou, Y. Xing, X. Wang, D. Cao, and C. Lv, "Review and perspectives on driver digital twin and its enabling technologies for intelligent vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 417–440, 2022.

[23] S. Beregi, S. S. Avedisov, C. R. He, D. Takács, and G. Orosz, "Connectivity-based delay-tolerant control of automated vehicles: theory and experiments," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 275–289, 2023.

[24] W. B. Qin, Y. Zhang, D. Takács, G. Stépán, and G. Orosz, "Nonholonomic dynamics and control of road vehicles: moving toward automation," *Nonlinear Dynamics*, vol. 110, no. 3, pp. 1959–2004, 2022.

[25] X. A. Ji and G. Orosz, "Trainable delays in time delay neural networks for learning delayed dynamics," *IEEE Transactions on Neural Networks and Learning Systems*, 2024. [Online]. Available: https://doi.org/10.1109/TNNLS.2024.3379020

[26] M. Hofbauer, C. B. Kuhn, G. Petrovic, and E. Steinbach, "Telecarla: An open source extension of the carla simulator for teleoperated driving research using off-the-shelf components," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 335–340.

[27] G. Buza, J. Milton, L. Bencsik, and T. Insperger, "Establishing metrics and control laws for the learning process: ball and beam balancing," *Biological Cybernetics*, vol. 114, pp. 83–93, 2020.