Learning to Simulate Vehicle Trajectories from Demonstrations

Guanjie Zheng*§, Hanyang Liu*†, Kai Xu‡, Zhenhui Li§
The Pennsylvania State University§, Washington University in St. Louis†, Tianrang Inc.‡
gjz5038@ist.psu.edu, hanyang.liu@wustl.edu, kai.xu@tianrang-inc.com, jessieli@ist.psu.edu

***Abstract*—Traffic simulations can help to explore novel and efficient transportation solutions that overcome traffic problems such as traffic jams and road planning. Traditional traffic simulators usually leverage a car-following model to simulate the vehicle's behavior in the real-world traffic environment. However, these calibrated simplified physical models often fail to accurately predict the pattern of vehicle's movement in complicated real-world traffic environment. Considering the complexity and non-linearity of the real-world traffic, this paper unprecedentedly treat the problem of traffic simulation as a learning problem, and proposes learning to simulate (L2S) vehicle trajectory. We use the generative adversarial imitation learning framework to estimate the policy that provides sequential decisions for the vehicle given real-world demonstrations. The experiment on real-world traffic data shows the superior performance in simulating vehicle trajectories of our method compared to traditional traffic simulation approaches.**

***Index Terms*—Traffic simulation, imitation learning**

I. Introduction

Traffic simulation is the dynamics representation of the real-world traffic environment through mathematical or computer modeling. It acts as an essential component in the operation and planning of transportation systems. For example, a traffic simulator can be used to test the policies of a traffic signal plan, or generate sufficient simulation data for training an intelligent signal controller [1].

To build an effective traffic simulator that provides life-like microscopic simulations of vehicle movement, the key is to accurately imitate the observed vehicle behaviors and recover an individual vehicle's policy given current traffic state. Today's state-of-the-art and popular simulators such as SUMO [2], AIMSUN [3] and MITSIM [4] usually employ a so-called car-following model (CFM) to describe the kinematic movement of an individual vehicle. Basically, a CFM is a collection of several physical formulas that describe the car-following behaviors with empirically calibrated parameters, such as vehicle maximum acceleration and driver reaction time. However, the movement of real-world vehicles depends on many factors including speed, distance to neighbors, road networks, traffic signal, and even driver's psychological factors. Even with these parameters carefully calibrated, the CFM often fails to provide accurate policy estimation for the vehicle and exhibit realistic simulations, due to the nature of inferior approximation ability of the simplified physical models under intricate traffic environment. For example, Gipps model [5] (AIMSUN) is simply based on safety distance. It can never reflect other nonlinear factors such as the psycho-physical tendency in real-world drivers' decisions.

To overcome the deficiency in explaining real-world vehicle behaviors of current CFM based traffic simulators, a natural consideration is to learn the pattern of behaviors directly from real-world observations, instead of exclusively relying on unrealistic physical models. Imitation learning (IL), focused on making sequences of decisions similar to existing data, shows a promising avenue for learning from demonstrations [6]–[8]. Behavioral cloning (BC) [6], a variant of IL, is easy to implement due to its one-step supervised learning procedure, but the state distribution mismatch between training and testing makes it to perform poorly in practice. A solution to mitigate the accumulative mistakes in BC is to iteratively sample from the rollout of the trained policy, and thus maintain the consistency of state distribution. In this paper, we follow the generative adversarial imitation learning (GAIL) [8] that incorporates a GAN-like framework into an iterative learning procedure of IL, and extend it to the multi-agent context in the traffic simulation problem. Our proposed learning to simulate (L2S) takes each vehicle as an agent and directly learns a shared policy capable of controlling multiple vehicles simultaneously, through the alternation between a generator network and a discriminator network.

To the best of our knowledge, we are the first to consider the traffic simulation problem as a learning problem. The effectiveness of our proposed L2S is demonstrated by the experiment on real-world traffic data. Compared to traditional CFM based methods and BC based simulation, our method is shown to have superior performance in recovering the real-world vehicle trajectory.

II. Related Work

No previous approach on traffic simulation has considered learning-based methods before. But the idea of learning to simulate can be found in some other simulation problems. A few works as in [9]–[11] use reinforcement learning in pedestrian simulation and railway operational simulation. However, our method intrinsically differs from these works. They aim to train an agent to calibrate the parameters of either a selected physical model or a predefined distribution of generated data, while in our work the vehicle is as an agent and seeks to learn the policy from demonstrations on the vehicle level. The work [12] proposed to learn the autonomous driving policy

*The first two authors Guanjie Zheng and Hanyang Liu contribute equally.

from demonstrations, which has a similar formulation to our paper. But traffic simulation focus on the fidelity of simulated traffic flow and car-following behavior, as well as correct reactions to traffic light switch-over, while [12] ignores traffic signals, and only underlines accident avoidance.

In traditional simulators, usually a car-following model (CFM) is employed to simulate the individual vehicle's behaviors. The basic idea of the car-following theories is that the change in velocity of a vehicle depends on the velocity of the preceding vehicle as well as the gap between the leader and the follower. For instance, SUMO [2] uses the Krauss model [13] as the default CFM, with two key components: 1) safety speed, defined by a kinematic formula that considers the speed of the leading vehicle and a safety distance to it; 2) desired speed, incorporating the safety speed, speed limit and the acceleration of vehicles. With these rule-based CFMs, the simulator is enabled to simulate the individual vehicle's interactions with other vehicles and the road network. For calibrating a certain CFM, usually a heuristic search algorithm such as random search, Tabu search [14], and genetic algorithm [15] is used to select the best parameters for the CFM.

III. Problem Definition

The driving behavior of one vehicle can be formulated as a Markov Decision Process (MDP) defined by the tuple $\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma$. Here, $\mathcal{S}$ and $\mathcal{A}$ represent the state and action space correspondingly, $\mathcal{T}$ is the transition matrix between states, r is the reward function, and γ is discount for future reward. Thus, our problem can be formally defined as below.

Problem 1: Consider a set of M expert drivers with driving policy $\pi^{*,(m)}$, where $m = 1, 2, ...M$. Given their expert demonstrations $\mathcal{D} = \{\tau_1, \tau_2, ..., \tau_M\}$, our goal is to recover the driving policy $\pi^{(m)}$, so that $\pi^{(m)}$ is close enough to $\pi^{*,(m)}$.

The above formulation may seems not clear. It has been shown in [16] that imitating the policy from the trajectories is equivalent to maximizing the log likelihood of the trajectories being generated by the learned policy. Hence, the above problem is equivalent to the following maximization.

$$\max_{\pi^{(1)},\pi^{(2)},...,\pi^{(M)}} \sum_{m=1}^{M} \log P(\tau_m|\pi^{(m)}). \tag{1}$$

Here, $P(\tau_m|\pi^{(m)})$ is the probability of generating trajectory τ_m given that driver m is following policy $\pi^{(m)}$. Each trajectory is a sequence of states and actions, i.e., $\tau = s_0, a_0, s_1, a_1, ..., s_T, a_T$, where T is the total time span.

IV. Method

A. Overview

In order to learn how people drive in the real world, we can formulate each vehicle as an independent agent. Our goal is to learn the policy for each agent. The learned policy will serve as an engine in the traffic simulator that enables to produce real-like simulations.

The whole framework of our solution is shown in Figure 1. Each agent receives observations from the environment and takes actions sequentially following the policy. Then, by comparing the generated trajectories with the expert trajectories, a similarity score (actually a log likelihood) will be calculated to guide the optimization of the policy.

In Section IV-B, we introduce how each agent is defined and how they interact with each other and the environment. Then, we introduce how the policy for each agent is obtained via generative adversarial imitation learning in Section IV-C.

B. Multi-Agent Simulation System

Given the context that each driver is making their decisions independently based on his or her own observations, we model each driver as an agent m. For simplicity, we will omit the superscript (m) when no confusion would be introduced.

State: a set of features are selected to describe the current observations that the vehicle m receives:

- Road network: length of current lane, speed limit of current lane, whether current lane is an exit to the system.
- Current vehicle m: speed, position in current lane, distance to preceding traffic signal.
- Preceding vehicle m': speed of m', position of m' in lane, gap from m, whether m' in same lane as m.
- Traffic signal: phase of the preceding traffic signal.

Action: defined as the next-step speed for vehicle m.

In this paper, we assume that all drivers share homogeneous policy π. But with different state, each vehicle agent yields different actions. In other words, each agent m takes its own observation $s^{(m)}$ and gets its action $a^{(m)} = f_\pi(s^{(m)})$.

C. Generative Adversarial Imitation Learning

Imitation Learning. We utilize imitation learning approach to learn the policy from the pre-logged data. It has been shown that the imitation learning (or inverse reinforcement learning) problem can be reduced to the feature matching (for discrete-state problems) or state occupancy matching problem [8], [17] with some constraints (e.g., entropy), i.e., we aim to minimize the discrepancy with an entropy regularization H (as in [8]).

$$\underset{\pi}{\text{minimize}}\, d_(\rho_\pi, \rho^*) - H(\pi) \tag{2}$$

where ρ denotes the state action occupancy and d measures the discrepancy between the state occupancy of learned policy π and expert policy π^*.

Generative Adversarial Imitation Learning. As in [8], we replace the discrepancy measure d with a binary classification negative log loss. Then, we have the objective as [8]

$$\min_\theta \max_\psi \mathbb{E}_{\pi_\theta}[\log(D_\psi(s,a))] + \mathbb{E}_{\pi^*}[\log(1 - D_\psi(s,a))] - \lambda H(\pi_\theta) \tag{3}$$

where $D(s,a)$ is a binary classifier that discriminates generated trajectories from the expert ones.

Concretely, there are actually two steps in doing this optimization, discriminating step (D-step) and generating step (G-step). For the D-step, we can update the discriminator parameters to maximize

$$\max_\psi \mathbb{E}_{\pi_\theta}[\log(D_\psi(s,a))] + \mathbb{E}_{\pi^*}[\log(1 - D_\psi(s,a))] \tag{4}$$

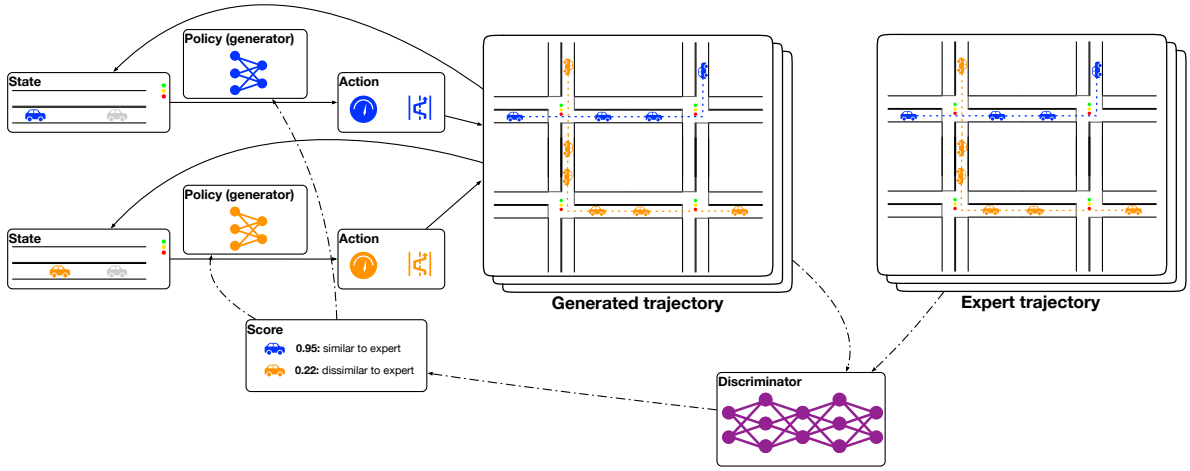

Fig. 1. Framework of proposed model. Each agent m receives observations from the current traffic environment, and takes actions based on the policy π (i.e., generator). The trajectories of each vehicle (formed by its state representations and actions) are compared against the expert trajectories. The similarity score obtained from the comparison guides the optimization of the policy towards the real-world policy.

Then, for the G-step, we optimize the policy towards the direction that the generated trajectories are more similar to the expert ones, using any reinforcement learning algorithms. Mathematically, the objective of G-step is

$$\min_{\theta} \mathbb{E}_{\pi_\theta}[\log(D_\psi(s,a))] - \lambda H(\pi_\theta) \tag{5}$$

Specifically, for instance, [8] uses $\log(D(s,a))$ as the cost to guide the optimization of the policy (or use the negative log likelihood as the reward).

For the policy optimization algorithm, we follow [18] and employ proximal policy optimization (PPO) algorithm to update the policy. By doing the D-step and G-step iteratively, we can finally get π_θ close enough to the expert policy π^*.

V. Experiment

A. Dataset

Three real world datasets are used for the experiments.

- **Hangzhou (HZ)**. This dataset is composed of surveillance camera records from various 4-way intersections in Hangzhou, China. Each record has the attribute vehicle id, arriving location and arriving time. We input this data into CityFlow [19] controlled by a pre-set driving policy to generate the expert trajectories.
- **Gudang (GD)**. This dataset covers a 4×4 road network in the Gudang area in Hangzhou (also used in [20], [21]). This data has been collected in the same fashion as the HZ dataset. We input it into CityFlow [19] like the HZ dataset.
- **Los Angeles (LA)**. This dataset is collected from Lankershim Boulevard, Los Angeles on June 16, 2005 and made public online [1]. This dataset records how each vehicle proceed (e.g., location and speed) on a 1×4 road network, in the temporal resolution of 0.1 second. These trajectories are used directly as the expert trajectories.

[1]https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm

B. Compared Methods

We compare our method **L**earning to **S**imulate (L2S) with two types of baseline methods: traditional CFM based methods and learning based methods. For the CFM based methods, we select the popular Krauss model [13] as the CFM and employ two search methods that are frequently used in calibrating these parameters.

- **CFM-RS**: calibrating the parameters with random search. The algorithm will randomly generate one set of parameters for the CFM in each round. The set of parameters that yields the best results are kept.
- **CFM-TS**: calibrating the parameters with Tabu search [14]. Starting from an initial set of parameter, a new set of parameters is chosen within the proximity of the current parameters in each round.

In order to better evaluate the learning performance of our method, we also apply **BC** [6] in vehicle trajectory learning and compare it to our method.

C. Experiment Settings

Our experiments are conducted in a traffic simulation platform called CityFlow [19]. This simulator can take traffic flow as input and simulate how each vehicle run towards its destination (including its interaction with traffic signals and other vehicles). For HZ and GD dataset, since they only have the arriving flow of the traffic, we will feed the traffic into the simulator (controlled by a pre-set driving policy) and generate the expert trajectories. For LA dataset, we directly use the provided trajectories as the expert. Then, we use different methods to recover the driving policy and compare the recovered trajectories w.r.t. the expert ones, in terms of root mean square error (RMSE) of the speed and position.

TABLE I

PERFORMANCE COMPARISON OF L2S WITH STATE-OF-ART METHODS IN TERMS OF RMSE OF RECOVERED VEHICLE POSITION (M) AND SPEED (M/S). RELATIVE IMPROVEMENTS ARE CALCULATED OVER THE BEST BASELINE. OUR PROPOSED L2S SHOWS BEST PERFORMANCE ON ALL DATASETS.

Method	HZ-1		HZ-2		HZ-3		GD		LA	
	Pos	Speed	Pos	Speed	Pos	Speed	Pos	Speed	Pos	Speed
CFM-RS	173.0	5.3	153.0	5.6	129.0	5.7	286.0	5.3	1280.9	10.3
CFM-TS	188.3	5.8	147.0	6.1	149.0	6.1	310.0	5.5	1294.7	10.8
BC	225.0	5.7	151.0	5.2	170.0	6.1	485.0	5.6	1003.3	7.6
L2S	**147.1**	**4.3**	**66.9**	**2.4**	**98.5**	**4.1**	**157.6**	**2.5**	**749.0**	**5.7**
Improvement	15.0%	18.9%	54.5%	53.8%	23.6%	28.1%	44.9%	52.8%	25.3%	25.0%

Concretely, we have

$$RMSE = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(s_t^{(m)} - \hat{s}_t^{(m)}\right)^2} \qquad (6)$$

where $s_t^{(m)}$ and $\hat{s}_t^{(m)}$ are the true value and recovered value of the position or speed of vehicle m at timestamp t respectively.

D. Results

We compare L2S with the other methods on the three datasets. The results are shown in Table I. L2S outperforms all the baseline methods in terms of RMSE in position and speed (compared with the expert trajectories) with improvement of at least 15%. It is observed that, the calibration methods (CFM-RS and CFM-TS) can achieve reasonable results on single-intersection cases in HZ dataset, but perform very poorly in multi-intersection cases GD and LA. The inferior results of BC is mainly due to its failure in modeling interactions between vehicles, which is more obvious in multi-intersection cases.

VI. CONCLUSION

In this paper, we formulate traffic simulation as an imitation learning problem and directly learn the pattern of individual vehicle's behaviors from real-world demonstrations. Through experiments we show the superior performance of our method in recovering real-world vehicle trajectory. For future work, we will work on improving the model's generalization ability in different traffic environment.

ACKNOWLEDGMENT

This work was supported in part by NSF awards #1652525, #1618448 and #1639150. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2496–2505.

[2] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[3] J. Barceló and J. Casas, "Dynamic network simulation with aimsun," in *Simulation approaches in transportation analysis*. Springer, 2005, pp. 57–98.

[4] Q. Yang and H. N. Koutsopoulos, "A microscopic traffic simulator for evaluation of dynamic traffic management systems," *Transportation Research Part C: Emerging Technologies*, vol. 4, no. 3, pp. 113–129, 1996.

[5] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[6] D. Michie, M. Bain, and J. Hayes-Miches, "Cognitive models from subcognitive skills," *IEEE control engineering series*, vol. 44, pp. 71–99, 1990.

[7] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *International Conference on Machine Learning (ICML)*, vol. 1, 2000, p. 2.

[8] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 4565–4573.

[9] Y. Cui, U. Martin, and W. Zhao, "Calibration of disturbance parameters in railway operational simulation based on reinforcement learning," *Journal of Rail Transport Planning & Management*, vol. 6, no. 1, pp. 1–12, 2016.

[10] F. Martinez-Gil, M. Lozano, and F. Fernández, "Calibrating a motion model based on reinforcement learning for pedestrian simulation," in *International Conference on Motion in Games*. Springer, 2012, pp. 302–313.

[11] N. Ruiz, S. Schulter, and M. Chandraker, "Learning to simulate," *arXiv preprint arXiv:1810.02513*, 2018.

[12] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1534–1539.

[13] S. Krauß, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics," Ph.D. dissertation, Dt. Zentrum für Luft-und Raumfahrt eV, Abt. Unternehmensorganisation und ..., 1998.

[14] C. Osorio and V. Punzo, "Efficient calibration of microscopic car-following models for large-scale stochastic network simulators," *Transportation Research Part B: Methodological*, vol. 119, pp. 156–173, 2019.

[15] A. Kesting and M. Treiber, "Calibrating car-following models by using trajectory data: Methodological study," *Transportation Research Record*, vol. 2088, no. 1, pp. 148–156, 2008.

[16] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 8, 2008, pp. 1433–1438.

[17] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the International Conference on Machine learning (ICML)*. ACM, 2004, p. 1.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[19] H. Zhang, S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, W. Zhang, Y. Yu, H. Jin, and Z. Li, "Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario," in *International World Wide Web Conference (WWW) Demo Paper*, 2019.

[20] G. Zheng, Y. Xiong, X. Zang, J. Feng, H. Wei, H. Zhang, Y. Li, K. Xu, and Z. Li, "Learning phase competition for traffic signal control," *arXiv preprint arXiv:1905.04722*, 2019.

[21] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, and Z. Li, "Colight: Learning network-level cooperation for traffic signal control," *arXiv preprint arXiv:1905.05717*, 2019.