Framework for Control and Deep Reinforcement Learning in Traffic

Cathy Wu*, Kanaad Parvate*, Nishant Kheterpal*, Leah Dickstein*,
Ankur Mehta†, Eugene Vinitsky‡, Alexandre M Bayen*§
*UC Berkeley, Electrical Engineering and Computer Science
†UCLA, Electrical Engineering
‡UC Berkeley, Mechanical Engineering
§UC Berkeley, Institute for Transportation Studies

***Abstract*—Recent advances in deep reinforcement learning offer an opportunity to revisit complex traffic control problems at the level of vehicle dynamics, with the aim of learning locally optimal (with respect to the policy parameterization) policies for a variety of objectives such as matching a target velocity or minimizing fuel consumption. In this article, we present a framework called CISTAR (Customized Interface for SUMO, TraCI, and RLLab) that integrates the widely used traffic simulator SUMO with a standard deep reinforcement learning library RLLab. We create an interface allowing for easy customization of SUMO, allowing users to easily implement new controllers, heterogeneous experiments, and RL cost functions that depend on arbitrary state variables. We demonstrate the usage of CISTAR with several benchmark control and RL examples.**

***Index Terms*—Simulation; deep reinforcement learning; control; vehicle dynamics**

I. Introduction

Modeling and analysis of traffic dynamics is notoriously complex and yet is a prerequisite for model-based traffic control [1], [2]. Researchers classically trade away the complexity of the model (and thus the realism of the model) in favor of the tractability of analysis, often with the goal of designing optimal controllers or other controllers with desirable properties, such as safety or comfort [3], [4], [5], [6]. Consequently, results in traffic control can largely be classified as simulation-based numerical analysis (for example, [7], [8], [9], [10]) or theoretical analysis on simple settings such as assuming non-oscillatory responses (e.g. [11]) or focusing on a single-lane ring road (e.g. [12], [13], [14], [15], [16], [17]). In the present article, we largely focus our discussion on microscopic longitudinal dynamics, also called car following models [18], but our proposed framework largely extends to other dynamics such as lateral dynamics [19] and coordinated behaviors [20].

Deep reinforcement learning (RL) is a powerful tool for control and has already had demonstrated success in complex but data-rich problem settings such as Atari games [21], 3D locomotion and manipulation [22], [23], [24], chess [25], among others. RL testbeds exist for different problem domains, such as the Arcade Learning Environment (ALE) for Atari games [26], DeepMind Lab for a first-person 3D game [27], OpenAI gym for a variety of control problems [28], FAIR TorchCraft for Starcraft: Brood War [29], MuJoCo for multi-joint dynamics with Contact [30], TORCS for a car racing game [31], among others. DeepMind and Blizzard will collaborate to release the Starcraft II AI research environment [32]. Each of these RL testbeds enables the study of control through RL of a specific problem domain by taking advantage of the data-rich setting of simulation. One of the primary goals of this article is to present a similarly suitable RL testbed for traffic dynamics by making use of an existing traffic simulator.

The main contributions of this article are as follows:

- We introduce the **C**ustomized **I**nterface for **S**UMO, **T**raCI, **a**nd **R**LLab (CISTAR) library, a framework for reinforcement learning and control experiments for traffic microsimulation.
- CISTAR integrates the traffic simulator SUMO with a standard deep reinforcement learning library RLLab.
- CISTAR extends SUMO to support high frequency simulation and more flexibility in controllers.
- Using CISTAR, we replicate and demonstrate new control experiments for single- and multi-lane circular roads, including homogeneous and non-homogeneous vehicle types.
- Using CISTAR, we demonstrate the first deep reinforcement learning experiments on traffic at the level of multi-agent vehicle control.

II. Related Work

A. Deep learning and traffic

Several recent studies incorporated ideas from deep learning in traffic optimization. Deep learning has been used for traffic prediction [33], [34] and control [35]. A deep learning architecture was used in [34] to predict traffic flows, demonstrating success even during highly nonlinear special events; to learn features to represent states involving both space and time, [33] additionally used hierarchical autoencoding in the traffic flow prediction problem. A multi-agent deep reinforcement learning algorithm was introduced in [35] to learn a policy for ramp metering. For additional uses of deep learning in traffic, we refer the reader to [36], which presents an overview comparing non-neural statistical methods versus neural networks in transportation research. These recent results demonstrate that deep learning and deep reinforcement learning are a

promising approach to traffic problems. Our work aims to further this by providing a framework for traffic experiments using reinforcement learning for control.

B. Traffic simulators

Traffic microsimulators include Quadstone Paramics [37], VISSIM [38], [39], AIMSUN [40], MATSIM [41], and SUMO [42]. The first three are commercial, whereas the latter two are open source software. Each of these tools are capable of large-scale traffic microsimulation and can handle a variety of policies and control strategies. Each tool offers an Application Programming Interface (API) which permits overriding or extending the default models such as car following, lane changing, route choice, etc. Each of these simulators are widely used in the research community. These tools differ in their precise offerings and features, such as visualization tools, supported models, and simulation speed. Because most studies focus their study on a single simulator, a comprehensive comparison of these tools is unfortunately lacking.

In our work, we choose to integrate SUMO, an open-source, extensible, microscopic simulator that can simulate large road networks. SUMO discretizes time and progresses the simulation for a user-specified timestep; furthermore, because SUMO is microscopic, individual vehicles are controlled by car following models—functions of the vehicle's headway, velocity and the velocity of the preceding vehicle. The acceleration provided by the car following model is applied as a change of velocity over the course of the next timestep. SUMO's car following models include IDM, IDMM, and Wiedermann.

Because the results of an RL experiment rely on the realism of the model/simulator, we need the traffic models to capture more realistic fine-grained dynamics, including operating at a higher granularity (smaller simulation step), with a different model of time delays, with acceleration-based control, etc. SUMO, in particular, has several current issues which limit its suitability for RL. Firstly, all built-in car following models are configured with a minimal time headway, τ, that is used to ensure safety [43], and do not support time delays. Secondly, SUMO's car following models are calibrated for a simulation timestep of 1.0 seconds, and their behavior for smaller timesteps is known to produce unnatural behaviors, [44] whereas we would like to simulate at 10-100ms timesteps. Finally, there does not yet exist an interface between SUMO and reinforcement learning libraries.

Our work aims to address each of these limitations. CISTAR extends SUMO to permit rich custom controllers which may operate at smaller simulation steps, with time delays, and be acceleration-based controllers. These richer control actions allow SUMO to support a larger class of controllers, thus permitting a more realistic and suitable testbed for reinforcement learning. SUMO also includes a Python API called TraCI (TRAffic Control Interface), from which the user can retrieve information about the vehicles' current states and issue precise commands to set the vehicles' velocities, positions, lanes. Using this interface, we can interface SUMO with RL libraries, read out state information, issue actions, define our own car following models, etc.

III. Preliminaries

In this section, we describe two well-studied topics, which are key to understanding CISTAR: longitudinal dynamics [12] and Markov decision processes [45]. Longitudinal dynamics describe the forwards-backwards control of vehicle control models. Markov decision processes is the problem framework under which reinforcement learning optimizes policies.

A. Longitudinal dynamics

Longitudinal dynamics are usually defined by car following models [12]. Standard car following models (CFMs) are of the form:

$$a_i = \dot{v}_i = f(h_i, \dot{h}_i, v_i), \tag{1}$$

where the acceleration a_i of vehicle i is some typically nonlinear function of $h_i, \dot{h}_i, v_i$, which are respectively the headway, relative velocity, and velocity for vehicle i. A general model may include time delays from the input signals $h_i, \dot{h}_i, v_i$ to the resulting ouput acceleration a_i. Example CFMs include the Intelligent Driver Model (IDM) [46] and the Optimal Velocity Model (OVM) [47], [48]. Our presented system implements several known CFMs and provides an easy way to implement custom CFMs.

B. Markov decision processes and reinforcement learning

Reinforcement learning problems are typically studied in the framework of Markov decision processes (MDPs) [45], [49]. A MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma, T)$, where $\mathcal{S}$ is a (possibly infinite) set of states, $\mathcal{A}$ is a set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the initial state distribution, $\gamma \in (0, 1]$ is the discount factor, and T is the horizon.

Reinforcement learning addresses the problem of how agents should learn to take actions to maximize cumulative reward through interactions with the environment. We use a class of reinforcement learning algorithms called policy gradient methods [50], which optimize a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$. OpenAI's RLLab is an open source framework that facilitates running and evaluating reinforcement learning algorithms on a variety of different scenarios, from classic tasks such as cartpole balancing to more complicated tasks such as 3D humanoid locomotion [51]. To perform an experiment with RLLab, we must first define an environment encapsulating the MDP or problem setting, such as velocity matching on a ring road.

In the next section, we introduce CISTAR, which defines environments to capture various traffic problem settings and uses RL libraries to achieve user-defined learning goals.

IV. Overview of CISTAR

CISTAR encapsulates the use of SUMO with TraCI that allows them to support the MDP required for a reinforcement learning experiment in RLLab. After initializing the simulation with a number of vehicles in some initial configuration, RLLab collects samples by stepping through the simulation and then resets the simulation when the simulation is terminated. In each step, the vehicles are provided actions through a controller or through a learned policy. These actions are then applied via TraCI and the simulation progresses. At the end of an episode, RLLab issues a reset command to the environment, which returns vehicles to their initial position.

CISTAR can be used to perform purely control theoretic experiments, by only relying on existing controllers for actions; or it can be used for experiments with a mix of existing and learned controllers, such as heterogeneous or mixed autonomy experiments. Vehicles can have both a longitudinal and a lateral controller, with longitudinal safety guaranteed by supported fail-safe models (see Section V-A).

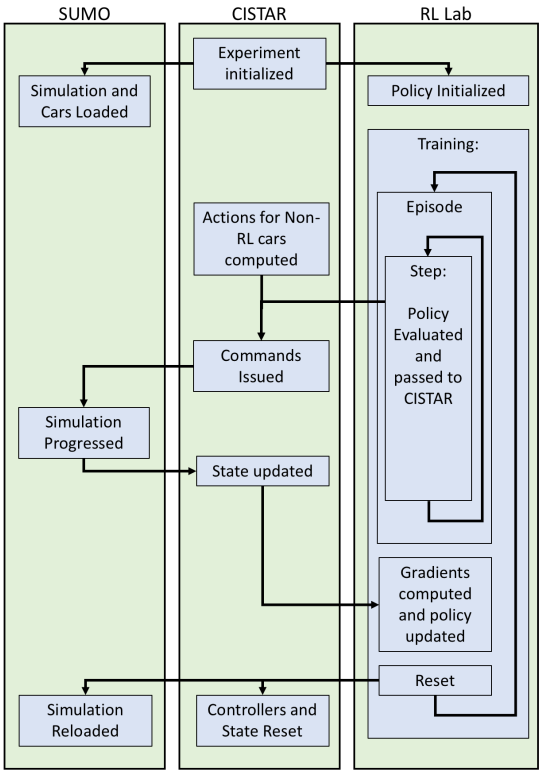

Fig. 1: CISTAR Process Diagram.

A. Architecture of CISTAR

Experiments are initialized and run using CISTAR by defining two components: a scenario and an environment.

The **scenario** for an experiment specifies network configuration (i.e. shape and attributes, e.g. two-lane loop road with circumference 200m). Based on the specifications provided, the net and configuration files needed by SUMO are generated. The user also specifies the number and types of vehicles (car following model and a lane-change controller), which will be placed in the scenario.

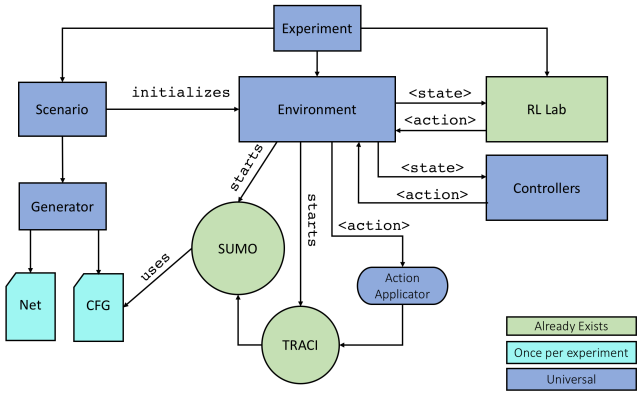

Fig. 2: CISTAR Architecture (See Section IV-A).

The **generator** is a predefined class, which allows for rapid generation of scenarios with user-defined sizes, shapes, and configurations. The experiments presented in this article use a large loop road generated by specifying the number of lanes and ring circumference.

The **environment** encodes the MDP, including functions to step through the simulation, retrieve the state, sample and apply actions, compute the reward, and reset the simulation. The environment is updated at each timestep of the simulation and, importantly, stores each vehicle's state (e.g. position and velocity). Information from the environment is provided to a controller or passed to RLLab to determine an action for a vehicle to apply, (e.g. an acceleration). Note that the amount of information provided to either RL or to a controller can be restricted as desired, thus allowing fully observable or partially observable MDPs. The experiments presented in this article are fully observed (full knowledge of all vehicle positions and velocities).

When provided with actions to apply, CISTAR calls the **action applicator** which uses TraCI to enact the action on the vehicles. Because TraCI can only set velocities, not accelerations, we convert the acceleration into an instantaneous $\delta v = \alpha \cdot dt$, where α is the acceleration and dt is the simulation time step-size.

V. Features of CISTAR

A. Fail-safes

CISTAR supplements its car following models with a variety of safe driving rules that prevent the unstable car following models from crashing. As SUMO experiments terminate when a collision occurs, we built a fail-safe mechanism, which must run constantly. Fail-safes are passed in the action commanded by the vehicle controller, regardless of whether that is an action specified by RL or one specified by a control model. Fail-safes are a standard feature in any traffic simulator that is required to handle large perturbations and string unstable traffic. The conservativeness of the fail-safe affects the braking behavior of the traffic. In general, fail-safes operate according

to the principle of maintaining a minimum safe distance from the leading vehicle where the maximum acceleration and deceleration of the leading vehicle is stochastically generated [52], [53]. The current implementation of CISTAR does not account for this second half and assumes that all vehicles are equipped with the same deceleration.

1) Final Position Rule: For this fail-safe we aim to keep a velocity such that if the preceding vehicle suddenly starts braking with max deceleration a, then even if the following vehicle has a delay τ it can still slow down such that it comes to rest at the final position of the rear bumper of the preceding vehicle. If the preceding vehicle is initially at position $x_{i-1}(0)$, and decelerates maximally, it will come to rest at position $x_{i-1}(0) + \frac{v_{i-1}^2(0)}{2a}$. Because we are calculating the maximum velocity, if the second vehicle has delay τ, it will first travel a distance of $v_{\text{safe}}\tau$ and then begins to brake with maximum deceleration, which brings it to rest at position $x_i(0) + v_{\text{safe}} \cdot \left(\tau + \frac{v_{\text{safe}}}{2a}\right)$.

B. Longitudinal controllers

CISTAR supports a variety of car following models, including default models from SUMO and custom models not provided by SUMO. Each model specifies the acceleration for a vehicle at a given time, which is commanded to that vehicle for the next time-step using TraCI. Controllers with arbitrary time delays between perception and action are supported in CISTAR. Delays are implemented by storing control actions in a queue. For delayed controllers, a new action is computed using the state at each timestep and enqueued, and an action corresponding to some previous state is dequeued and commanded. Descriptions of supported car-following models follow below.

1) Second-order linear model: The first, and simplest, car following model implemented is the forward-looking car following model specified in [12]. The model specifies the acceleration of vehicle i as a function of a vehicle's current position and velocity, as well as the position and velocity of the vehicle ahead. Thus: $\dot{v}_i = k_d(d_i - d_{\text{des}}) + k_v(v_{i-1} - v_i) + k_c(v_i - v_{\text{des}})$ where v_i, x_i are the velocity and position of the i-th vehicle, $d_i := x_{i-1} - x_i$ is the headway for the i-th vehicle, k_d, k_c, k_v are controller gains for the difference between the distance to the leading car and the desired distance, relative velocity, and the difference between current velocity and desired velocity, respectively. $d_{\text{des}}, v_{\text{des}}$ are desired headways and velocities respectively.

2) Optimal Velocity Model (OVM): Another car following model implemented in CISTAR is the optimal velocity model from [14]. A variety of optimal velocity functions exist for use in specifying car following models [54], [1]; [14] uses a cosine-based function to define optimal velocity $V(h)$ as a function of headway. They define

$$V(h) = \begin{cases} 0 & h \leq h_{\text{st}} \\ \frac{v_{\text{max}}}{2}(1 - \cos(\pi \frac{h - h_{\text{st}}}{h_{\text{go}}} - h_{\text{st}})) & h_{\text{st}} < h < h_{\text{go}} \\ v_{\text{max}} & h \geq h_{\text{go}} \end{cases}$$

The values $h_{\text{st}}, h_{\text{go}}$ correspond to headway thresholds for choosing an optimal velocity, so that for headways below h_{st}, the optimal velocity is 0, and for headways above h_{go}, the optimal velocity is some maximum velocity v_{max}. The optimal velocity transitions using a cosine function for headways between h_{st} and h_{go}. $V(h)$ is used in the control law for the acceleration of the i-th vehicle, where $\dot{v}_i = \alpha[V(h_i) - v_i] + \beta[v_{i-1} - v_i]$ at each timestep. This controller can also be implemented with delay to simulate perception and reaction times for human drivers, in which case $\dot{v}_i(t)$ would be a function of states $h_i(t-\tau), v_i(t-\tau), v_{i-1}(t-\tau)$.

3) Bilateral control model (BCM): The bilateral controller presented by [15], [16] considers not only the relation of a subject vehicle to the vehicle ahead but also to the vehicle behind it. In their controller, the subject vehicle's acceleration depends on the distance and velocity difference to both the vehicle ahead and behind, with

$$\dot{v}_i = k_d h_i + k_v((v_{i-1} - v_i) - (v_i - v_{i+1})) + k_c(v_i - v_{\text{des}})$$

where $h_i := (x_{i-1} - x_i) - (x_i - x_{i+1})$.

C. Lateral controllers

SUMO comes with models dictating when and how to lane change [55]; however, in order to provide a means to conduct RL-based lane changing experiments, CISTAR introduces a framework to easily design new lane changing models and optimize them for small time-steps. The current implementation of CISTAR includes preliminary lane-changing models in which vehicles change lanes stochastically, if adjacent lanes satisfy a set of constraints. CISTAR vehicles do not check to change lanes at each timestep, as that might lead to an excessive number of lane changes. Instead, at some time interval, the vehicle determines if it should lane change. This rudimentary controller provides an initial scaffold for supporting more sophisticated lane changing behavior, which is the subject of ongoing work.

The initial lane-changing model iterates through each adjacent lane, determining which lane offers the highest speed and storing the speed of traffic for each lane in an array. It begins by identifying the gap available in each lane for a controlled vehicle to move into, by determining the distances to the vehicles ahead and behind of where the subject vehicle would be in that lane. If the gap in some lane is too small, that lane's available speed is set to 0 in the array. If the gap is large enough, the model sets the lane's speed in the array to the average speed of all the vehicles some distance ahead and behind of the controlled vehicle in that lane. Once all lanes have been investigated, the lane with the maximum available speed is identified using the array. If the subject vehicle is not in the optimal lane, a lane-change maneuver to the optimal lane is commanded using TraCI with some probability.

D. Heterogeneous settings

CISTAR supports traffic settings with heterogeneous vehicle types. By initializing different "types" of vehicles, each with its own controllers and parameters, vehicles with different

controllers be simulated simultaneously. Additionally, simulations can contain both RL-controlled vehicles and control-based vehicles. This permits the use of CISTAR for mixed autonomy experiments.

E. Perturbations

Because of the fine-grained control over vehicles and their motion, arbitrary perturbations can be specified in an experiment. For example, the experiments defined below randomly choose and perturb a vehicle by overriding its control inputs and commanding a deceleration for some duration.

F. Markov decision processes (MDPs)

CISTAR supports a large range of traffic MDPs, including custom reward functions, full and partially observed states, full and partial controlability, noisy observations, etc. For instance, the controller may observe only local information from only the preceding vehicle, only nearby vehicles, or all vehicles. The reward function can be any function of vehicle speed, position, fuel consumption, acceleration, distance elapsed, etc.

VI. CISTAR USE CASES

This section details various example experiments using CISTAR, demonstrating the capability of the system to simulate homogeneous and mixed vehicle traffic in ring-road settings from arbitrary initial conditions and including perturbations.

A. Experimental Scenario

In the well-known result of [56], Sugiyama demonstrates in a physical experiment on a single-lane road of length 230m that 22 vehicles traveling at 8.3m/s produce backwards propagating waves, causing part of the traffic to come to a complete stop. Each experiment ran for 100 seconds. As a demonstrative example, we aim to reproduce their experiment in CISTAR. We additionally test various iterations of it, observing the resulting behavior.

B. Control experiments

1) OVM from uniform flow (Figure 3): The first experiment runs the Sugiyama setup from an initial state in which all 22 vehicles were spaced evenly around the ring road and start with the same velocity (also called uniform flow). Each of the vehicles was using a Optimal Vehicle Model (OVM) controller, as described in the section on controllers above. The experiment begins from a stopped state, gets up to speed, and proceeds free of traffic shockwaves for its duration.

2) OVM from a nonuniform flow state (Figure 4): This experiment simulates the Sugiyama setup but from a non-uniform initial configuration. Starting with the first vehicle, the subsequent position of each vehicle is drawn from a Gaussian distribution with mean equal to the length of track divided by number of vehicles and a standard deviation given by one fifth the mean. The unstable starting state also incorporates a bunching factor, in which no vehicles are placed on some segment of the track, with the length of that segment being a user-defined variable. All 22 vehicles use the OVM controller. Instability is apparent from the beginning, with traffic rapidly degrading into traffic shockwaves and failing to recover.

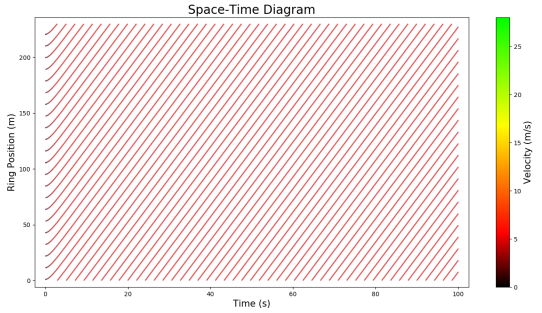

Fig. 3: OVM from uniform flow, showing an average speed of 4.87 m/s across all vehicles.

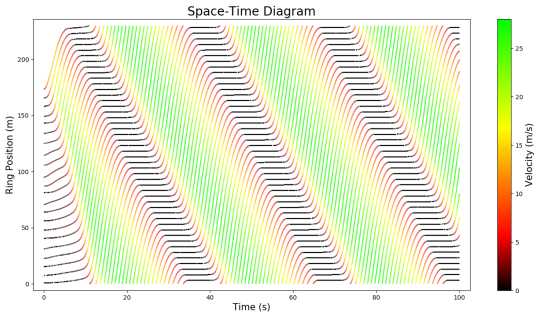

Fig. 4: OVM from a nonuniform state, showing stop-and-go traffic with an average speed of 7.8 m/s.

3) OVM with a perturbation (Figure 5): In this experiment, 22 OVM vehicles are run from a uniform, evenly-spaced starting state. No traffic shockwaves form until the system is perturbed 9 seconds into the experiment, once the vehicles have roughly reached their equilibrium velocities from the unperturbed scenario. One vehicle is randomly chosen and an acceleration of -5 m/s^2 is applied for 1.5 seconds. The braking of that vehicle forces the vehicles behind it to slow down as well, and the system degrades into stop-and-go traffic.

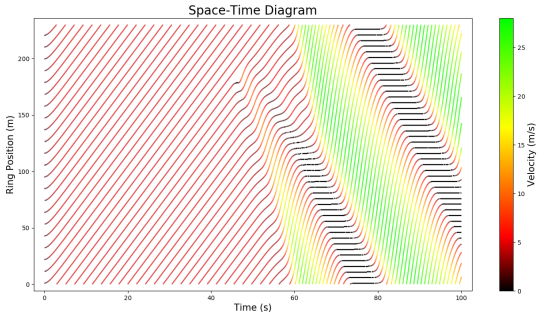

Fig. 5: OVM with a perturbation, breaking down from uniform flow into stop-and-go traffic with an average speed of 7.5 m/s.

4) BCM with a perturbation (Figure 6): 22 vehicles implementing the bilateral car following model (BCM), described in the controllers section, are implemented in this simulation. The simulation begins from a uniform, evenly-spaced starting state. As with the experiment above, a random vehicle is perturbed

at an acceleration of $-5\text{m}/\text{s}^2$, 9 seconds into the simulation for 1.5 seconds. Some braking results, but unlike the OVM case described above, the BCM vehicles recover from this perturbation and traffic returns to uniform flow shortly after.

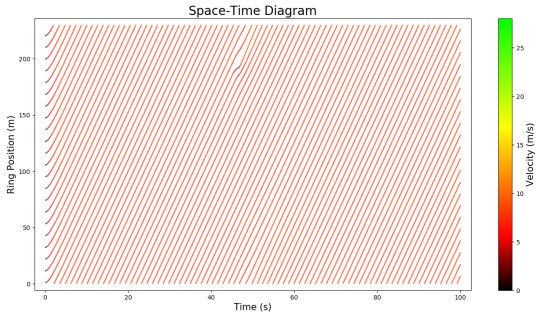

Fig. 6: BCM with a perturbation, showing an average speed of 7.9 m/s.

5) BCM from an nonuniform state (Figure 7): Again, 22 BCM vehicles are run in this simulation, but from the same nonuniform flow starting state as in the nonuniform flow OVM case, in which vehicles are spaced randomly subject to a bunching factor. There is some initial instability and small traffic shockwaves, but again the BCM vehicles recover from this non-stable state and return to uniform flow.

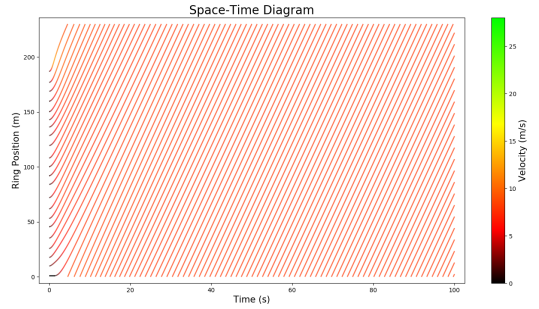

Fig. 7: BCM from a nonuniform state, showing an average speed of 7.9 m/s.

6) Mixed BCM/OVM from a nonuniform flow state (Figure 8): Here, 11 BCM vehicles and 11 OVM vehicles begin from a nonuniform flow, randomly spaced, and bunched starting state as described above. The proportion of bilateral control vehicles proves sufficient to prevent the stop-and-go waves seen in the unstable OVM scenario. Some velocity variation persists, however, unlike the full-BCM unstable scenario which returns to a completely uniform flow state.

7) Multi-lane experiment (Figure 9): A set of preliminary multi-lane experiments implementing the rudimentary lane-changing controller were also run, one of which is pictured below. In the experiment, all vehicles were initialized in the outer lane of a 200 meter two-lane road, and, as expected, a set of vehicles set to higher speeds changed into the inner lane, moving faster than cars in the inner lane.

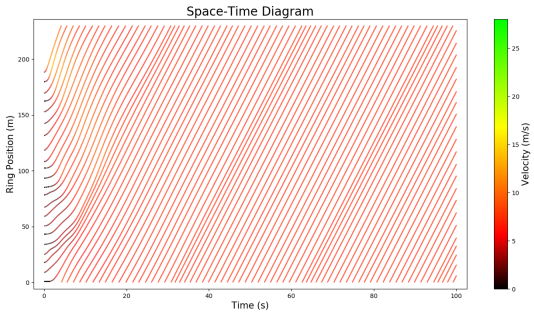

Fig. 8: BCM/OVM, nonuniform state, showing an average speed of 7.1 m/s.

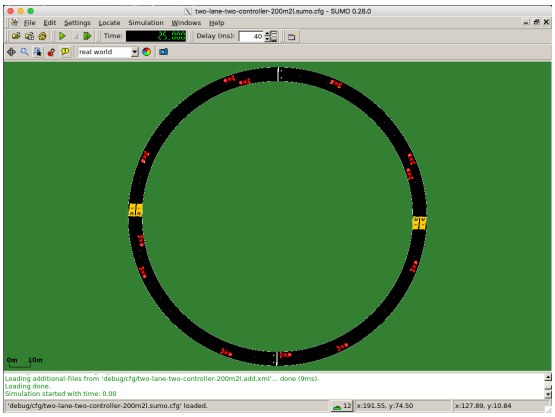

Fig. 9: A sample multi-lane experiment

C. Reinforcement learning experiments

CISTAR permits configuring and running reinforcement learning experiments on traffic problems through RLLab. RLLab enables the easy tuning of the baseline, policy parameterization, and numerous hyperparameters such as path length and batch size. In addition, CISTAR Scenarios allow easy formulation of different MDPs for studying different traffic problems, such as optimizing for different rewards like fuel consumption or matching a target speed. The following sample experiment use an RL algorithm called Trust Region Policy Optimization (TRPO) [23] due to its high performance on a number of reinforcement learning benchmarks [51], including in a traffic application [35].

1) Sample Experiment: target velocity: In this sample experiment, we demonstrate as a proof-of-concept a problem which can also be solved using linear control. All vehicles in the setup are controllable, and the goal is to control all vehicles to drive at a target velocity, from a given initial condition. The state of the MDP is represented by the current velocity of all vehicles in the system, and the action is represented by an instantaneous velocity that will be applied in the following timestep. The reward signal is the mean-squared error of the current velocity of all vehicles with the target velocity. The policy is a Gaussian multi-layer perceptron represented by a single-layer 16-neuron neural network, and the hyperparameters were tuned accordingly. For a simple

setup of four vehicles, we observe in Figure 10 that the reward signal converges to near zero in 600 iterations, indicating that the vehicles successfully learn to drive at a target velocity. We then tried a more congested track with 25 vehicles, but the space limitations prevent the policy from learning efficiently, as can be seen in Figure 11 and Figure 12. Training a policy efficiently on a congested environment is the subject of ongoing work.

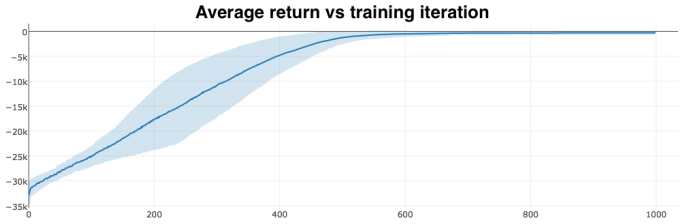

Fig. 10: The convergence curve for the target velocity reinforcement learning experiment indicates convergence to near-zero reward (indicating matched target velocity) in 600 training iterations.

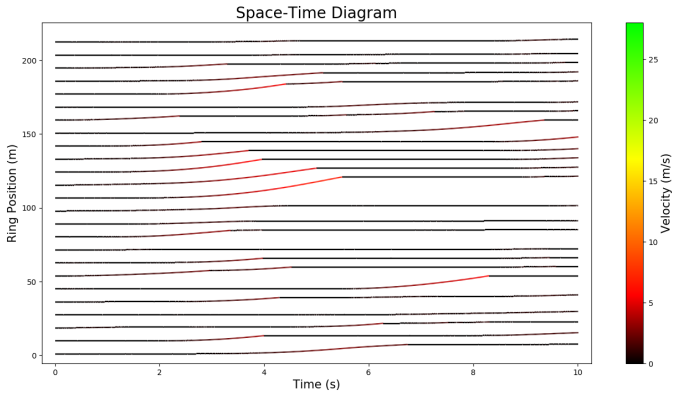

Fig. 11: Spacetime diagram for a very congested track. The learned policy can barely move cars since they are so close.

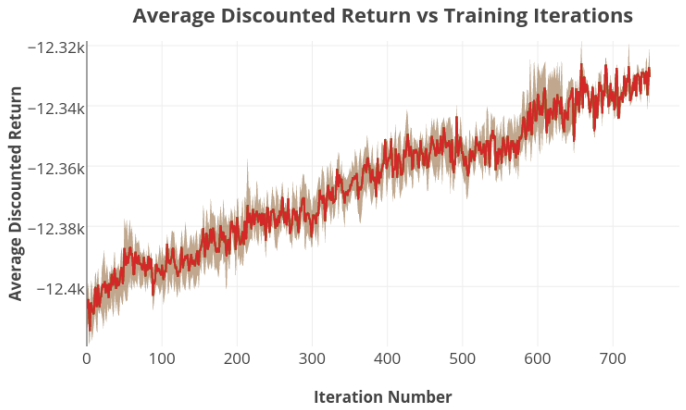

Fig. 12: The discounted reward for a congested shows improvement in a policy; however, the reward is still very far from 0.

VII. Conclusion

CISTAR is a tool that quickly and efficiently facilitates simulating complex traffic settings built on widely-used open source tools. Researchers can use it to exercise fine-grained control over vehicles in simulation and test a variety of different controllers against each other. RLLab integration enables the creation of environments in which teams of autonomous vehicles can learn and execute policies in various settings.

The specific use of CISTAR to achieve locally optimal (with respect to the policy parameterization) behaviors for teams of connected automated vehicles on a variety of traffic "tasks" is the subject of our ongoing research, including on ring roads, merges, intersections, etc. Additional ongoing work includes further development of CISTAR features, including more realistic lane-changing models, multi-lane safety, and preparation for an open-source release of CISTAR. Other future work include establishing a set of benchmark tasks (similar in spirit to [51]) for a wide variety of traffic scenarios and integration of CISTAR with OpenAI gym [28] for further compatibility with reinforcement learning tools.

Acknowledgments

The authors would like to thank Nathan Mandi for early design discussions, Rocky Duan and Alex Lee for RLLab support, Jakob Erdmann for SUMO support, Aboudy Kreidieh for contributing documentation, and Professor Alexander Skabardonis for several insightful discussions about vehicle dynamics and fail-safes.

References

[1] M. Treiber and A. Kesting, "Traffic flow dynamics," *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg*, 2013.

[2] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.

[3] P. A. Ioannou and C.-C. Chien, "Autonomous intelligent cruise control," *IEEE Trans. on Vehicular technology*, vol. 42, no. 4, pp. 657–672, 1993.

[4] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE transactions on intelligent transportation systems*, vol. 4, no. 3, pp. 143–153, 2003.

[5] I. t. s. Technical Committee ISO/TC 204, *Intelligent transport systems – Adaptive Cruise Control systems – Performance requirements and test procedures*, ISO ISO 15 622:2010, 2010.

[6] E. W. Martin, K. Boriboonsomsin, N. D. Chan, N. Williams, S. A. Shaheen, and M. Barth, "Dynamic ecodriving in northern california: A study of survey and vehicle operations data from an ecodriving feedback device," in *92nd Annual Meeting of the Transportation Research Board, Washington, DC, January*, 2013.

[7] C.-Y. Liang and P. Huei, "String stability analysis of adaptive cruise controlled vehicles," *JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing*, vol. 43, no. 3, pp. 671–677, 2000.

[8] A. Bose and P. A. Ioannou, "Analysis of traffic flow with mixed manual and semiautomated vehicles," *IEEE Trans. on Intelligent Transportation Systems*, vol. 4, no. 4, pp. 173–188, 2003.

[9] P. A. Ioannou and M. Stefanovic, "Evaluation of acc vehicles in mixed traffic: Lane change effects and sensitivity analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 79–89, 2005.

[10] M. A. S. Kamal, J.-i. Imura, T. Hayakawa, A. Ohata, and K. Aihara, "Smart driving of a vehicle using model predictive control for improving traffic flow," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 878–888, 2014.

[11] D. Swaroop, "String stability of interconnected systems: An application to platooning in automated highway systems," *California Partners for Advanced Transit and Highways (PATH)*, 1997.

[12] G. Orosz, R. E. Wilson, and G. Stépán, "Traffic jams: dynamics and control," *Philosophical Trans. of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4455–4479, 2010.

[13] G. Orosz, J. Moehlis, and F. Bullo, "Delayed car-following dynamics for human and robotic drivers," in *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2011, pp. 529–538.

[14] I. G. Jin and G. Orosz, "Dynamics of connected vehicle systems with delayed acceleration feedback," *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 46–64, 2014.

[15] B. K. Horn, "Suppressing traffic flow instabilities," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 13–20.

[16] L. Wang, B. K. Horn, and G. Strang, "Eigenvalue and eigenvector analysis of stability for a line of traffic," *Studies in Applied Mathematics*, 2016.

[17] C. Wu, A. Bayen, and A. Mehta, "Stabilizing traffic with autonomous vehicles," in *Submission*, 2017.

[18] M. Brackstone and M. McDonald, "Car-following: a historical review," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 2, no. 4, pp. 181–196, 1999.

[19] Z. Zheng, "Recent developments and research needs in modeling lane changing," *Transportation research part B: methodological*, vol. 60, pp. 16–32, 2014.

[20] A. Kotsialos, M. Papageorgiou, and A. Messmer, "Optimal coordinated and integrated motorway network traffic control," in *14th International Symposium on Transportation and Traffic Theory*, 1999.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[22] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[23] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015, pp. 1889–1897.

[24] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa, "Learning continuous control policies by stochastic value gradients," in *Advances in Neural Information Processing Systems*, 2015, pp. 2944–2952.

[25] M. Lai, "Giraffe: Using deep reinforcement learning to play chess," *arXiv preprint arXiv:1509.01549*, 2015.

[26] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.(JAIR)*, vol. 47, pp. 253–279, 2013.

[27] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik *et al.*, "Deepmind lab," *arXiv preprint arXiv:1612.03801*, 2016.

[28] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[29] G. Synnaeve, N. Nardelli, A. Auvolat, S. Chintala, T. Lacroix, Z. Lin, F. Richoux, and N. Usunier, "Torchcraft: a library for machine learning research on real-time strategy games," *arXiv preprint arXiv:1611.00625*, 2016.

[30] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5026–5033.

[31] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," *Software available at http://torcs. sourceforge. net*, 2000.

[32] O. Vinyals, "Deepmind and blizzard to release starcraft ii as an ai research environment," https://deepmind.com/blog/deepmind-and-blizzard-release-starcraft-ii-ai-research-environment/, 2016.

[33] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[34] N. Polson and V. Sokolov, "Deep learning predictors for traffic flows," *arXiv preprint arXiv:1604.04527*, 2016.

[35] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, "Expert level control of ramp metering based on multi-task deep reinforcement learning," *CoRR*, vol. abs/1701.08832, 2017. [Online]. Available: http://arxiv.org/abs/1701.08832

[36] M. G. Karlaftis and E. I. Vlahogianni, "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 3, pp. 387–399, 2011.

[37] G. D. Cameron and G. I. Duncan, "Paramicsparallel microscopic simulation of road traffic," *The Journal of Supercomputing*, vol. 10, no. 1, pp. 25–53, 1996.

[38] M. Fellendorf, "Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority," in *64th Institute of Transportation Engineers Annual Meeting*. Springer, 1994, pp. 1–9.

[39] M. Fellendorf and P. Vortisch, "Microscopic traffic flow simulator vissim," in *Fundamentals of traffic simulation*. Springer, 2010, pp. 63–93.

[40] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, "Traffic simulation with aimsun," in *Fundamentals of traffic simulation*. Springer, 2010, pp. 173–232.

[41] K. N. Horni, A. and K. A. (eds.), *The Multi-Agent Transport Simulation MATSim*. Ubiquity, London, 2016.

[42] 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, 2012.

[43] (2016) Car-following-models. [Online]. Available: http://sumo.dlr.de/wiki/Car-Following-Models#tau

[44] (2016) Simulation/basic definition. [Online]. Available: http://sumo.dlr.de/wiki/Simulation/Basic_Definition#Defining_the_Time_Step_Length

[45] R. Bellman, "A markovian decision process," DTIC Document, Tech. Rep., 1957.

[46] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[47] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, "Structure stability of congestion in traffic dynamics," *Japan Journal of Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 203–223, 1994.

[48] ——, "Dynamical model of traffic congestion and numerical simulation," *Physical review E*, vol. 51, no. 2, p. 1035, 1995.

[49] R. A. Howard, "Dynamic programming and markov processes," 1960.

[50] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *NIPS*, vol. 99, 1999, pp. 1057–1063.

[51] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," *CoRR*, vol. abs/1604.06778, 2016. [Online]. Available: http://arxiv.org/abs/1604.06778

[52] R. Dowling, A. Skabardonis, and V. Alexiadis, "Traffic analysis toolbox volume iii: guidelines for applying traffic microsimulation modeling software," Tech. Rep., 2004.

[53] H. Yeo, A. Skabardonis, J. Halkias, J. Colyar, and V. Alexiadis, "Oversaturated freeway flow algorithm for use in next generation simulation," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2088, pp. 68–79, 2008.

[54] M. Batista and E. Twrdy, "Optimal velocity functions for car-following models," *Journal of Zhejiang University-SCIENCE A*, vol. 11, no. 7, pp. 520–529, 2010.

[55] J. Erdmann, "SUMO's lane-changing model," in *Modeling Mobility with Open Data*. Springer, 2015, pp. 105–123.

[56] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki, and S. Yukawa, "Traffic jams without bottlenecks–experimental evidence for the physical mechanism of the formation of a jam," *New Journal of Physics*, vol. 10, no. 3, p. 033001, 2008.