2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Neural Network Modeling for Steering Control of an Autonomous Vehicle

Gowtham Garimella[1], Joseph Funke[2], Chuang Wang[2], and Marin Kobilarov[3]

Abstract— **Model-based control of dynamical systems typically requires accurate domain-specific knowledge and specifications of possibly proprietary system components. In the context of autonomous driving, steering actuator dynamics can be difficult to model due to an integrated proprietary power steering control module. While first-principles models derived from physics laws can often approximate the system behavior, it remains generally difficult to capture non-physically derived behavior based on proprietary software algorithms in the power steering system. To overcome this limitation, this work instead employs a recurring neural network to model the steering dynamics of an autonomous vehicle. The resulting model is then integrated into a Nonlinear Model Predictive Control scheme to generate feedforward steering commands for embedded control. The proposed approach is compared to traditional first-principles steering modeling through on-vehicle experiments and statistical data validation. As a result, it is shown that the neural network model can be automatically generated with less domain-specific knowledge, can predict steering dynamics more accurately, and perform comparably to a high-fidelity first principles model when used for controlling the steering system of a self-driving vehicle.**

I. Introduction

This paper considers the dynamical modeling of steering systems in passenger vehicles and the use of derived models for steering control, as a basic building block for autonomous driving. We propose an architecture consisting of a nominal physics-based model that is embedded as a block inside a data-driven recurrent neural network (RNN). This RNN model serves as a transition function for model-predictive-control (MPC) to achieve desired steering behavior. The approach is suitable for systems with known electromechanical specifications but also for black-box (e.g. third-party OEM) vehicle systems with unknown characteristics. We show that this method achieves marked improvement over traditional feed-forward techniques and study the effects of different strategies for combining RNNs with a simple physics-based model.

The steering system can nominally be modeled based on known physics laws from first principles. The steering dynamics are usually modeled as a second-order forced dynamical system [24] with torque inputs induced by the steering actuator and tire forces [15]. For modern steering systems, a power steering motor is used as the steering actuator and is governed by a steering controller. In this work, we assume that no exact knowledge of the internal controller logic is available and the first-principles approach is thus a very coarse approximation to the actual behavior.

Unlike the first principles models, non-parametric models do not depend on strict physics-based assumptions and instead can be derived solely from experimental input-output data. Such models can better capture complex actuator nonlinearities and delays and can thus provide higher predictive accuracy with limited domain specific knowledge [14].Recently, RNN models have been employed in Model Predictive Control (MPC) framework to produce a controller for robot manipulation [11], using deep layers [18] to model the robot dynamics and controls cutting task accurately.

A non-parametric approach can be used in conjunction with control policy learning [20]. Deep neural networks have been proposed to learn a non-parametric control policy based on experience. In a reinforcement learning setting, the control policy tries to maximize a reward function by balancing exploration and exploitation of the dynamics of the system [16]. Guided Policy Search Methods have been proposed to improve the convergence of the reinforcement learning methods [25]. While these techniques are very general and could work directly with the physical system, our present work focuses on first learning an accurate and robust dynamics model only, which can then be rigorously validated and used for traditional model-based control.

Steering control has been achieved using simple first principles model with robustness to disturbances [6], [9]. Similarly, lateral control schemes that control the lateral displacement of the vehicle using first principles models have been developed [8], [19], [17]. The success of these methods depends on an accurate steering model, which is non-trivial since the power steering dynamics is unknown.

RNN-based models have been employed for vehicle controls in many simulation studies [7], [22], [10], [13]. Rivals et.al [23] experimentally demonstrated neural network based lateral control of a four-wheel-drive car. Apart from the model-based control schemes, vision based neural network control policies have also been developed [21], [5]. However, these control policies rely heavily on the trained vision data and when presented with new examples, can perform unexpectedly. While the approach of mapping from sensory inputs to actions directly holds promise, for safety considerations we consider a more traditional multi-layered model-based approach consisting of high-level planner providing reference

*This work was supported by and conducted at Zoox Inc.

[1]Gowtham Garimella is with the Department of Mechanical Engineering, Johns Hopkins University, 3400 N Charles Str, Baltimore MD 21218, USA ggarime1@jhu.edu

[2]Joseph Funke and Chuang Wang are with Zoox Inc., Menlo Park, California, USA joe.funke|mrwang@zoox.com

[3]Marin Kobilarov is with Zoox Inc., Menlo Park, California, USA and with the Johns Hopkins University, 3400 N Charles Str, Baltimore MD 21218, USA marin@zoox.com

trajectories that are tracked by a low-level controller. This decomposition allows for individual verification of each layer on a wide range of datasets.

Contributions.: In this work, a model based control strategy is developed for the steering dynamics of an automotive vehicle. An RNN model has been used to model the steering dynamics of the vehicle. The model is augmented with known physics-based transition blocks to improve its predictive capacity. The learned model is then employed in an MPC framework to produce the feed-forward control torques for a low-level embedded steering PID controller. Experiments on Toyota Highlander vehicles have been conducted to compare the performance of the steering tracking behavior using feedforward controls from different models. The limitations and advantages of several configurations of the proposed architecture are examined.

Organization.: The rest of the paper is organized as follows. In section II, the steering control architecture is laid out, and the necessity for feedforward steering torque is explained. Next, the steering models required to compute the feedforward steering torque are explained in section III. The procedure to compute feedforward steering torque using the models is specified in section IV. Finally, the effect of adding the feedforward steering torque using different models is compared in section V.

II. Control Architecture

Controlling the steering system requires tracking a reference steering angle trajectory by applying torque inputs to the steering system. The steering reference trajectory is generated from a high-level planner as shown in Fig 1. The usage of a layered structure separates the vehicle behavior from low-level vehicle tracking. Following the control structure, the high-level planner produces a trajectory that is consistent with obstacles and road rules. The MPC trajectory tracking then produces a steering reference trajectory necessary to achieve the high-level planner trajectory. Finally, the low-level steering PID controller computes the actuator steering torque input required to track the steering trajectory.

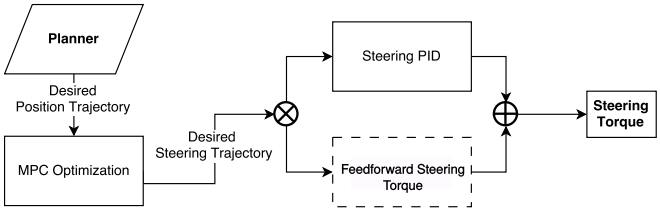

Fig. 1: Control architecture showing how a desired trajectory is converted to steering actuator commands. The focus of the paper is to develop a feedforward steering torque to improve steering controller.

The focus of this work is to develop an appropriate feedforward steering torque input to improve the steering PID controller. The steering PID controller tracks a reference steering trajectory (δ_r, $\dot{\delta}_r$), by commanding the steering torque τ_{cmd} as

$$\tau_{cmd} = -k_p(\delta - \delta_r) - k_d(\dot{\delta} - \dot{\delta}_r) - k_i \int (\delta - \delta_r)dt + \tau_{ff}.$$

Using an accurate feedforward steering torque τ_{ff} is critical as can be observed from Fig 2. The figure shows the steering torques necessary for tracking a reference trajectory using a PID controller with the feedforward steering torque turned off ($\tau_{ff} = 0$). The bulk of the control input during the tracking is provided by the integrator alone which leads to poor tracking performance. The PID gains have been chosen based on gain scheduling to track reference trajectories closely without oscillations. The performance of PID controller is limited by the time delays inherent in the system. The PID gains used ensure stability but allows error to accumulate over time, which is then compensated through the integrator. The goal of introducing the feedforward steering torque is to improve the system response and reduce tracking error without increasing PID gains. The parametric and non-parametric models necessary for generating the feedforward steering torque are investigated next.

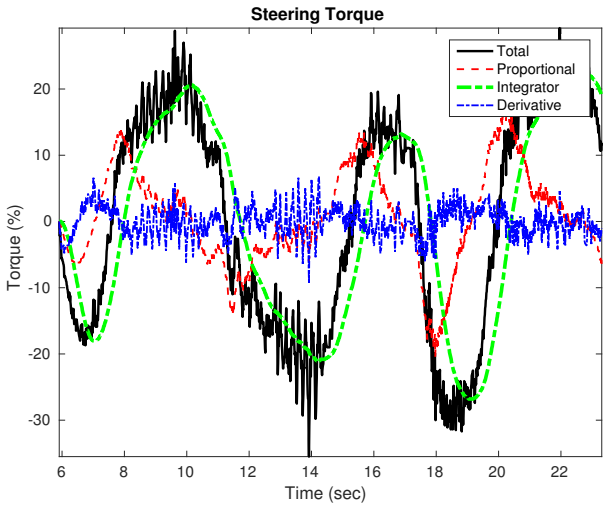

Fig. 2: Steering torque generated by a PID controller during a trajectory tracking experiment. The majority of the torque output is provided by the integrator indicating that the steering system is highly nonlinear.

III. Modeling Steering Dynamics

The steering model predicts the steering angle δ and the steering rate $\dot{\delta}$ based on the applied steering torque τ_s. Steering dynamics are highly coupled with the lateral dynamics of the vehicle, which in turn depends on the longitudinal velocity v_x, which we regard as a recorded input. The lateral states, lateral velocity v_y and yaw rate $\dot{\phi}$, are therefore included as a part of our steering dynamics formulation. The complete state of the steering dynamics is $x = [\delta, \dot{\delta}, \dot{\phi}, v_y]^T$ and controls by $u = [\tau_s, v_x]^T$. The stacked states and controls are represented by $z = [x^T, u^T]^T$.

The discrete steering dynamics evolve according to the discrete-time model $x_{i+1} = f(z_i)$ where i defines the time index along the trajectory. The discrete steering dynamics

predicts the next state x_{i+1} given the current state x_i and control u_i. The state transition function f is an unknown nonlinear function of the previous state and controls. It can be derived from a first principles approach or using a non-parametric model such as a neural network.

A. First principles model

The first principles model is a second order model that integrates the net steering torque to obtain the steering angle and its rate. The net steering torque is given by a combination of the steering system dynamics, road wheel interactions, and power steering torque. The steering system dynamics consists of Coulomb friction from the steering rack, jacking torque due to camber angle, and damping caused by rigid body dynamics [12]. The self-aligning torque is produced due to tire deformation when the steering wheel moves against the tire thread. At small slip angles, the self-aligning torque is proportional to the lateral force F_{yf} applied to the front tire. The resulting first principles model is

$$\ddot{\delta} = \underbrace{k_p\delta}_{\text{jacking}} + \underbrace{k_d\dot{\delta}}_{\text{damping}} + \underbrace{k_aF_{yf}}_{\text{Aligning}} + \underbrace{k_c\text{sgn}(\dot{\delta})}_{\text{Coulomb}} + \underbrace{g(\tau_s, v_x)\tau_s}_{\text{Power Steering}},$$

$$\text{sgn}(\dot{\delta}) = \begin{cases} 1 & \text{if } \dot{\delta} > 0 \\ -1 & \text{otherwise} \end{cases}.$$

A tire model specifies the lateral force applied to the front tire [15]. The lateral velocity and the yaw rate of the car are propagated using the bicycle model [24].

The power steering system applies an actuation torque based on the torque sensor input τ_s and longitudinal velocity v_x. The net torque from power steering system is modeled as a nonlinear gain $g(\tau_s, v_x)$ on the torque sensor input.

$$\begin{aligned} g(\tau_s, v_x) &= g_1(\tau_s)g_2(v), \\ g_1(\tau_s) &= \gamma_1\left[1 - e^{-\left(\frac{\tau_s-\gamma_2}{\gamma_3}\right)^2}\right], \\ g_2(v) &= \left[\gamma_4 + (1-\gamma_4)e^{-v_x/\gamma_5}\right]. \end{aligned}$$

The power steering gain $g(\tau_s, v_x)$ is decomposed into two multiplicative gains. The first gain is dependent on driver input. It increases with the driver input and saturates to a constant value γ_1 as driver input becomes large. The parameters γ_2, γ_3 specify the bias and the rate of change of the gain respectively. The second gain decreases with vehicle velocity, where the rate of decrease is specified by γ_5 and the saturated value by γ_4. The form of the gain function has been chosen based on observed experimental data between driver input torque and assist torque. The power steering dynamics presented here is an approximation based on the graphs shown in Badawy et.al [2], since the true dynamics is unknown. The unknown parameters in the first principles model are obtained using standard least-squares regression based on the error between predicted and measured steering angle and steering rate.

B. Neural Network Model

The neural network model approximates the nonlinear function $x_{i+1} = f(z_i)$ to predict the discrete-time steering dynamics. The model consists of several units of neural network blocks stacked in time. Each neural network block is divided into a known physics function and a stack of fully connected layers. Figure 3 shows a block diagram of a neural network block. The current inputs and previous outputs are stacked together and fed into the fully connected layers and the physics function. The output of these layers is combined to predict the state at the next step. These predicted variables together with new control inputs are fed back into the network to continue prediction for the next step.

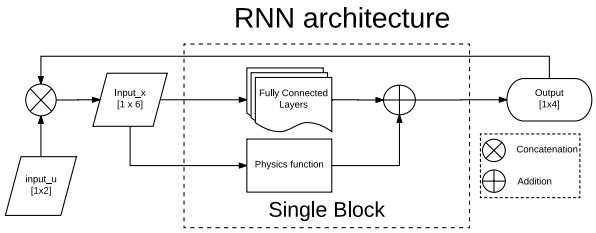

Fig. 3: The neural network architecture used for learning lateral dynamics.

The overall discrete dynamics for a neural network model can be written as $f(z) = f_{ph}(z) + f_{nn}(z)$, where f_{ph} denotes the physics function and f_{nn} represents the neural network layers. The physics function incorporates basic domain knowledge by predicting the steering angle as an integral of the steering rate and the yaw rate based on the no-slip condition for a kinematic car model. The physics function can be mathematically stated as

$$f_{ph}(z_i) = \begin{bmatrix} \delta_i + k_1\dot{\delta}_i \\ 0 \\ v_{xi}\tan(k_2\delta_i) \\ 0 \end{bmatrix},$$

where the unknown parameters are the time-constant for integration k_1 and the inverse of the wheelbase of the car k_2. Similarly, the neural network layers can be expressed as

$$\begin{aligned} f_{nn}(z_i) &= g_n\left(g_{n-1}\left(g_{n-2}\left(\cdots g_0\left(z_i\right)\right)\right)\right), \\ g_i(x) &:= \sigma(W_ix + b_i), \quad i = 0, \ldots, n-1, \\ g_n(x) &:= W_nx + b_n. \end{aligned}$$

The function $g_i(\cdot)$ defines a single fully connected layer for the neural network with the parameters W_i, b_i. The function $\sigma(\cdot)$ denotes the activation function used in the network. The activation function is chosen to be the hyperbolic tangent function. The last layer $g_n(\cdot)$ is specified as a unit activation function since the neural network is used in a regression problem.

The neural network layers predict the residual dynamics not modeled by the physics function. The neural network layers, therefore, capture the effects of road-tire interactions, power steering logic, and steering system dynamics. The addition of physics function improves the gradient flow of the network thereby providing better prediction performance. The increase in depth of the fully connected layers increases the ability of the RNN to model higher nonlinear models.At

the same time, it is also harder to train deeper neural network models due to the vanishing gradient problem [3]. In this work, the depth of the neural network layers has been experimentally determined to capture the unknown dynamics well.

Training: The neural network weights and the physics function parameters are optimized using time series data collected from driving the vehicle along variable curvature paths with different desired longitudinal velocities. The steering dynamics are controlled using a preliminary Proportional Integral Derivative (PID) controller during the data collection phase.

The collected time series data is divided into fixed time horizon segments to train the neural network model. The loss function during neural network training is set to minimize the difference between propagated states and measured states for each of the fixed time horizon segments. The loss function also adds an L2 regularization with a user-selected gain c_r to avoid overfitting of the parameters. The goal of the training phase is to find the optimal parameters θ^* that minimize the training cost as

$$\theta^* = \arg\min_{\theta} \sum_{j=1}^{m} C(\tilde{x}_{0:N}, u_{0:N-1}, \theta), \tag{1}$$

$$C(\tilde{x}_{0:N}, u_{0:N-1}, \theta) = \sum_{i=1}^{n} (\tilde{x}_i - x_i)^T P (\tilde{x}_i - x_i) + c_r \theta^T \theta, \tag{2}$$

$$\text{s.t} \quad x_{i+1} = f_{ph}(z_i, \theta) + f_{nn}(z_i, \theta), \tag{3}$$

$$z_i = [x_i, u_i], x_0 = \tilde{x}_0, \tag{4}$$

$$\theta = [W_0, b_0, W_1, b_1, \cdots, W_n, b_n, k_1, k_2], \tag{5}$$

where θ denotes the weights of the network along with the unknown physics parameters.

The cost function C measures the deviation of the propagated states x_i from measured states $\tilde{x}_i$ for m sample trajectories. The state deviations are weighed using a diagonal matrix P to enforce a uniform scale across the deviations. The matrix P is usually chosen as the inverse covariance of the sensor measurements.

The propagation of steering dynamics results in the propagated states significantly diverging from the measured states for random initialization of parameters. The optimization is thus performed in two stages: first by limiting the propagation to a single step and initializing the parameters obtained from the previous stage and optimizing over the entire trajectory segment. This two-stage optimization proposed by Lenz et al. [11] has been shown to perform better than random initialization of parameters.

Implementation: The RNN is coded as a computational graph in TensorFlow [1] package. The training data is provided by 20,000 trajectory segments with a 0.5-second horizon which corresponds to approximately 150 hours of driving data. The neural network has been trained with two fully connected layers. The optimization has been performed using mini-batch gradient descent with a batch size of 200 samples.

The optimal parameters are used to verify the performance of the model on a test data set of 5,000 trajectory segments. The RNN model is used to propagate the lateral dynamics using the initial state and controls along the trajectory. Figure 4 shows the RMS error between the propagated and measured handwheel angle along the trajectory using different trained models. The addition of a second layer to the RNN along with physics function improves the performance of prediction significantly. Increasing the RNN layers further does not result in an improvement of performance. Hence the depth of neural network is limited to two layers.

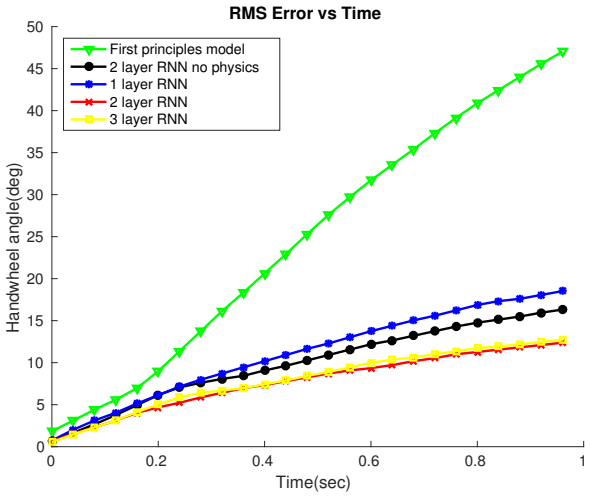

Fig. 4: Averaged RMS Error between propagated and measured hand wheel angle along the trajectory for all the test segments plotted for different dynamics models.

IV. Steering Control

The steering torque used to track a steering reference trajectory consists of a PID component and a feedforward component. The steering reference trajectory comprises of a reference steering angle δ_r and steering rate $\dot{\delta}_r$. The feedforward steering torque estimates the necessary command to track the desired reference and thereby improves tracking performance when compared to applying the PID component alone. The procedure to compute feedforward steering torque using different models is discussed next.

A. Inverting First Principles Model

Here, the feedforward steering torque for controlling the steering system is computed by inverting the first principles model. Denoting the power steering dynamics at a given longitudinal velocity by $P(\tau_s) := g(\tau_s, v_x)\tau_s$, the input torque required to track a steering reference trajectory is computed as

$$\tau_s^* = P^{-1}\left(-k_p\delta_r - k_d\dot{\delta}_r - k_a F_{yf}\right).$$

The desired steering angle, steering rate and forward velocity along the trajectory are assumed to be known during inversion. The tire force F_{yf} is computed based on reference trajectory as

$$F_{yf} = k_f m v_x \dot{\phi},$$

where the gain k_f is the ratio of the distance between the center of mass to front wheels to the wheelbase. The feedforward steering torque for trajectory tracking is chosen to be equal to the computed input torque $\tau_{ff} = \tau_s^*$, since the computed input torque is only dependent on the reference trajectory.

The double derivative of the reference steering angle, the Coulomb friction and the desired steering acceleration are not accounted for when computing the feedforward steering torque to avoid inducing high-frequency oscillations into the steering control.

B. Lookup table

The feedforward steering torque can also be computed using a lookup table under steady state assumptions. The lookup table provides the steering torque required to achieve a steady state steering angle for a given velocity. The data in the lookup table is filled based on experiments in which the steering torque is adjusted to achieve the desired steering angle for a given forward velocity. The feedforward steering torques for a given reference steering angle and forward velocity are then computed by interpolating the data points from the lookup table.

C. NMPC Steering Control

When using the neural network model, the feedforward steering torque is computed by solving an NMPC optimization problem. The optimization problem is set to track a reference steering trajectory $\tilde{s}_{0:N}$ consisting of desired steering angle δ_r and desired steering rate $\dot{\delta}_r$ using the neural network model. The longitudinal velocity is fixed to be equal to the reference velocity from high-level planner during the optimization process. In contrast to the neural network optimization in Eq 5, this optimization fixes the neural network parameters and optimizes over the feedforward steering torques to track the desired steering trajectory. The optimization problem for computing the feedforward steering torque can be written as

$$\tau^*_{0:N-1} = \underset{\tau_{0:N-1}}{\arg\min} \sum_{i=1}^{N} (s_i - \tilde{s}_i)^T P_i (s_i - \tilde{s}_i) + \tau_{i-1}^T R_i \tau_{i-1},$$

$$\text{s.t} \quad x_{i+1} = f_{ph}(z_i) + f_{nn}(z_i),$$

$$s_i = [\delta_i, \dot{\delta}_i]^T, \tilde{s}_i = [\delta_r, \dot{\delta}_r]^T$$

$$\text{Given } \{v_{x0:N-1}, \tilde{x}_{0:N}\}.$$

The first component of the optimal steering torque from NMPC optimization is sent to the PID controller as feedforward steering torque ($\tau_{ff} = \tau_0^*$). If there are delays in sending the torque command, the feedforward steering command can correspond to the future stamped steering torque as in $\tau_{ff} = \tau^*_{delay}$

Optimization Algorithm: The NMPC optimization is performed using a Stage-wise Newton method [4]. The algorithm consists of repeatedly application of two steps: backward pass to compute a quadratic relaxation of the value function and forward pass to update the states and controls that minimize the approximated value function. This procedure is repeated until a specific time limit. If the optimization did not succeed by then, the feedforward term is set to zero. The gradient of the cost function with respect to the input variables can be obtained analytically for the neural network model. The analytical gradients avoid using expensive finite differencing scheme for computing gradients.

V. Results

The MPC controller for steering dynamics has been tested on a passenger car equipped with a high-precision GPS system, and an onboard compute stack. The onboard compute stack consists of two computational modules: high-level computer and low-level microcontroller. The high-level computer performs the high-level planning and provides a steering reference trajectory as an output. It also produces the feedforward steering torque necessary for tracking the steering reference trajectory using NMPC optimization described in section IV. The low-level microcontroller runs a steering PID controller that outputs a torque command based on the input steering reference and feedforward torque.

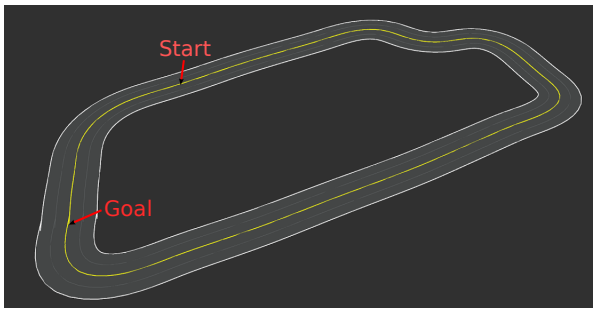

Fig. 5: Test track used for autonomous driving experiments

The goal of the experiments is to closely track the desired waypoint trajectory in a small test track as shown in Fig 5. The steering controller is tested at longitudinal velocities of 5mph and 10mph under a maximum lateral acceleration of 3.5m/ss. Large lateral acceleration and low speeds are used to test the system, as these conditions tended to cause worse steering tracking performance. The effect of adding a feedforward steering torque using different models on the steering tracking performance is shown in Table I. Not including a feedforward in the controller results in the largest RMS error, as expected. Adding the lookup feedforward model improves the performance only slightly, probably because these are more dynamic rather than steady state maneuvers. Computing the feedforward steering torque using the first principles model, which incorporates these dynamic effects, outperforms the lookup table model.

The neural network performs comparably to first principles model with regards to the RMS error. It also performs better than the first principles model at low velocities. The first principles model is not able to model the road-wheel interactions correctly at low velocities. This is likely because at lower velocities, the power steering system provides larger assist, and that effect is difficult to model directly without

knowing the underlying power steering software algorithm. The neural network, however, benefits from its own internal representation based simply on training data.

Controller/Longitudinal velocity	5mph	10mph
No Feedforward	18.63	18.61
Lookup table	16.43	14.88
First principles model	12.48	9.56
Neural network model	10.21	9.53

TABLE I: RMS Handwheel error (degrees) for different models based on multiple trials.

The steering performances of different steering controllers are shown in Fig 6 for a single trial at 5mph and 10mph longitudinal velocities. The initial steering angle differences are matched across different controllers before computing the RMS error to remove the effect of various initial conditions. The PID controller without feedforward steering torque has the highest RMS steering error as seen in Table I, and as a result of the larger errors, the integrator performs much of the tracking task. Adding a lookup table offers marginal improvement, since the maneuvers are highly dynamic. The first principles model and the neural network model outperform the other methods with regards to RMS error. The magnitude of the integrator is also small and most of the tracking performed by the feedforward steering torque.

VI. Conclusions

The use of RNN, a non-parametric model, for modeling and control of a vehicle's steering system has been presented. The creation of the neural network model did not require domain specific knowledge about the power steering module and steering system dynamics. It used only minimal knowledge of the nominal car dynamics to improve the prediction capability. The resulting neural network model outperformed a first principles model in the long-term prediction of steering dynamics and operated equally well in generating a feedforward reference command for control. These results demonstrate that a simple RNN, augmented with simple dynamics information but lacking domain specific knowledge, can be suitable for dynamical modeling and control of a vehicle steering system. The current system is limited to learning RNN model offline and computing feedforward torque online. Future work will include learning the non-parametric model online and testing the vehicle on changing road conditions.

References

[1] TensorFlow. `https://www.tensorflow.org/`, 2017.

[2] Badawy Aly, Zuraski Jeff, Bolourchi Farhad, and Chandy Ashok. Modeling and analysis of an electric power steering system. In *SAE Technical Paper*. Society of Automotive Engineers(SAE) International, 03 1999.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, Mar 1994.

[4] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[5] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[6] Zhengrong Chu and C. Q. Wu. Active disturbance rejection control for automated steering in vehicles and controller tuning. In *2016 American Control Conference (ACC)*, pages 7567–7572, July 2016.

[7] K T R Van Ende, D Schaare, J Kaste, F Küçükay, R Henze, and F K Kallmeyer. Practicability study on the suitability of artificial, neural networks for the approximation of unknown steering torques. *Vehicle System Dynamics*, 54(10):1362–1383, oct 2016.

[8] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, May 2007.

[9] J. Guldner, W. Sienel, Han-Shue Tan, J. Ackermann, S. Patwardhan, and T. Bunte. Robust automatic steering control for look-down reference systems with front and rear sensors. *IEEE Transactions on Control Systems Technology*, 7(1):2–11, Jan 1999.

[10] A Haddoun, M E H Benbouzid, D Diallo, R Abdessemed, J Ghouili, and K Srairi. Modeling, Analysis, and Neural Network Control of an EV Electrical Differential. *IEEE Transactions on Industrial Electronics*, 55(6):2286–2294, jun 2008.

[11] Ian Lenz AND Ross Knepper AND Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, Rome, Italy, 2015.

[12] Krisada Kritayakirana and J Christian Gerdes. *Autonomous vehicle control at the limits of handling*. PhD thesis, Stanford University, 2012.

[13] S Kumarawadu and T T Lee. Neuroadaptive Combined Lateral and Longitudinal Control of Highway Vehicles Using RBF Networks. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):500–512, dec 2006.

[14] I J Leontaritis and Stephen A Billings. Input-output parametric models for non-linear systems Part I: deterministic non-linear systems. *International journal of control*, 41(2):303–328, 1985.

[15] William F Milliken and Douglas L Milliken. *Race car vehicle dynamics*, volume 400. Society of Automotive Engineers Warrendale, 1995.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *Deep Learning Workshop at Neural Information Processing Systems NIPS*, 2013.

[17] L. Ni, A. Gupta, P. Falcone, and L. Johannesson. Vehicle lateral motion control with performance and safety guarantees. *IFAC-PapersOnLine*, 49(11):285 – 290, 2016. 8th IFAC Symposium on Advances in Automotive Control AAC 2016.

[18] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to Construct Deep Recurrent Neural Networks. *International Conference on Learning Representations ICLR*, 2014.

[19] S. Patwardhan, Han-Shue Tan, and J. Guldner. A general framework for automatic steering control: system analysis. In *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*, volume 3, pages 1598–1602 vol.3, Jun 1997.

[20] M M Polycarpou. Stable adaptive neural control scheme for nonlinear systems. *IEEE Transactions on Automatic Control*, 41(3):447–451, mar 1996.

[21] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, DTIC Document, 1989.

[22] G V Puskorius, L A Feldkamp, and L I Davis. Dynamic neural network methods applied to on-vehicle idle speed control. *Proceedings of the IEEE*, 84(10):1407–1420, oct 1996.

[23] Isabelle Rivals, Lon Personnaz, Grard Dreyfus, and Daniel Canas. Real-time control of an autonomous vehicle: a neural network approach to the path following problem. In *5th International Conference on Neural Networks and their Applications*, pages 219–229. Citeseer, 1993.

[24] Jihan Ryu. *State and parameter estimation for vehicle dynamics control using GPS*. PhD thesis, Stanford University, 2004.

[25] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search. *arXiv preprint arXiv:1509.06791*, 2015.

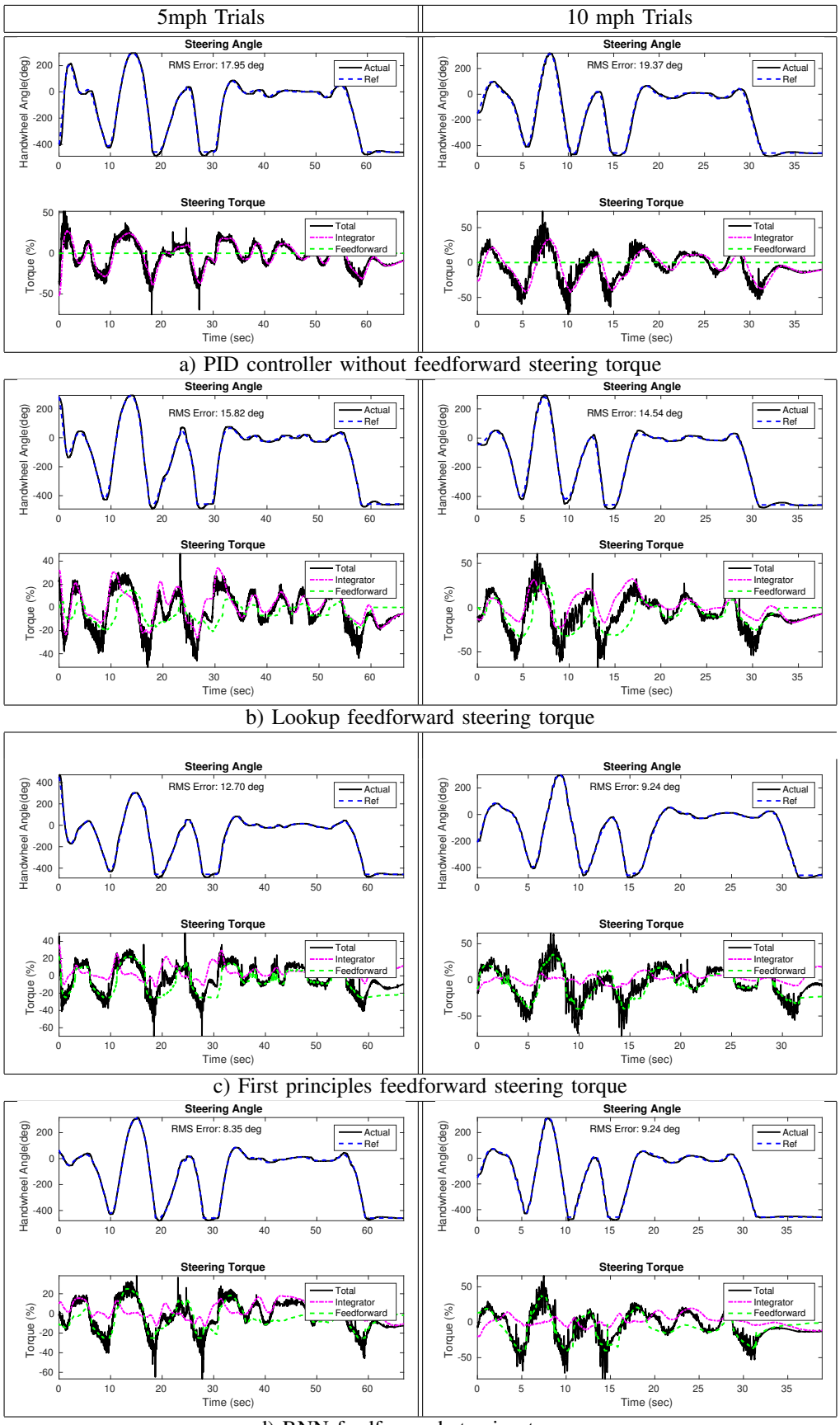

Fig. 6: Comparison of steering performance using steering controllers with different feedforward steering torque methods for a single trial.