

An Automated Learning-Based Procedure for Large-scale Vehicle Dynamics Modeling on Baidu Apollo Platform

Jiaxuan Xu^{†*}, Qi Luo^{†*}, Kecheng Xu*, Xiangquan Xiao*, Siyang Yu*, Jiangtao Hu*, Jinghao Miao* and Jingao Wang*

Abstract—In the autonomous driving industry, vehicle dynamic models are important to control-in-the-loop simulations. For current commercial self-driving simulators, vehicle dynamic models are expressed explicitly by sophisticated analytical equations, which are accurate but difficult to build and expensive to scale to fleets of vehicles of different brands. In this paper, we introduce a highly automated learning-based vehicle dynamic modeling procedure, which has been deployed on Baidu Apollo self-driving platform, to support cross-vehicle data-driven applications on a large scale. Compared with our previous analytical models, the end-to-end learning-based dynamic models can achieve high accuracy with significantly reduced re-development effort.

I. INTRODUCTION

Autonomous driving technology has been popular in industrial and research communities for the past few years. Baidu, one of the leading companies in this field, has been putting great efforts into building an active autonomous driving community since 2013. *Apollo*, Baidu’s self-driving platform, has been open-sourced to external developers since 2017 [1]. After years of development, Baidu has tested hundreds of vehicles for tens of thousands of hours with multiple generations of algorithms.

Currently, in the autonomous driving industry, motion planning and vehicle control algorithms are critical, both to driving safety and riding experiences. However, it could be tedious work for developers to sit in the car and tune the parameters manually. Instead, simulation-based algorithm iterations, which rely on high-fidelity vehicle dynamic models, are widely used. In order to increase the software development efficiency, Baidu builds its own self-driving simulator, *Apollo Dreamland*, to speed up the algorithm iterations in motion planning, vehicle control and other tasks.

In order to conduct control-in-the-loop simulations, the simulator must have accurate vehicle dynamic models, which used to be expressed explicitly by sophisticated analytical equations and parameters. However, as the number of deployed vehicles keep increasing, we soon found that building high-fidelity analytical vehicle dynamic models for each vehicle brand, let alone for each vehicle, is infeasible. Current analytical dynamic modeling method not only involves considerable manual works which cost tremendously amount of time but also brings up great potential for man-made errors. Further, vehicle dynamics usually vary no-

ticeably in the developing phase (i.e. vehicle parts worn out over time, mechanical parts changes, center of mass shift, GNSS/IMU mounting point changes), and it is almost impossible to quickly adjust current equation-based vehicle dynamics. Taken together, it is surprising that most research papers did not attempt to address this cross-vehicle platform puzzle. One could attribute this to the fact that most research-oriented projects only focus on very few vehicles. To solve this industrial-specific problem, in this paper we propose a novel learning-based dynamics modeling procedure and its integration to Apollo simulation system.

Previous researches have investigated dynamic models extensively, and tried to solve it via either traditional system identification methods [2] [3] [4] or machine learning methods [5] [6] [7]. A popular solution is to establish this state-space relation based on Newton formula [10] [11]. However, vehicles’ transmission system, power-train system, and actuator system are all very complex. It is not only difficult to model these systems in explicit expressions but also unacceptable to compute all those expressions as more systems get involved [12] [13]. For those reasons, in this paper, we address this challenging topic from a machine learning perspective. Instead of using analytical expressions and then tuning the complicated parameters manually, we consider the entire dynamic modeling algorithm as an end-to-end system identification task or regression problem.

Closer to our idea, Ali Punjani et al. proposed a two-layer ReLU network model, combined with a quadratic lag model, to identify the dynamics of helicopters based on machine learning methods [8]. However, if the same method would be able to generate high-precision dynamic models for four-wheeled autonomous vehicles, which could be very different from helicopters, wasn’t discussed in their work. Guillaume Devineau et al. trained neural networks to compute vehicle controls in reference [9] based on vehicle dynamics and control-in-the-loop simulation. However, the vehicle dynamic model used in this paper was built on sophisticated analytical equations of chassis and tire dynamics, which still required prior knowledge and manual parameter-tuning. A Gaussian Process Regression model was trained by Theodosios Georgiou et al. to predict the driver’s actions as well as the vehicle states on a certain segmented path of track in racing car games. The non-parametric method was very computationally expensive and the prediction results were limited to a small segment of path [17], which was obviously difficult to use in control-in-the-loop simulation.

[†] Authors contributed equally to this paper

* Authors are with Baidu USA LLC, 250 E Caribbean Drive, Sunnyvale, CA 94089 jiaxuanxu@baidu.com luoqi06@baidu.com

Grady Williams integrated a neural network dynamic model into an information theoretic model predictive control (MPC) in reference[18]. Although the control algorithm was tested on a one-fifth scale rally car, neither the dynamic model's accuracy nor the adaptability to real-sized vehicles were discussed. As a result, there is still a question mark over if the learning-based dynamic models can meet the requirements of control-in-the-loop simulation tasks in the autonomous driving industry.

To solve this problem, in this paper, we contributed by proposing a standardized learning-based vehicle dynamic modeling procedure and tested it with a family of neural network models, as shown in Fig. 1. The dynamic modeling procedure consists of three main parts: *data generation part*, which contains raw data collection, *offline part*, which contains large-scale feature extraction, model training and offline grading, as well as *online part*, which contains control-in-the-loop simulation and automated control algorithm grading. More details on the procedure will be provided in section V. The machine learning models were trained by vehicle driving logs, collected daily by real vehicles driven on road and then stored on Apollo data platform, to represent the vehicles latest dynamic characters. We proved that, with proper data extraction and model training procedures, the learning-based vehicle dynamic model can achieve very high accuracy to meet the industrial requirements of control-in-the-loop simulation tasks. We also tested our method's scalability to different vehicle types (including sedan and van) as well as its versatility under different driving maneuvers (including Going Straight with Sudden Braking, U-turn and Reverse Parking).

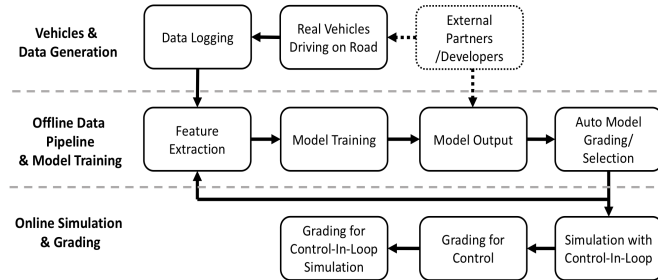


Fig. 1: Apollo Control-In-Loop Architect

Neural networks are known for their non-linear characters and being extremely expressive. By introducing this automated learning-based dynamic modeling procedure, we can achieve high modeling accuracy at much lower re-development effort for fleets of vehicles. On Apollo data platform, both internal users and external partners/developers can contribute either raw vehicle data for model training or ready-available dynamic models of different vehicles for control-in-the-loop simulations. We have implemented and tested the dynamic models to cover different Apollo vehicle types, and will publish more vehicle models in the future for control-in-the-loop simulations under Apollo Dreamland simulator (<http://azure.apollo.auto/>), which greatly improves

the development efficiency for motion planning and vehicle control algorithms.

In this paper, section II reviews a previous rule-based dynamic model and section III introduces our new learning-based dynamic models. Section IV illustrates the overall data extraction and model training procedure, and gives the experimental results. A conclusion is made in section V.

II. RULE-BASED MODELING METHODS

The function of dynamic model is to predict next step's vehicle states $s_{t+1} = f(s_t, a_t)$ (speed, acceleration, angular velocity) given the last step's states s_t and action inputs a_t (throttle, brake and steering wheel)

The real vehicle dynamic models can be very complicated if we take the vehicle's longitudinal-and-lateral correlation factors into consideration, which is nonlinear and difficult to be described in analytical expressions. However, there is a feasible walk-around by decoupling and linearizing the models into longitudinal and lateral models, referred as rule-based models on Apollo platform, with details below.

A. Longitudinal Model (Linear Interpolation)

The main idea behind the vehicles' longitudinal model is to predict acceleration \dot{v}_{t+1} on time step $t+1$ based on last step's speed v_t , throttle command and brake command.

$$\dot{v}_{t+1} = f^{lon}(v_t, a_t^{throttle}, a_t^{brake}) \quad (1)$$

Fig. 2 left is the 3D plot of a real vehicle's acceleration \dot{v}_{t+1} regards to the vehicle's current speed v_t (m/s) and the throttle percentage command a_t^x (positive means throttle is being applied, negative means brake is being applied). Obviously, f^{lon} should be a nonlinear mapping. However, we can record those mappings in a calibration table and use linear interpolation function to approximate \dot{v}_{t+1} in real time.

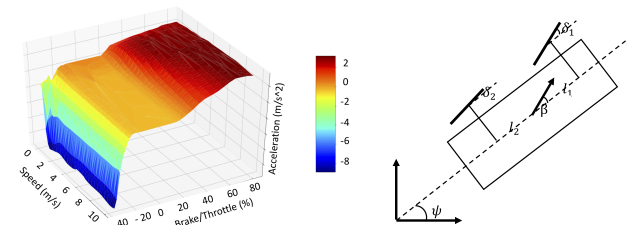


Fig. 2: Linear Analytic Model, Longitudinal Calibration Table (left) and Lateral Bicycle Model (right)

Based on the predicted \dot{v}_{t+1} , the vehicle's velocity and position can be calculated by \dot{v}_{t+1} 's first and second integral over time, as shown in equations (2) to (4). ψ represents for the vehicle's current heading angle.

$$v_{t+1} = v_t + \dot{v}_{t+1} \times \Delta t \quad (2)$$

$$x_{t+1} = x_t + v_{t+1} \times \cos \psi \times \Delta t \quad (3)$$

$$y_{t+1} = y_t + v_{t+1} \times \sin \psi \times \Delta t \quad (4)$$

B. Lateral Model (Bicycle Model)

Fig. 2 also shows a classic bicycle model on the right. By using the bicycle model, we assume that the vehicle is a perfectly symmetric rigid body that the rotating speed of left wheels are equal to right wheels. ψ is the current heading angle of the vehicle in the global coordinate. β is the angle between the angle of vehicle's current velocity v and the angle of the vehicle body. δ_1 and δ_2 are the angles between the vehicle body and front / rear wheels. l_1 and l_2 are the distance between the center of mass of the vehicle and the front / rear wheel axles. The classic bicycle model calculates the vehicle's angular velocity ω in the equation (5).

$$\omega = \frac{v \times \cos \beta}{(l_1 + l_2)} (\tanh \delta_1 - \tanh \delta_2) \quad (5)$$

Usually, people assume that front wheels' angle δ_1 is proportional to the angle of the steering wheel a_t^{steer} . What's more, the angle of rear wheels and current velocity are regarded the same as the angle of vehicle body for the purpose of simplicity, thus δ_2 and β can be further simplified to 0. If δ_1 is small, $\tanh \delta_1$ is close to δ_1 . In this way, the lateral dynamic model can also be written as equation (6) and (7).

$$\omega_{t+1} = k \times v_t \times a_t^{steer} \quad (6)$$

$$\psi_{t+1} = \psi_t + \omega_{t+1} \times \Delta t \quad (7)$$

III. NEURAL NETWORK MODELS

Since there is always a complexity-versus-accuracy trade-off for analytic models, we propose a learning-based dynamic model built on neural networks. Instead of writing the functions analytically, we will turn the problem into a parameter estimation problem (or a system identification problem) by using machine learning techniques.

The input features of the neural networks include last one or several steps' control commands $\{a_{t-1}, a_{t-2}, \dots\}$ as well as vehicle states (acceleration, angular, velocity, speed, heading and position) $\{s_{t-1}, s_{t-2}, \dots\}$. The output features of the neural networks are the predictions of the current vehicle states. The dynamic transform samples $\mathcal{M} : \{\dots, s_{t-1}, a_{t-1}, s_t, a_t, s_{t+1}, a_{t+1}, \dots\}$, which will be used as the training data to fit our machine learning models, can be collected by Baidu Apollo autonomous vehicles at a frequency of 100HZ.

In order to take the longitudinal and lateral models' inner-correlation and model non-linearity into consideration, a simple feed-forward neural network structure with nonlinear activation functions (such as sigmoid or ReLU) will be introduced in this paper. On the other hand, since real-world vehicles always have complicated power transmission systems, a phase delay is expected to exist between the action inputs and state changes. For this reason, dynamic models that are built on recurrent neural networks (such as LSTM), which can represent long sequences of data, will also be proposed in this section.

A. Feed-forward Neural Nets

A feed-forward neural network is the simplest structure of artificial neural nets, where the nodes and links don't form a cycle. One of the feed-forward neural networks, multi-layer perceptron (MLP), consists of at least three layers: an input layer, a hidden layer and an output layer. The MLP training process can be regarded as adjusting the network parameters through back-propagation and gradient descent to minimize the target losses.

In order to represent the vehicle's holistic dynamics, the input layer contains the vehicle's longitudinal and lateral states s_{t-k} and actions a_{t-k} k steps before. The output layer predicts the current incremental status values (acceleration and angular velocity). When the duration value k equals to 1, we use the last frame to predict the current frame, which occurs 10 ms afterward the input. However, since the vehicle dynamic system is usually a complicated inertial system, a phase delay larger than 10 ms between the control commands and the vehicle state reactions is expected in the vehicle dynamics. We can set k to values larger than 1 to take the larger phase delays into consideration. In this way, we use the state-action-pair $\{s_{t-k}, a_{t-k}\}$ that occurs k frames before to predict the current states s_t at time t . The training samples $\zeta : \{s_{t-k}, a_{t-k}, s_t\}$ can be extracted from dynamic transform samples \mathcal{M} .

$$\varphi^* = \arg \min_{\varphi} \sum_{\zeta \in \mathcal{M}} \|f_{\varphi}(s_{t-k}, a_{t-k}) - s_t\|^2 \quad (8)$$

The parameters of the MLP model are marked as φ . The loss function for the neural network can be set as the mean squared error (MSE) between the predicted incremental status $f_{\varphi}(s_{t-k}, a_{t-k})$ and the ground-truth incremental status s_t given by the localization module. Optimizers such as Stochastic Gradient Descent (SGD) or Adaptive Moment Estimation (Adam) can be used to minimize the loss over iterations.

The reason we use feed-forward neural networks, instead of a simple linear regression, to represent the vehicles' dynamic model is for neural networks' strong ability to accurately express a complicated non-linear dynamic system. We will compare the prediction accuracy of MLP models at different duration values k with a simple linear regression model in section IV.

It's worth pointing out that, instead of taking all status as the output, here we just focus on incremental parameters including acceleration \dot{v}_t and angular velocity ω_t . Other longitudinal and lateral status such as linear velocity and heading angle can be calculated by the integration of \dot{v}_t and ω_t over time. We found focusing on incremental values can improve the stability and accuracy of our dynamic models.

B. Recurrent Neural Nets

Recurrent neural network (RNN) is an artificial network structure whose nodes and connections form a directed graph along the temporal sequences. Different from feed-forward networks, RNN can take sequences of data as inputs and

has its internal memory. Not limited to taking just one past frame's state and action $\{s_{t-k}, a_{t-k}\}$ as input, RNN can take all past k frames' actions plus the initial frame's state $\zeta = \{s_{t-k}, a_{t-k}, a_{t-k+1}, \dots, a_{t-1}\}$ as input to predict the current state s_t at time t . For this reason, RNN, by its nature, can be trained to express a system with phase delay between its inputs and outputs.

$$\varphi^* = \arg \min_{\varphi} \sum_{\zeta \in \mathcal{M}} \|f_{\varphi}(s_{t-k}, a_{t-k}, a_{t-k+1}, \dots, a_{t-1}) - s_t\|^2 \quad (9)$$

The value of k , which is the length of the input sequences, can be defined. When $k = 1$, the predictions will be made only based on one past frame's states and action inputs. h_t is the hidden layers of RNN, which encodes the historical information as memory. a_t is the action input and s_t is the vehicle states at time t . \hat{s}_t is the predicted output given by the neural nets. ϕ is the activation function of neural nets. The loss is calculated by the mean squared error between last step's predicted output \hat{s}_t and the previously collected ground-truth s_t .

The update equations for each step's forward propagation of RNN are shown in equation (10) and equation (11), which have the bias vector c along with the weight matrices W, U, R, V , respectively for input-to-hidden, hidden-to-hidden, output-to-hidden and hidden-to-output connections.

$$h_t = \phi_1(Wa_t + Uh_{t-1} + Rs_{t-1}) \quad (10)$$

$$\hat{s}_t = \phi_2(Vh_t + c) \quad (11)$$

In our work, a gated recurrent structure called LSTM (Long Short-Term Memory) is used. LSTM is widely-used RNN structure for its ability to handle gradient vanishing or gradient exploding problems. The experimental results of LSTM will be presented in section IV.

C. Neural-Net-Based Dynamic Model Summary

To conclude, the neural-network-based dynamic models we introduced in our work have many advantages.

- 1) *Accurate*. It is more expressive and accurate when compared with the linear analytical models introduced in section II. The experiment results will be introduced later in section IV.
- 2) *Versatile*. By combining the longitudinal and lateral dynamic into one model, users can add, delete or replace any input features or output features with little effort and train the neural net model just in minutes. For example, in order to take the slope information (pitch angle) into consideration, users can replace the original 2-Dimensional heading angle with 3-Dimensional Euler Angle.
- 3) *Adaptive*. As the vehicle's dynamics can change from time to time due to the variance of the vehicle loading, weather or road conditions, the neural network dynamic model can be fine-tuned online in real-time to support adaptive controls. For example, the dynamic models can fine-tune the parameters of the last layer

of its neural network by the newly record samples and support model-based control algorithms, such as model predictive control (MPC).

- 4) *Scalable*. The dynamic modeling schema is easy to deploy and can be scaled to a large number of vehicles. The whole modeling process can be convenient and automatic. Once the training samples are collected, the dynamic models can be tuned or retrained with little human intervention, which can meet the need of large fleets of vehicles.

IV. MODEL TRAINING AND EXPERIMENT RESULTS

In this section, the autonomous vehicle's dynamic models are trained by real-world road data collected from Apollo autonomous vehicles driving on urban roads in Sunnyvale, California. We have done experiments with a bunch of vehicle types, including sedans, vans and last-mile delivery robot-cars, and we are going to present the experimental results of two vehicle models in this paper, shown in Table I.

As introduced in section V, the learning-based model's input features includes previous control commands and vehicle states. The output features are the predictions of current vehicle states. All the feature data will be collected from Apollo autonomous vehicles' CAN bus module and localization module at a fixed frequency of 100 HZ. The control commands (throttle, brake, steering) are read from the vehicle's CAN bus, which is used for the communication between the central controller and the vehicle's chassis. The vehicle's states, such as acceleration, angular speed, linear velocity and position, are read from Apollo localization modules, which utilize the sensors mounted on the autonomous vehicles, including IMU (Inertial Measurement Unit), LiDAR (Light Detection and Ranging) and GNSS (Global Navigation Satellite System). [19].

A. Offline Data Collection and Pre-processing

TABLE I: Vehicle Dimensions

	Lincoln MKZ	Ford Transit
Power	Hybrid	Electric
Size(m^3)	$4.93 \times 1.86 \times 1.47$	$4.82 \times 1.84 \times 1.85$
Weight(kg)	1,769	2,449
Type	Sedan	Van

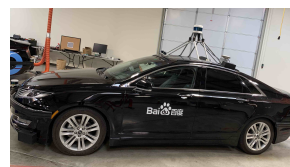


Fig. 3: Lincoln MKZ Sedan



Fig. 4: Ford Transit Van

TABLE II: Training Data Extraction Standard

Speed	0 - 5 m/s, 5 - 10 m/s, 10 - 15 m/s, 15 - 20 m/s, 20 - 25 m/s, above 25 m/s
Accelerator	Throttle: deadzone - 20%, 20% - 25%, 25% - 30%, 25% - 35%, 35% - 40%, 40% - 45%, 45% - 50%, above 50%
Decelerators	Brake: deadzone - 20%, 20% - 25%, 25% - 30%, 25% - 35%, 35% - 40%, above 40%
Steering Angle	left and right: 0 - 20%, 20% - 40%, 40% - 60%, 60% - 80%, 80% - 100%

Either manual driving data or autonomous driving data (as long as the throttle/brake/steering behaves the same under control commands) can be taken as training data. In order to make training data evenly distributed in input and output spaces, a feature extraction standard is given in Table II. The percentages of the accelerator, decelerator and steering angle are scaled by the upper limit of the actuation inputs. For example, the maximum steering angle of Lincoln MKZ sedan is 480-degree, then a steering control command of 240-degree to the left would be converted to 50%.

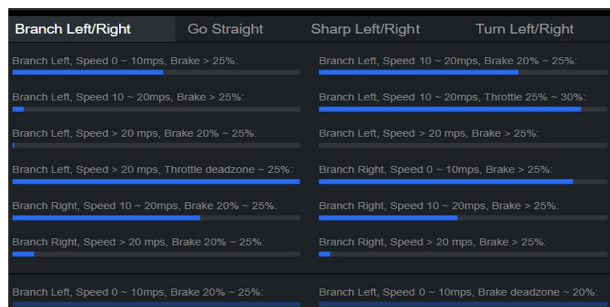


Fig. 5: Apollo Dynamic Model Data Collection Interface

We implemented a data collection interface integrated with Apollo simulation, as shown in Fig.5. We divided all the categories from table II into several groups, such as sharp left/right and go straight, regarding to their steering angles. Users can define the the amount of data frames they want for each category and the progress will be indicated by blue bars.

The collected data will then be processed in automated data pipeline, which includes data cleaning, auto-correction for any potential sensor shift, noise filtering and feature re-extraction if current samples are not sufficiently balanced. The data pipeline is built on Apache Spark, which is an open-source distributed cluster-computing framework, to handle large-scale data processing tasks.

B. Model Training and Evaluation Metrics

An analytic (rule-based) dynamic mode, a linear regression model and a set of non-linear neural network dynamic models (MLP and LSTM) are implemented and analyzed with the same input and output features. The input features contains the vehicle’s status as well as longitudinal and lateral control commands (throttle, brake, steering). The output predicts the small incremental values for the next step, including acceleration and angular velocity. Other vehicle status, such

as speed, heading and position can be calculated by the integral of the incremental values.

The rule-based dynamic model, as discussed in section II.A, is divided into a longitudinal model based on linear interpolation and a lateral model based on analytic bicycle model. In our implementation, the 2-Dimensional interpolation table contains 1297 records of the vehicle’s acceleration corresponding to the speed and throttle (or brake). The bicycle model is the same as introduced in section II.B, fitted with proper vehicle parameters, such as the length of wheelbase.

The machine learning models, including a linear regression model, a feed-forward MLP model, a MLP model considering time delays and a recurrent LSTM model, were initially trained on around 330,000 samples collected by the Lincoln MKZ on real roads in Sunnyvale, CA from August 28 to August 31, 2018. The samples, which were collected at a rate of 100 HZ, recorded the vehicle’s real-time control commands (throttle, brake, steering angle, gear) as well as the localization status (position, speed, acceleration, heading, angular velocity). The training dataset was automatically extracted to cover the different speed conditions and different maneuvers (turning right and left, stopping and restarting), as indicated in Table II. After the initial training, the models will be regularly re-trained with daily updated road test data from Apollo data platform. We also expanded the same process with similar data collection plan to Ford Transit as well.

The MLP model we implemented is a three-layer-perceptron, with a 5-dimensional input layer, a 10-dimensional hidden layer as well as a 2-dimensional output layer. The non-linear activation function we used for the hidden layer is ReLU while the output layer is fully-connected without activation functions. For the MLP model with delay, we exploit the same feed-forward structure but taking the vehicle states 30 ms after the inputs as output features, since the system delay between control commands and vehicle chassis is around 30 ms by phase measurements. For the LSTM, we place an 8-dimensional hidden layer with ReLU activation between the 20-step-long 5-dimensional input sequences and the 2-dimensional outputs.

The testing dataset is a complete one-minute driving record. Given the initial states and a sequences of action inputs, our dynamic models will predict the vehicle states in the next frames and rollout virtual trajectories in simulator. In order to evaluate the performances of dynamic models, the RMSE (rooted-mean-squared-error) between the predicted states (acceleration, angular speed, speed, heading, position) and the ground-truth states given by localization module will be calculated.

C. Experimental Results

As shown in Fig. 6, the grey lines represent for 1-minute’s raw data measured by the vehicles’ localization module and the green lines are the predicted results given by rule-based model. The pink lines are given by a simple linear regression model. The red, blue, and purple lines are the outputs of MLP, MLP model considering delays and LSTM.

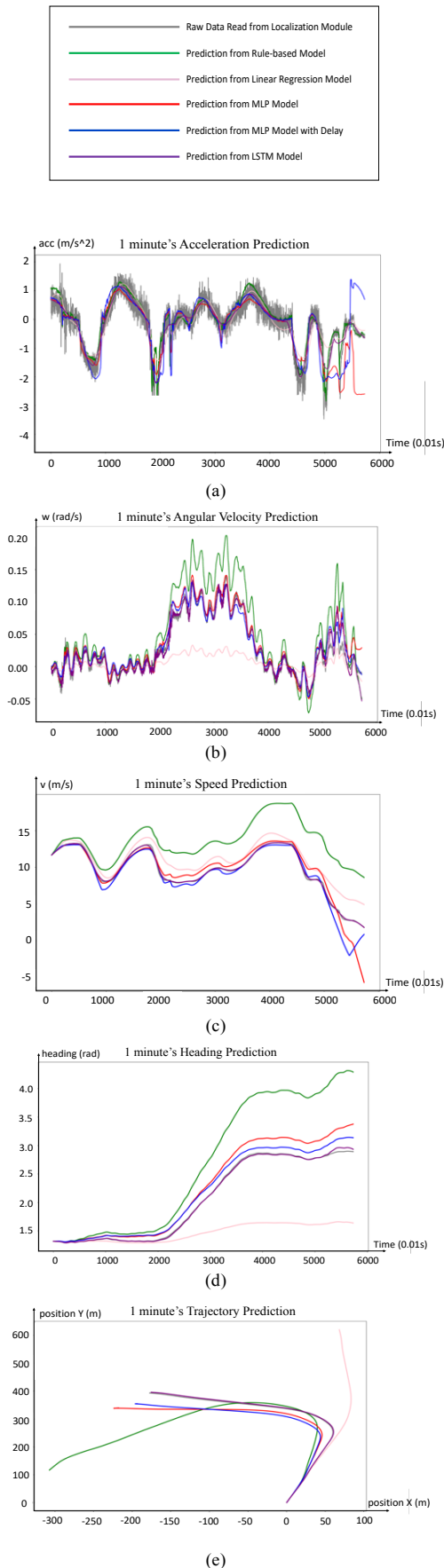


Fig. 6: Experiment Results

From Fig. 6 (c) to (e), we demonstrate the accumulated results of 1-minute's prediction. The speed prediction and heading prediction are calculated by the first-order integral of acceleration and angular speed over time. The trajectory prediction is calculated by second-order integral. In this way, we can visualize the accumulated errors.

In Table III and IV, we analyze the errors of the dynamic models' direct outputs, including acceleration and angular speed. Part of the prediction errors are caused by the errors and noises existing in the direct localization outputs of incremental states, such as acceleration and angular speed (as indicated in Fig. 6), which were mainly measured by the vehicle's GNSS and IMU. From Table V to Table VI, accumulated errors are listed. If the errors of our dynamic models is ϵ and the length of the trajectory is T , then the direct outputs' mean errors along the trajectory are expected to be $\propto \epsilon T$, while the first and second integral errors are $\propto \epsilon T^2$ and $\propto \epsilon T^3$. Since the 1-minute's raw data was collected at 100 HZ, the length of the trajectory T equals to 6000 in the experiment.

TABLE III: Direct Model Output RMSE of Lincoln sedan

RMSE	Acceleration (m/s^2)	Angular Speed (rad/s)
Rule-based model	0.73991	0.07615
Linear Regression	0.63318	0.04052
MLP	0.77724	0.06013
MLP with 30ms delay	0.61115	0.01025
LSTM	0.54993	0.00875

TABLE IV: Direct Model Output RMSE of Ford Transit van

RMSE	Acceleration (m/s^2)	Angular Speed (rad/s)
Rule-based model	1.27887	0.59255
Linear Regression	0.60719	0.16137
MLP	1.17480	0.10526
MLP with 30ms delay	0.90785	0.05517
LSTM	0.36027	0.04734

TABLE V: Incremental Output RMSE of Lincoln sedan

RMSE	Heading (rad)	Speed (m/s)	Location (m)
Rule-based Model	0.77454	4.34572	92.92622
Linear Regression	0.88032	2.60303	134.13891
MLP	0.21898	1.31107	28.04443
MLP with 30ms delay	0.12543	1.06349	20.24793
LSTM	0.02025	1.79645	5.92931

TABLE VI: Incremental Output RMSE of Ford Transit Van

RMSE	Heading (rad)	Speed (m/s)	Location (m)
Rule-based Model	5.17190	6.66788	57.96974
Linear Regression	3.00272	1.99159	49.28665
MLP	1.91477	2.14916	31.56056
MLP with 30ms delay	0.57855	1.27561	20.85167
LSTM	0.27776	1.21835	6.03559

From the results, we can easily conclude that our neural network models can decrease not only the direct errors but also the accumulated errors greatly. The errors (especially lateral errors) can be further reduced if we take the time delay and historical sequences into consideration.

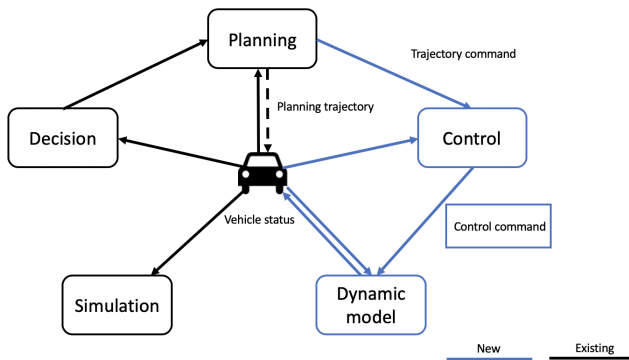


Fig. 7: Dynamic Model Integration with Simulation

D. Dynamic Model Online Simulation Integration

The rule-based and learning-based dynamic models share the same interfaces and both have been integrated into the backend of Apollo simulator, which can simulate vehicle's decision making, motion planning and control tasks. As shown in Fig. 7, black lines indicates the previous control-out-of-the-loop simulation, where the virtual vehicle's states directly follow the planning trajectory without considering real vehicles' dynamics and control algorithms. However, in this way, vehicle's performance could be very different in simulator compared with real-world self-driving tests. After integrated with the dynamic models proposed in this paper, developers can tune the control parameters and discover more potential problems in advance by control-in-the-loop simulation (shown in blue lines), which will further speed up the software iterations.

To prove the versatility of our learning-based dynamic models under different driving maneuvers, we demonstrate three typical simulating scenarios from Fig. 8 to Fig. 10. Fig. 8 showed a forward-driving scenario, where a pedestrian (indicated by the yellow bounding box in the upper part) suddenly walked across the street. As a result, the ego vehicle made a stop decision and applied the brake to avoid collision. The middle and lower part of Fig. 8 gave the vehicle's longitudinal speed and acceleration responses based on the control commands. The light green lines were the target values of speed and acceleration given by planning module. Given the target states, the control module calculated the proper commands for throttle, brake, steering and gear. For example, in Fig. 8, the control command was applying brake at 30%. Taken the control commands as model inputs, the learning-based dynamic models predicted the vehicle's next states, shown in blue lines. The gap between the green lines and blue lines represented the errors between planning and control module, which could only be tested on real vehicles in the past but now can be simulated with dynamic models.

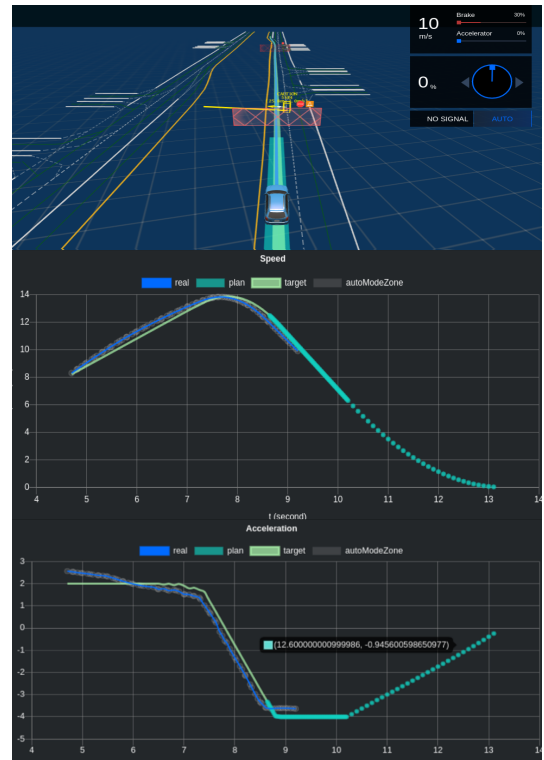


Fig. 8: Longitudinal Control Test under Forward Driving

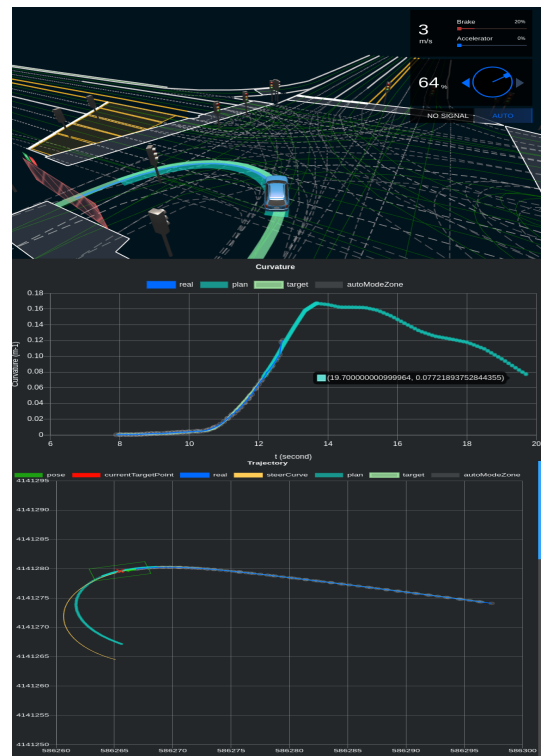


Fig. 9: Lateral Control Test under U-turn

ACKNOWLEDGMENT

The authors would like to specially thank Shu Jiang, Natasha Dsouza and Car Operations team in Baidu Apollo for helping with different aspects, from data collection to test validation and polishing this paper.

REFERENCES

- [1] Apollo Auto, 1 July 2019, github.com/ApolloAuto/apollo.
- [2] Gordon, N. J., D. J. Salmond, and A. F. M. Smith. "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation." *Radar and Signal Processing*, IEE Proceedings F 140.2(1993):107-113.
- [3] Nelles, Oliver. *Nonlinear System Identification: from Classical Approaches to Neural Networks and Fuzzy Models*. Springer, 2011.
- [4] Rong, Hai Jun, et al. "Sequential Adaptive Fuzzy Inference System (SAFIS) for nonlinear system identification and prediction." *Fuzzy Sets & Systems* 157.9(2006):1260-1275.
- [5] Ghahramani, Z. "Learning nonlinear dynamical systems using an EM algorithm." *Proc. Conf. Advances in Neural Information Processing Systems*, 1999 11(1999):431-437.
- [6] Langford, John, R. Salakhutdinov, and T. Zhang. "Learning nonlinear dynamic models." *International Conference on Machine Learning* 2009.
- [7] Shah, A. A., W. W. Xing, and V. Triantafyllidis. "Reduced-order modelling of parameter-dependent, linear and nonlinear dynamic partial differential equation models." *Proceedings Mathematical Physical Engineering Sciences* 473.2200(2017):20160809.
- [8] Ali Punjani, and Pieter Abbeel. *Deep Learning Helicopter Dynamics Models*. 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, doi:10.1109/icra.2015.7139643.
- [9] Guillaume Devineau, et al. *Coupled Longitudinal and Lateral Control of a Vehicle Using Deep Learning*. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, doi:10.1109/itsc.2018.8570020.
- [10] Arulampalam, M.s., et al. *A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking*. *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 2002, pp. 174188., doi:10.1109/78.978374.
- [11] Brunton, Steven L., J. L. Proctor, and J. N. Kutz. "Discovering governing equations from data: Sparse identification of nonlinear dynamical systems." *Proceedings of the National Academy of Sciences of the United States of America* 113.15(2015):3932.
- [12] Rudy, Samuel H., et al. "Data-driven discovery of partial differential equations." *Science Advances* 3.4(2017):e1602614.
- [13] Raissi, Maziar, P. Perdikaris, and G. E. Karniadakis. "Numerical Gaussian Processes for Time-dependent and Non-linear Partial Differential Equations." *Siam Journal on Scientific Computing* 40.1(2017):A172-A198.
- [14] Raissi, Maziar, P. Perdikaris, and G. E. Karniadakis. "Machine Learning of Linear Differential Equations using Gaussian Processes." *Journal of Computational Physics* 348(2017):S0021999117305582.
- [15] Beck, Christian, et al. "Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations." *ArXiv.org*, 18 Sep. 2017, arxiv.org/abs/1709.05963.
- [16] Anusha Nagabandi, et al. "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning." *ArXiv.org*, 2 Dec. 2017, arxiv.org/abs/1708.02596.
- [17] Theodosios Georgiou, and Yiannis Demiris. *Predicting Car States through Learned Models of Vehicle Dynamics and User Behaviours*. 2015 IEEE Intelligent Vehicles Symposium (IV), 2015, doi:10.1109/ivs.2015.7225852.
- [18] Grady Williams, et al. *Information Theoretic MPC for Model-Based Reinforcement Learning*. 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, doi:10.1109/icra.2017.7989202.
- [19] Guowei Wan, et al. *Robust and Precise Vehicle Localization Based on Multi-Sensor Fusion in Diverse City Scenes*. 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, doi:10.1109/icra.2018.8461224.

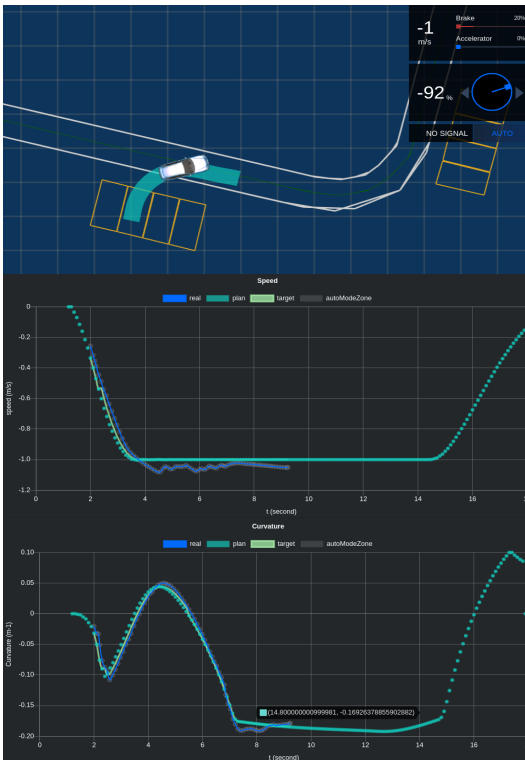


Fig. 10: Coupled Control Test under Reverse Parking

While Fig. 8 mainly tested the longitudinal control parameters, Fig. 9 demonstrated another case to test lateral control parameters under U-turn scenario. When the ego vehicle approached the intersection, the control module applied the brake and turned the steering wheel to the left in order to make a U-turn. The planned speed and curvature (green lines) as well as the simulated vehicle states (blue lines) were shown in the middle and lower parts in Fig. 9.

In most simulation cases, longitudinal and lateral controllers are tested in a coupled manner as presented in Fig. 10, where the ego vehicle was maneuvered to finish a reverse parking task. The planned speed and curvature were drawn in green lines while the predicted speed and curvature given by learning-based dynamic model were drawn in blue lines, as shown in the middle and lower part in Fig. 10.

V. CONCLUSIONS

The results show that with our proposed data-driven procedure and learning-based models, the vehicle dynamics adhere closely in simulation to the real vehicles' performances on road. What is more, the models are proven to be accurate, versatile, adaptive and scalable, thus easy to support large vehicle fleets of different brands for control-in-the-loop simulation. The data-driven procedure can also be useful in other meaningful applications, such as monitoring the vehicle health conditions from daily road test data (i.e. vehicle parts worn out) to further reduce the fleet management cost for autonomous driving.