From Sim to Real: A Pipeline for Training and Deploying Traffic Smoothing Cruise Controllers

Nathan Lichtlé*, Eugene Vinitsky*, Matthew Nice*, Rahul Bhadani, Matthew Bunting, Fangyu Wu, Benedetto Piccoli, Benjamin Seibold, Daniel B. Work, Jonathan W. Lee, Jonathan Sprinkle, Alexandre M. Bayen

***Abstract*—Designing and validating controllers for connected and automated vehicles to enhance traffic flow presents significant challenges, from the complexity of replicating real-world stop-and-go traffic dynamics in simulation, to the intricacies involved in transitioning from simulation to actual deployment. In this work, we present a full pipeline from data collection to controller deployment. Specifically, we collect 772 kilometers of driving data from the I-24 in Tennessee, and use it to build a one-lane simulator, placing simulated vehicles behind real-world trajectories. Using policy-gradient methods with an asymmetric critic, we improve fuel efficiency by over 10% when simulating congested scenarios. Our comprehensive approach includes reinforcement learning for controller training, software verification, hardware validation and setup, and navigating various sim-to-real challenges. Furthermore, we analyze the controller's behavior and wave-smoothing properties, and deploy it on 4 Toyota Rav4's in a real-world validation experiment on the I-24. Finally, we release the driving dataset [1], the simulator and the trained controller [2], to enable future benchmarking and controller design.**

***Index Terms*—Intelligent Transportation Systems, Autonomous Vehicle Navigation, Energy and Environment-Aware Automation, Reinforcement Learning.**

Manuscript created October, 2023. This work was supported by the National Science Foundation, the Savio cluster at UC Berkeley and the U.S. Department of Energy (more details in Sec. VIII).

(Corresponding author: Nathan Lichtlé.)

Nathan Lichtlé, Fangyu Wu, Jonathan W. Lee and Alexandre M. Bayen are with the Department of Electrical Engineering and Computer Sciences (EECS), University of California, Berkeley, CA 94720-1770, USA (email: nathan.lichtle@gmail.com; fangyuwu@berkeley.edu; jonny5@berkeley.edu; bayen@berkeley.edu).

Nathan Lichtlé is also with the Centre d'Enseignement et de Rercherche en Mathématiques et Calcul Scientifique (CERMICS), École des Ponts ParisTech, Champs-sur-Marne, 77455 Marne-la-Vallée, France.

Eugene Vinitsky is with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720, USA (email: vinitsky.eugene@gmail.com).

Matthew Nice is with the Department of Civil and Environmental Engineering, Vanderbilt University, Nashville, TN 37235, USA (email: matthew.nice@vanderbilt.edu).

Rahul Bhadani is with the Department of Electrical and Computer Engineering, The University of Alabama in Huntsville, Huntsville, AL 35899, USA (email: rahul.bhadani@uah.edu).

Matthew Bunting and Daniel B. Work are with the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37235, USA (email: matthew.r.bunting@vanderbilt.edu; dan.work@vanderbilt.edu).

Benedetto Piccoli is with the Department of Mathematical Sciences and Center for Computational and Integrative Biology, Rutgers University, Camden, NJ 08102, USA (email: piccoli@camden.rutgers.edu).

Benjamin Seibold is with the Departments of Mathematics and Physics, Temple University, Philadelphia, PA 19122, USA (email: benjamin.seibold@temple.edu).

Daniel B. Work and Jonathan Sprinkle are with the Department of Computer Science, Vanderbilt University, Nashville, TN 37235, USA (email: jonathan.sprinkle@vanderbilt.edu).

* **These authors contributed equally to this work.**

I. Introduction

THE increased availability of automated lane and distance-keeping in modern vehicles has rapidly transitioned our roadways into the mixed autonomy regime, i.e. automated and human drivers all operating together. With the availability of connected and automated vehicles (CAVs) as mobile traffic actuators, it is now possible to perform Lagrangian traffic control in which control of the highway is dispersed amongst many vehicles in the flow. The ability to perform distributed control using automated vehicles as actuators has brought closer a long-standing goal of research in that field [3]–[7]: to use the programmability and fast reaction time of CAVs to improve highway metrics like congestion and energy efficiency for the entirety of the highway traffic.

In this work, we focus on designing controllers that are able to smooth waves using only local information that would be accessible via in-vehicle sensors. Note that this precludes us from smoothing low-frequency waves that are distributed over a wide spatial distance; these waves are likely not observable via individual vehicle sensors alone and would require the inclusion of downstream information from loop sensors or cameras. We first leverage data we collected by driving on the highway to create a simulation replaying real vehicle trajectories. Using Proximal Policy Optimization [8], an RL policy gradient algorithm, we learn a controller that decreases the fuel consumption of the platoon in simulation by 16% for the CAV and 10% on average for the platoon vehicles. We verify the controller's behavior in a series of tests in Gazebo, a 3D robotics simulator, then embed our controller into the hardware stack, enabling it to read information from the vehicle's sensors and to send acceleration commands back. Finally, we deploy the controller on four real vehicles in highway traffic, showing that the resultant controllers exhibit behaviors that are robust to potential gaps between field deployment and our simplified simulator. An overview of the full pipeline can be seen in Fig. 1.

Prior work [9] has shown that even at current low penetration rates of less than 4%, empirical and theoretical evidence suggests that CAVs can significantly reduce stop-and-go traffic, a pernicious transitory phenomenon in which vehicles alternate between starting and stopping, consuming extra fuel in the process. However, prior approaches have a unifying limitation: they are developed and analyzed in simplistic settings such as vehicles traveling around a closed ring or hand-designed input perturbations. Testing on more complex settings is difficult as: 1) real-world highway sensor

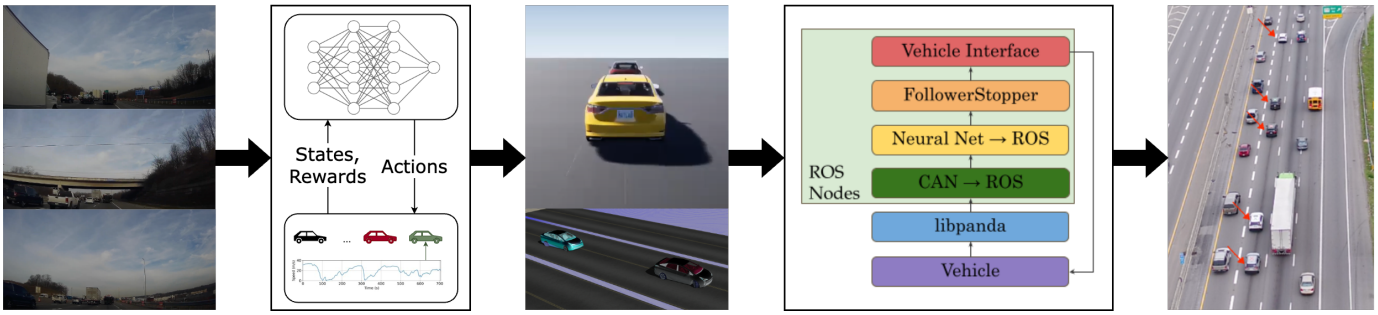

Fig. 1. Our pipeline for designing and validating automated-vehicle traffic-smoothing controllers. From left to right: collection of trajectory data by driving on the highway; RL training in a simulation replaying the highway trajectories; software verification of the controller in Gazebo; integration of the trained neural network into the hardware stack; validation experiment with 4 vehicles running the controller in highway traffic, and 7 human-driven vehicles for metrics collection.

data is sparse and lacks the required resolution and detail needed for accurate modeling; 2) developing simulators that properly reproduce emergent traffic phenomena from many-vehicle-interactions is challenging.

Even leaving aside the software engineering challenge of designing large, calibrated micro-simulations, building complex models of a highway is heavily data-constrained. Loop detectors only yield macroscopic statistics such as the number of vehicles crossing them and their speed, while cameras tend to cover only a small portion of the roadway. This lack of available data is a fundamental issue as the trajectories of vehicles traveling through waves depend on the wave speed [10], and yet the wave speed and constituent frequencies are difficult to estimate with available stationary sensors. However, without an accurate means of reconstructing the stop-and-go traffic that is likely to occur on a particular highway, it is difficult to validate how a controller will perform when deployed on that highway. Consequently, it is unclear whether progress on control design for real-world smoothing is being made.

The main contribution of this work is the design and demonstration of a safe pipeline for training and deploying efficient wave-smoothing longitudinal controllers for CAVs. Our pipeline avoids the aforementioned modeling challenges by leveraging field-deployment data to accurately reproduce the distribution of waves observed on the I-24 in Tennessee, on which we intend to deploy CAVs. Instead of attempting to build or tune a large high-fidelity simulation, we evaluate and train our controllers on this collected highway trajectory data. We construct our simulation in a way that enables a realistic representation and evolution of waves from the particular highway on which we intend to deploy CAVs, and train controllers to smooth these waves using state-of-the-art RL methods. We design our controllers to be deployable on real-world cars with built-in hardware sensors and actuation, and optimize them over several criteria including energy efficiency, safety, smoothness and adherence to human norms. Controllers then undergo software verification, after which they are exported and wrapped for deployment on a specific hardware stack. Once deployed onto the vehicle and tested in simple scenarios, the controllers are ready to safely drive on the highway, under human supervision. While all the blocks in our pipeline work together, they can be switched with other components. For instance, training could be done using different data or simulations, and different car models might require a different hardware stack.

The rest of this article is organized as follows: in Section II, we discuss the related work. In Section III, we discuss the data collection process and how it was used to build simulations that replicate realistic traffic dynamics. In Section IV, we describe the controller design process, by which we trained safe and efficient flow-smoothing controllers using our data-based simulation. In Section V, we describe the deployment pipeline that safely and robustly ports our controllers trained in simulation onto the physical vehicle's hardware via a battery of software and hardware tests. In Section VI, we discuss the performance and behavior of the controllers in simulation, as well as results from the field tests we conducted on the highway. Finally, Section VII concludes this work and provides practical considerations to be considered in future work.

II. Related Work

Prior work has investigated the efficacy of traffic smoothing controllers on settings such as rings or hand-designed input perturbations. The most closely related works are [9], [11]–[13]. In [9], the authors showed that a single CAV could be used to dampen stop-and-go waves on a ring with 21 human drivers, yielding sharply improved fuel efficiency. The work in [11] studies traffic smoothing with connected CAVs and demonstrates that the connectivity can be used for more effective dampening of waves on a single-lane, eight-mile-long public road. The work in [12] conducted an experiment in which three control vehicles were lined up across three highway lanes and their preferred speed is selected by an external centralized controller with the goal of smoothing traffic flow. Finally, [13] uses a similar approach as this work, using the highD dataset [14] to generate realistic waves that are then dampened by a following RL-controlled vehicle, while [15] designs CAV smoothing controllers using microscopic models calibrated from NGSIM trajectory data [16]. The primary distinction between this work and [13] is the use of a calibrated energy model and the physical deployment of our system onto the roadway.

Other works have considered the wave-dampening properties of existing commercially-available cruise controllers, with [17]–[19] all observing that the vehicles they tested were string unstable (see [20] for a definition of string instability). Finally, [21] has studied some of the sim-to-real challenges in deploying RL-learned cruise controllers into more realistic settings. Prior work has also studied the use of reinforcement learning (RL) and optimal control for developing micro-level controllers that optimize mixed autonomy traffic. [22] learns memory-based policies that infer ring densities and consequently outputs near-optimal policies for the ring, [23] uses multi-agent reinforcement learning (MARL) to optimize the throughput of a merging region, and [24] employs MARL to investigate the potential impacts of altruistic automated driving on a merge scenario. At a network level, RL has been used to learn routing behaviors for CAVs that induce human drivers to select paths that lead to decreased congestion [25].

This work is based upon preliminary work [2], which primarily introduces the data collection process, releases the associated dataset, and provides an overview of how the RL controller is trained as well as initial simulation and deployment results. In this work, a detailed pipeline for the design, development and deployment of RL-based controllers using real trajectory data is introduced. This encompasses every step, from data acquisition, RL setup and training (which were previously introduced in [2]) to all the stages of deployment onto vehicles, including software verification, hardware validation and hardware setup. Special attention is given to the inclusion of a safety controller and the development of a hardware stack for controller-to-vehicle communication. Essential adjustments are detailed, like aligning the acceleration command with the vehicle's hardware capabilities to match the desired acceleration command with the realized one, or to adapt to limitations of the vehicle's sensors. Finally, the controller's behavior is analyzed more in depth, with an emphasis on understanding how it is able to smooth traffic.

III. Data-Based Simulation

This section details the process by which we build a data-based simulator from highway trajectory data we collected. This simulator serves as the basis for training and evaluating our wave-smoothing controllers: a simulated vehicle replays the real-world trajectory in simulation, and a simulated CAV drives behind it and learns to smooth the perturbations that the trajectory exhibits, such as stop-and-go waves.

A. Data Collection and Processing

We collected a driving dataset by recording trajectory data on a 14.5-kilometer-long segment (displayed in Fig. 2) of the I-24 located southeast of Nashville, Tennessee. Each drive is conducted in an instrumented vehicle that logs Controller Area Network (CAN) and GPS data (time, longitude and latitude) via libpanda [26]. Collected CAN data includes measurements from the vehicle's sensors such as the velocity and instantaneous acceleration of the *ego vehicle* (the vehicle being driven), the relative velocity of the *lead vehicle* (the vehicle in front of the ego vehicle), and the *space-gap* (bumper-to-bumper distance).

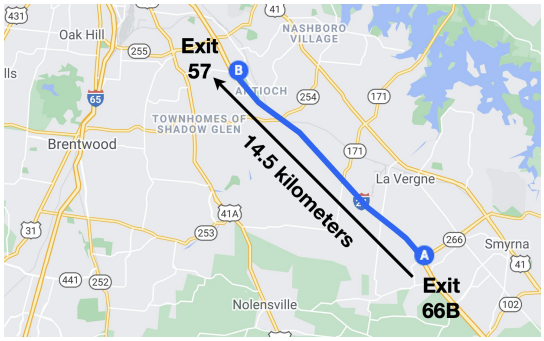

Fig. 2. Portion of the I-24 highway on which we collected most of the dataset described in Sec. III-A, and where we ran the experiments described in Sec. VI.

The raw data from each drive was stored in two files: a CAN data file and a GPS file. The pertinent data was extracted from the CAN data and decoded into readable CSV format using Strym [27], combined with the GPS data. In order to synchronize the CAN data with the GPS time, which is recorded at 10 Hz, high-frequency CAN data was down-sampled and then linearly interpolated, while low-frequency data directly underwent linear interpolation. Additionally, the distance traveled and direction of travel were calculated using GPS positional data. The processed dataset, used to train the algorithm in this work, is made publicly available at [1].

The drives are varied in the time of day, day of the week, direction of travel on the highway, and level of congestion. Each drive is made up of one or more passes through the highway stretch of interest. As such, the data is collected over a wide range of traffic conditions ranging from congested traffic to free-flow traffic, including many successive acceleration and deceleration patterns that characterize stop-and-go waves. Fig. 3 shows an example trajectory from the dataset where we can observe such patterns.

Our main goal is to smooth higher-frequency waves, which typically happen in the lower range of speeds. However, the distribution of speeds in the training dataset, shown in Fig. 4, is skewed towards higher speeds. While we could filter the dataset to only contain low speeds, likely making the learning problem simpler, Fig. 3 suggests that regions of congestion are often quickly followed by regions of high speed. To ensure our controller behaves appropriately at high speeds and in transitions between high and low-speed regions, we keep both low and high velocities in the training dataset. However, we observe that the westbound data contains a lot more stop-and-go traffic than the collected eastbound data, which is mostly free-flow traffic. Thus, we focus on westbound data in this work , which contains 60 trajectories (also including free-flow traffic), representing 8.8 hours and 772.3 kilometers of driving. We observe that this is more than enough data to generalize and prevent overfitting, especially given that we use a very small and local observation space and small neural networks.

Finally, we note that our collected dataset contains both the trajectory of our drivers and the vehicles in front of them (via

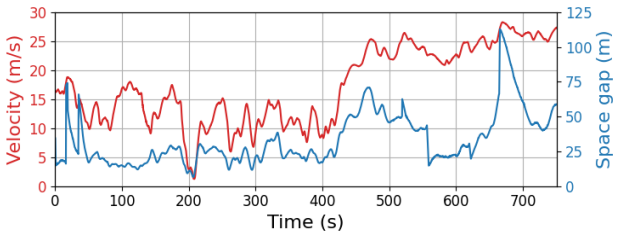

Fig. 3. Velocity of the ego vehicle (blue) and space-gap to the lead vehicle (red) for a single trajectory in the dataset, containing sharp variations in both velocity and space-gap.

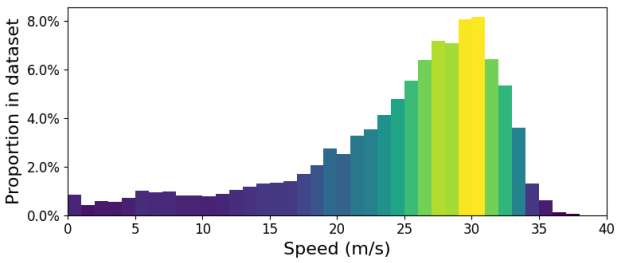

Fig. 4. Histogram showing the distribution of velocities of the ego vehicle in the dataset.

Fig. 5. Vehicle formation used in simulation. A trajectory leader (in green) driving a speed profile drawn from the dataset is placed in front of a CAV (in red) which is followed by a platoon of 5 human vehicles (in black), modeled using the Intelligent Driver Model.

space-gap and relative velocity data logged on the CAN). We discard the lead trajectories and do not use them for simulation for two reasons. First, the lead trajectories contain both cut-ins (a vehicle cuts in between the lead vehicle and the ego driver) and cut-outs (the lead vehicle changes lanes). While cut-outs are likely unaffected by the behavior of the ego driver, cut-ins are likely a function of the spacing between the ego driver and the lead vehicle. Since our trained controller will have different space-gap-keeping patterns, it is possible that the observed cut-outs would not occur given the controller's choice of space-gaps. Secondly, since the lane changes cause sudden variations in the spacing to the leader vehicle, it is possible for the leader vehicle to wind up behind the CAV if it keeps a closer gap to the leader than the vehicle that collected the data.

B. Constructing the Training Environment

In order to use the collected data to learn traffic-smoothing controllers, we build a fast single-lane simulator in which the real-world trajectory is replayed through a single vehicle placed at the front of a simulated platoon. The platoon consists of a mix of CAVs and *human* vehicles, described in Sec. III-C. During training, a CAV is typically driving right behind the dataset trajectory and is followed by from 5 to 25 human vehicles; this scenario is depicted in Fig. 5. The CAV's goal is to learn to smooth out the waves and perturbations introduced from the real-world trajectory, in a way that also stabilizes the platoon of human vehicles behind it.

Although having a full micro-simulation of the I-24 would allow for training on a model with complex long-range interactions between the vehicles, the simulator proposed here allows us to train on realistic driving dynamics that are representative of both the types of waves on this highway and how drivers react to wave formation. As an additional benefit, this single-lane simulation is significantly faster than a comparable micro-simulation of the full 14-kilometer road section with thousands of vehicles, which has a high computational cost.

C. Human Driver Model

We use the standard IDM model [28] to model human vehicles in our simulation. Indicating by x_i, v_i, the position and velocity of the i-th vehicle, such that $i-1$ is the vehicle directly in front of vehicle i, we have (for IDM-controlled vehicles) $\dot{x}_i = v_i$ and

$$\dot{v}_i = a\left(1 - \left(\frac{|v_i|}{v_f}\right)^{\delta} - \left(\frac{s_0 + v_i\tau + \frac{v_i(v_i - v_{i-1})}{2\sqrt{ab}}}{x_{i-1} - x_i - \ell}\right)^2\right) \quad (1)$$

where the parameters are as follows: a is the maximum vehicle acceleration, b the comfortable braking deceleration, v_f the desired free-flow velocity, τ the desired time gap, s_0 the minimum desired space gap, ℓ the vehicle length, and δ the acceleration exponent.

We consider the following parameters in our work: $a = 1.3\frac{\text{m}}{\text{s}^2}$, $b = 2.0\frac{\text{m}}{\text{s}^2}$, $v_f = 45\frac{\text{m}}{\text{s}}$, $\delta = 4$, $s_0 = 2\text{m}$ and $\ell = 5\text{m}$. Moreover, we add a small Gaussian noise to the IDM acceleration output $\dot{v}_i$, modeled as a centered normal distribution $\mathcal{N}(0, \sigma^2)$ with standard deviation $\sigma = 0.1$ (which accounts for the fixed discretization timestep $dt = 0.1\text{s}$).

There are some important considerations that we made regarding the choice of the IDM parameters. Previous works observed that existing commercially available cruise controllers are string-unstable [17], [19], meaning that they will amplify traffic waves rather than smooth them. Human driving is also generally string-unstable, as stop-and-go wave patterns are ubiquitous in highway traffic, especially above a certain network capacity. As such, we want to replicate this behavior of growth and propagation of stop-and-go waves in our simulation. Indeed, if we used a string-stable car-following model, the waves would naturally dissipate and there would be nothing left for our CAVs to learn to smooth.

In order to test for the string-instability of a particular set of IDM parameters, we employed a method similar to [19], placing vehicles on a ring road and doing a numerical analysis of the propagation of perturbations along the string of vehicles, analyzing whether they are growing or vanishing. In particular, the model was tested to be string-unstable below $18\frac{\text{m}}{\text{s}}$ (as can be observed in Fig. 13, left), where this specific value is chosen empirically as a boundary between the congested regime, in which stop-and-go waves typically happen, and the free-flow regime, in which there is not much for the CAVs to smooth except for small local perturbations. We thus chose

the a and b parameters to ensure the instability of the model in the congested region, while staying close to the original model parameters [28]. These two parameters can typically make or break the string-stability, as they control the sharpness of accelerations. The maximum speed parameter v_f, usually set to the speed limit, is empirically chosen to be around the largest speeds in our dataset, and the values for s_0 and δ are left to their typical value in the literature [28].

IV. Controller Design

In this section, we describe the process by which we design flow-smoothing energy-improving controllers using our data-based simulation. We first design the controller's inputs and outputs while keeping deployability in mind, then an objective function that balances energy efficiency, safety, smoothness and adherence to social driving norms. We then introduce the training algorithm we use, with slight modifications for better training stability in our partially-observable context. Finally, we introduce some details about the training experiments.

A. Partially-Observable Markov Decision Process

Due to the ability to acquire information about the state solely through CAN data (and optionally GPS), we model our problem as a Partially-Observable Markov Decision Process (POMDP) [29]. Below we describe the state and action spaces that are feasible to implement given our available sensing and actuation capabilities, as well as the reward function we aim to maximize.

1) State Space: The observations are $[v, v_{\text{lead}}, h]$ where v is the CAV speed, v_{lead} the speed of the vehicle right in front of it, and h the space-gap. All of these features can be acquired by using the stock forward-facing radar and the data collection software [26], [30] that we place on our vehicles. Note that this state is partially observed; the vehicle is not provided with information such as the state of vehicles in the platoon behind it nor information needed to predict the evolution of its leader vehicle.

2) Action Space: The policy's output is an instantaneous acceleration a, bounded between $[-3.0, 1.5]\,\frac{\text{m}}{\text{s}^2}$. It may be further clipped to respect the speed limit and is then applied to the CAV. Note that we do not allow the CAVs to lane-change in this work to minimize safety considerations.

3) Reward Function: The reward the CAV receives at time-step t is a combination of minimizing energy consumption, acceleration regularization, and penalties for leaving too small or too large gaps. It is given by

$$r_t = 1 - c_0 E_t - c_1 a_t^2 - c_2 P_t \ . \tag{2}$$

Here E_t is the instantaneous gallons of fuel consumed by the CAV (given by a piece-wise polynomial energy model calibrated to a Toyota Rav4; the fitting procedure and function coefficients are given in [31]), a_t the CAV's instantaneous acceleration in $\frac{m}{s^2}$ and P_t its gap penalty, all at time-step t. The first term is intended to discourage fuel consumption, the second to encourage smooth driving, and the third to discourage the formation of large gaps that induce cut-ins or small gaps that might be unsafe. P_t is essential as the energy-minimizing solution is to come to a full stop; P_t both removes this solution and is used to encourage the vehicle to drive with a "sensible" distance to its lead vehicle.

For our reward functions, we use coefficients $c_0 = 1.0\frac{\text{hour}}{\text{gallon}}$, $c_1 = 0.002\frac{s^2}{m}$ and $c_2 = 2$, and penalize with $P_t = 1$ when the gap is below 7m, above 120m or when the time-gap (i.e., space-gap over speed) to the leader is below 1 second. These particular values were selected via an informal hyperparameter search and found to yield improved fuel consumption of the platoon while maintaining both plausible roadway behaviors (via not opening too large a gap) and driver comfort (via not getting too close to the leader.

Note that our reward function does not directly optimize the stated objective of optimizing miles per gallon (MPG). Unfortunately, mile-per-gallon is a quantity that can only be calculated at the completion of a trajectory and is therefore challenging to use as a reward function since RL methods struggle with infrequent rewards [32]. However, we note that for a fixed-length trajectory, subject to the constraint that the vehicle must complete the entire trajectory before the episode terminates, minimization of consumed fuel and MPG maximization are equivalent. This is simply because, in the fixed-length trajectory, the numerator in the MPG term becomes a constant.

B. Asymmetric Actor-Critic Algorithm

We train our policy using Proximal Policy Optimization [8] (PPO), a policy gradient algorithm. We modify the standard PPO algorithm by providing the value function with a few additional inputs: the total distance traveled from start to time t, the total energy consumed by the agent at time t, and time t. The value function V^π estimates the reward-to-go from a given state s_t and a particularly controller π as

$$V^\pi(s_t) = \mathbb{E}\left[\sum_{j=t}^{T} \gamma^{(j-t)} r(s_j, a_j) \mid s_j\right]. \tag{3}$$

This quantity is difficult to estimate without the additional information we provide due to the partially observed state described in Sec. IV-A. The non-local information provided to the value function is used exclusively during training for variance reduction (see [8] for details on the usage of the value function), and these additional inputs are neither available nor needed by the controller during evaluation.

C. Training Details

Training was done using the PPO implementation provided in Stable Baselines 3 [33] version 1.0, a Pytorch-based deep RL library. Training details, a script to reproduce the results in the article, and hyperparameters are provided in the linked code-base.

Each episode is run with a fixed horizon of 1000 steps. For a leader trajectory of length M, we sample the start-point of a trajectory uniformly from the first $M-1000$ steps. Termination of the trajectory occurs when the fixed horizon is reached.

We note that there is an incongruity between the finite-horizon objective and our infinite-horizon field deployment. The shorter horizons are used here as gradient-based controller design methods (in particular the ones used here) can struggle with long horizons due to a linear dependency of the variance of the gradient estimator on the horizon [34]. It is possible that optimizing the gallons consumed on the finite horizon generates behavior that is sub-optimal for a longer horizon.

Finally, the dynamics of all vehicles in the system are double integrators updated with a ballistic update [35] and a 0.1 second time-step with the exception of the lead vehicle whose position is directly updated from the pre-recorded data. The vehicles following behind the RL vehicle are updated using a Gaussian-perturbed IDM model described in Sec. III-C.

V. Deployment Pipeline

Once a trained controller performs well in simulation, several modifications and validation steps are still needed before deploying it on the hardware. Firstly, the control is wrapped in a few layers for better out-of-training-distribution behavior and for increased safety, and the acceleration output is converted to a desired velocity that accounts for the specific vehicle's dynamics. After that, several tests are performed in a robotics software that simulates the vehicle's dynamics and stress tests the controller in different driving scenarios. Once the controller passes all the checks in simulation, it is finally deployed onto the vehicles and a battery of hardware tests are performed before allowing the controller to drive in real traffic conditions. The following subsections go over each of the aforementioned steps in more details.

A. Post-Training Controller Modifications

We wrap our controller with a few modifications to handle out-of-distribution behavior between the simulator and the field deployment test. In particular, there are several significant changes in the distribution of observed states caused by the presence of lane changes in the field-deployment test that require careful handling.

First, there are challenges related to headways (gaps to the leader, also called space gaps) that exceed the values observed in the simulator, and are consequently out-of-distribution (OOD). As discussed in Sec. IV-A, we penalize the agent if it gets more than 120 meters from the lead vehicle. As a consequence, the CAV learns to successfully stay less than 120 meters away in all the trajectories evaluated in the later stages of training. It is possible (and we observe it to be the case) that for headways significantly above 120 meters, the controller has unexpected and undesirable behavior, usually of opening too large gaps. That portion of the state space ceases to be explored during training, and so appropriate behavior in this region of state space is gradually forgotten. In particular, at high speeds and above 150 meters, the controller begins to slowly decelerate.

In the field-deployment, these large headways can occur via two distinct mechanisms. First, if the CAV experiences a close sequence of cut-outs of the lead vehicle during field deployment, the headway could rapidly increase to large values. Additionally, the radar can occasionally not detect a lead vehicle. This occurs when either a lead vehicle is more than 250 meters away i.e. outside the range of the radar, or when the vehicle is taking a sharp curve and the lead vehicle ceases to be visible. In both of these cases, the radar returns a distance of 250 meters when the lead vehicle is missing which puts us into the range of the OOD behavior of the controller. We handle this case and ensure that the gap above 120 meters is rapidly closed by smoothly interpolating between the acceleration output by the controller and an acceleration of $0.75\frac{\text{m}}{\text{s}^2}$ using a logistic function. We note that it would be difficult for the neural network itself to smoothly handle sharply losing the leader vehicle. Indeed, some history of past states would be required for the controller to keep some memory of the surrounding traffic speed, which we leave for future work as a controller with more inputs would be more prone to overfitting, and the higher-dimensional input space would be harder to verify.

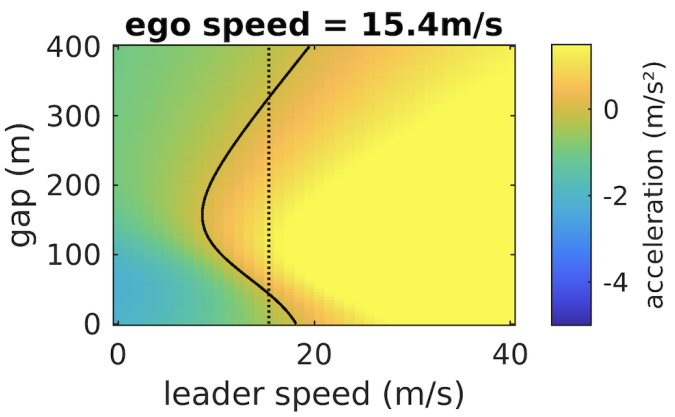

Fig. 6. Acceleration behavior of the RL controller (ego) as a function of the leader speed and gap to the leader for a fixed speed of the RL controller. The set of equilibrium values is given by the intersection of the thick black curve and the dotted curve. Due to out-of-distribution behavior, there are two unique equilibria, one of which occurs at an undesirably large gap.

However, for very large gaps, continually accelerating at $0.75\frac{\text{m}}{\text{s}^2}$ can lead to unreasonable speeds. For this reason, we cap the speed at $35\frac{\text{m}}{\text{s}}$ and apply an additional safety filter that smoothly interpolates between the desired acceleration down to an acceleration of -3.0 if the vehicle Time-To-Collision (time to close the headway to a stopped leader) falls below 6.5s.

After applying all these changes, the resultant controller is the following:

$$\begin{aligned} c_t &= \frac{1}{1+\exp\left(-\frac{h_t-120}{10}\right)} \\ a_t &= (1-c_t)\times\pi\left(s_t\right)+c_t\times 0.75 \end{aligned} \tag{4}$$

where $\pi(s_t)$ is the controller. Soft logistics are used rather than explicit if-else statements to account for the type of operands that are supported by the ONNX format we use to deploy controllers onto the vehicles (which we convert using `torch.onnx.export`).

In summary, the modifications described here are intended to keep the controller within the set of states that are in-distribution and override undesirable out-of-distribution behavior.

B. Converting Acceleration to Velocity

Here, we briefly describe the procedure used to convert our acceleration-based controller into a velocity-based controller. We refer to this as the "A-to-V trick". This step is necessary, as the action space of the RL-controller outputs an acceleration (see "Action space" in Sec. IV-A), but the vehicle control stack requires a velocity-based controller. We set the desired control speed as $v_{\text{des}}(t) = v(t) + Ta(t)$, $v(t)$ is the current vehicle speed, T is a constant and $a(t)$ is the output from the action space of our controller. The constant T must be tuned according to the specific vehicle's dynamics to account for the delay between commanding a certain velocity and the vehicle actually reaching this desired velocity.

In order to ensure that the commanded acceleration closely matches our desired acceleration, we fit a model of the vehicle's transfer function (see Appendix Sec. A for the system identification that led to our transfer function) to the first-order model in the s-domain $\Psi(s) = \frac{1}{1+\tau s}$. In state-space form this corresponds to the relaxation ODE $\dot{v}(t) = \frac{1}{\tau}\left(v_{\text{des}}(t) - v(t)\right)$. Using the relationship $v_{\text{des}}(t) = v(t) + Ta(t)$ and some minor simplifications, we arrive at $\dot{v}(t) = \frac{T}{\tau}a$: selecting $T = \tau$ gives us the desired property of our output acceleration matching out desired acceleration. We fit to the original transfer function and observe that $\tau = 0.6$ is the closest fit, so our final function is $v_{\text{des}}(t) = v(t) + 0.6a9t)$.

C. Safety Control Layer

We additionally wrap our controller in a safety layer to minimize potential risks. An overview of the different components of the model can be seen in Fig. 9. We use the FollowerStopper from [9], [36], with modifications to the previously used coefficients to decrease the range of actions where the safety controller might override our controller. while remaining safe

The FollowerStopper serves as a velocity controller, navigating the speed of individual vehicles to a predefined desired speed v_{des} while maintaining a safe gap with the vehicle ahead. Following this model, the command velocity v^{cmd} of an CAV is defined as:

$$v^{\text{cmd}} = \begin{cases} 0 & \text{if } h_\alpha \leq \Delta x_1 \\ v\frac{\Delta x - \Delta x_1}{\Delta x_2 - \Delta x_1} & \text{if } \Delta x_1 \leq h_\alpha \leq \Delta x_2 \\ v + (v_{\text{des}} - v)\frac{\Delta x - \Delta x_2}{\Delta x_3 - \Delta x_2} & \text{if } \Delta x_2 \leq h_\alpha \leq \Delta x_3 \\ v_{\text{des}} & \text{otherwise} \end{cases} \tag{5}$$

where $v = \min(\max(v_l, 0), v_{\text{des}})$ and Δx_k is defined as:

$$\Delta x_k = \Delta x_k^0 + \frac{1}{2d_k}(\Delta v_-)^2, \quad k = 1, 2, 3 \tag{6}$$

where $\Delta v_- = \min(\Delta v, 0)$ is the negative arm of difference between the speed of the lead vehicle and the speed of the CAV. See [9] for more details. Here v_{des} is the output of the controller described in Sec. V-B. We note that while the FollowerStopper controller is designed to drive smoothly and has safety mechanisms [9], safety can difficulty be fully guaranteed in real-world conditions. During the experiment, since safety was of utmost importance, drivers were instructed to disable the control whenever they had any doubt about the behavior of the control or of the safety wrapper.

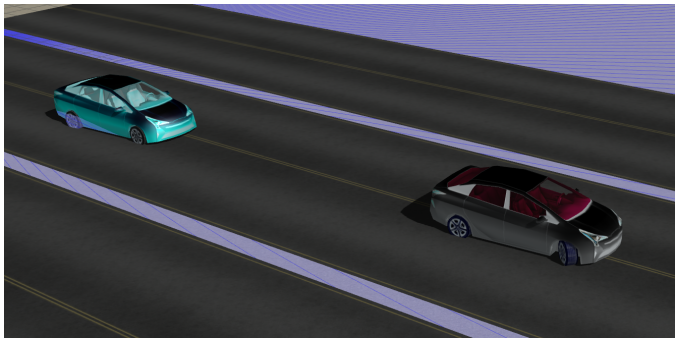

Fig. 7. The 2-vehicle simulated scenario used for assessing the behavior of the RL controller. The lead car performs a series of abrupt starts and stops that the RL car behind it must safely respond to.

D. Software Controller Verification

Before evaluating the controller as a candidate for deployment on hardware, a transfer-learning test is conducted to assess the effectiveness of the controller when exposed to varying dynamics that could potentially be experienced during the real-world test. We refer to this as the *Software-in-the-loop* (SWIL) tests. The SWIL test is intended to elucidate potential failure modes of the wrapped RL controller by running it in a different environment with out-of-distribution dynamics changes from the training environment. In particular, the SWIL test contains different sized time-steps for integrating the dynamics, unseen vehicular dynamics, and a new set of leader trajectories. The transfer-learning environment consists of a 2-vehicle leader-follower pair (in which our controller is the follower) where both vehicles are controlled via ROS [37] and simulated in Gazebo [38]. The vehicles in Gazebo have significantly different dynamics than our simulator.

An initial software-in-the-loop (SWIL) step is taken to check functional correctness and interface testing in a 2-vehicle Gazebo simulation [38] as is shown in Fig. 8. In this 2-vehicle scenario, structured velocity profiles (e.g., constant acceleration, sinusoidal, trapezoidal) are given as an input to the leader vehicle, and the RL controller is deployed on its follower vehicle to ensure outputs are not unusual (e.g. acceleration is bounded within reasonable values, appropriate safety gaps are maintained, etc.) and yield safe behavior. An example of one of the tests, consisting of sharp starts-and-stops of the lead vehicle is depicted in Figure 8.

E. Hardware Deployment Stack

Once the controller has passed all the tests in simulation, it can be deployed onto the hardware. Like many modern vehicles, the Toyota Rav4 on which we deploy our controller uses distributed communicating electronic control units (ECUs)

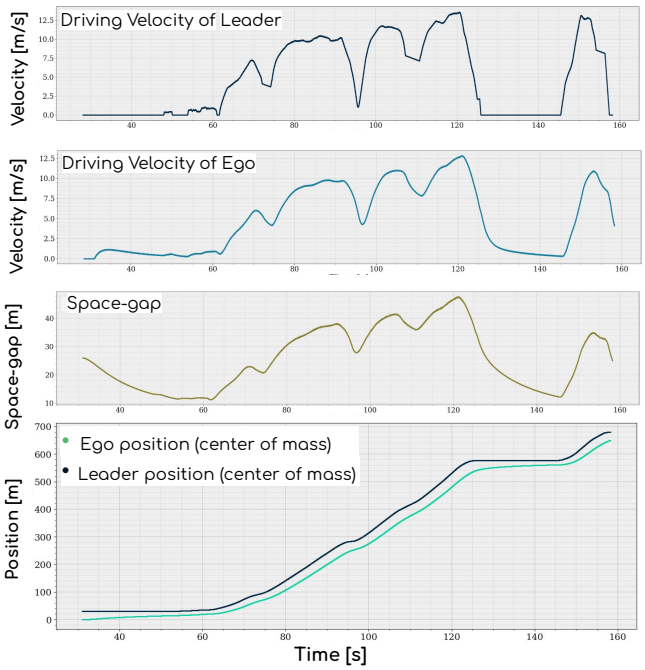

Fig. 8. Resultant RL velocity and space-gap profiles of the RL controller when following behind a rapidly accelerating and decelerating vehicle in Gazebo.

that exchange information using the Controller Area Network (CAN) protocol. This enables a drive-by-wire architecture, where sensors distribute messages about the vehicle's state and modules can actuate control using the CAN bus. This infrastructure makes it possible to connect third-party devices to the CAN bus to both read and send information to a vehicle's modules. This concept was leveraged both for recording data as well as executing the validation experiments that replace the vehicle's stock cruise controller with our experimental controller.

Since CAN is a standard protocol, the method of viewing CAN messages is a trivial task (if the CAN codes are known) with CAN hardware. Many companies provide hardware to interface with the CAN bus in vehicles, a common one being an OBD-II reader for mandated vehicle diagnostics and emission compliance. A greater challenge is to have a device that parses CAN data and then with a relay switch replace messages in real time to actuate according to experimental control algorithms. Fortunately, there are both commercial and open-sourced implementations of such a device, but software interfaces may not be sufficiently reliable to utilize them in real time.

Libpanda [26] was designed as a high-performance library for data collection and vehicle control on low-cost embedded computers like the Raspberry Pi. It creates a direct interface with the vehicle control system. The Can2Ros package [30] creates a bridge between vehicle CAN data and the Robotics Operating System (ROS) [39], to expose components of the vehicle's cruise controller to our customized software controllers, allowing for acceleration command requests.

In order to deploy our trained neural network controllers onto the hardware, we first convert them into the ONNX format. Using MATLAB's Simulink, the neural network is embedded in a ROS node subscribing to pertinent CAN-to-ROS data, and its output is sent to the vehicle interface (leveraging the libpanda package) which takes a desired ROS command and sends it via CAN to the vehicle. The vehicle snensors then send data on the CAN bus, which libpanda records and translates into ROS, allowing it to be read by the neural network.

F. Hardware Controller Validation

For hardware-in-the-loop (HWIL) deployment on the physical vehicle, the controller must first pass a series of three tests to mitigate safety risks from the transition from simulation to the physical vehicle. Once these tests have passed, the controller is eligible to be run in live traffic. All three tests modify the input from the leader vehicle, to ensure performance in non-equilibrium states. Such tests assess the performance for an initial response, vehicle collision, and space gap between two vehicles as the test progresses over time.

First, the controller is tested in a 'Ghost Mode' as in [40] where the vehicle follows a simulated 'Ghost' vehicle as its leader. In this scenario, the distance traveled by the ego car is compared with the distance traveled by the simulated car, and the relative distance is shared directly to the ego car (rather than depending on the on-board sensors to detect the presence of a physical car) through the ROS interface [30]. In the event that there are serious mismatches between the simulation/reality of the vehicle dynamics, the Ghost Mode test will fail due to a simulated collision into a virtual vehicle.

Second, the controller is tested with a human-in-the-loop (HIL) as described in [40], in which desired controller feedback is performed by the human operator. This second test occurs on a low-traffic, high-speed route. Here the vehicle sensors feed real-time data into the controller, and the controller outputs a desired velocity. A passenger periodically reads out the desired velocity to the driver who attempts to faithfully actuate it. If the controller provides unsafe input to the HIL it is rejected by the driver to maintain safety and replaced with human control.

Thirdly and finally, the controller is used on a low-traffic, high-speed route to test the complete hardware-in-the-loop control system. This test was typically performed on the same freeway route but during low-traffic periods.

Once these validation tests have been successfully passed, the controller is ready to be tested in heavy traffic on the high-speed I-24 roadway segment.

VI. Experiment Results

In this section, we present the results of our research, with a primary focus on analyzing the performance of our controller within a simulated environment. Our key metric of interest is energy efficiency, measured in terms of percentage improvements, as we compare a CAV using our RL controller against an IDM controller. The simulation results highlight substantial energy savings, particularly at lower speeds, where the RL controller exhibits notable wave-smoothing behavior, mitigating sharp accelerations and decelerations. Additionally,

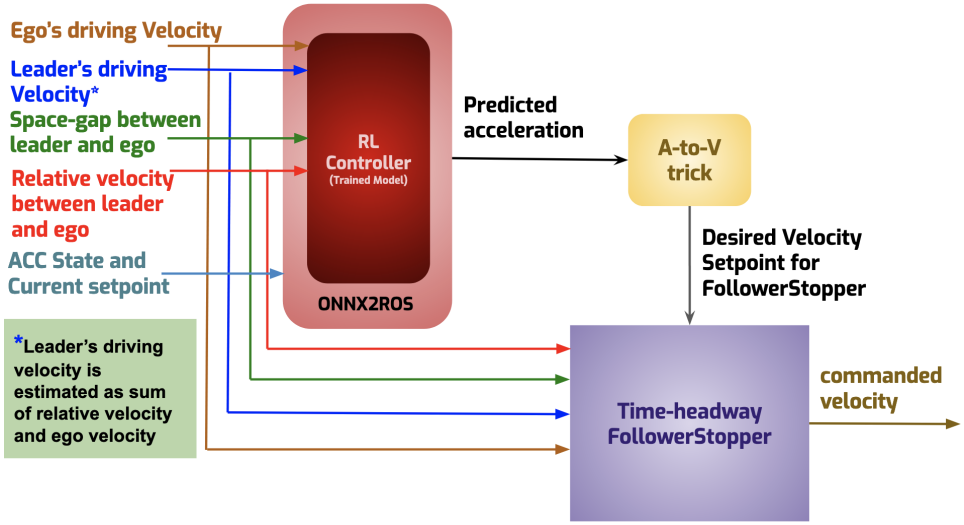

Fig. 9. A schematic diagram of the controller structure we used for the traffic smoothing experiment. The ONNX2ROS framework is used to make predictions using the trained RL model in real-time. The model's outputted acceleration is transformed into a desired set speed using the A-to-V trick described in Sec. V-B. That desired speed is then fed into the safety FollowerStopper controller described in Sec. V-C, which commands a velocity to the car. Note that both the RL model and the safety controller take some combination of ego speed, leader speed, and space gap as inputs.

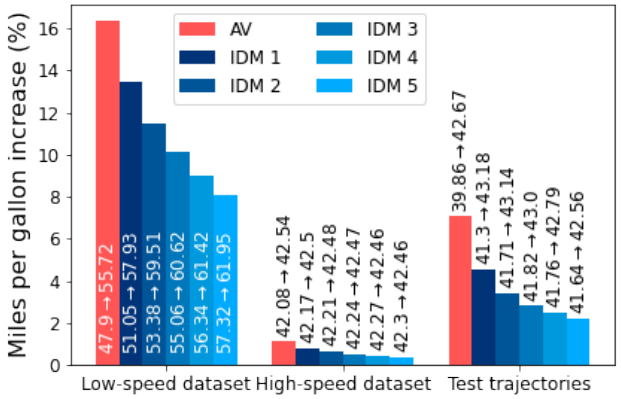

Fig. 10. Percent improvement in MPG relative to a baseline in which the CAV in the platoon in Fig. 5 is replaced by an IDM vehicle. Each column contains both percent improvement on the y-axis and MPG values used to compute this improvement inside each column with IDM (CAV) on the left (right) of the arrow. High and low-speed columns are over the training set. The "Test trajectories" column is the controller evaluated on data from the physical test.

we touch upon the analysis of the RL controller behavior, shedding light on its role in achieving these energy savings. While we acknowledge that only limited real-world experimentation has been conducted, we briefly mention the promising findings from these initial tests, indicating potential fuel efficiency improvements.

A. Simulation Results

Here we analyze the performance of the controller in our simulator in terms of energy efficiency, which we measure in the industry-standard miles per gallon (MPG) unit, although here we are only interested in percentage improvements. In Fig. 10, we compare the energy consumption of the CAV and all vehicles in the platoon (as shown in Fig. 5) when the CAV is using our RL controller compared to an IDM controller, over the whole training dataset. We split the trajectories by leader speed, computing the energy savings at leader speeds above and below $18\frac{\text{m}}{\text{s}}$, which is the speed boundary beyond which IDM vehicles with the parameters used in this work go from being string-unstable to string-stable. The results in the left and middle columns indicate that most of the expected energy improvements from the controller will come at low speeds. While these savings are significant, in more complex settings imperfections in actuation, modeling of human drivers, and cut-ins would likely lower the actual improvement. The rightmost column is described in Sec. VI-C.

In order to illustrate where the energy savings may come from, we show a simulated dataset trajectory as well as the velocity of a CAV driving behind it in Fig. 11. We can observe how the speed profile of the CAV is a smoothed version of the leader's, attempting to not go quite as high (resp. quite as low) as the leader in sharp acceleration (resp. braking) phases, and attenuating low-frequency oscillations.

Fig. 12 shows a similar plot, but using a sinusoidal leader trajectory instead of a real-world one, and comparing the wave-smoothing behavior of the RL control to what the IDM does. We can observe how the CAV reduces the amplitude of the wave by leaving some distance to the leader while the IDM follows the leader much more closely. This behavior repeats with other speeds and wave amplitudes (although the larger the period, the closer RL and IDM become), and can begin to explain the smoothing performances of RL.

B. Controller Behavior Analysis

To understand the improvements described in Sec. VI-A, we can build two-dimensional slices of the controller behavior to analyze what it is doing and how it differs from the IDM model.

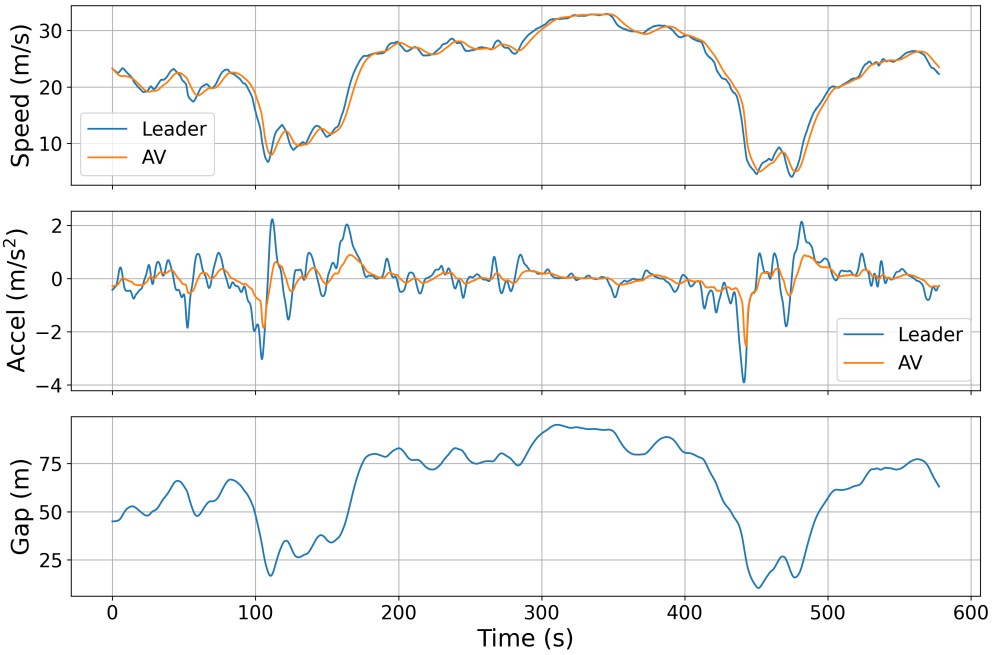

Fig. 11. Speed, instantaneous acceleration and space gap of an RL-controlled CAV following a leader vehicle replaying one trajectory from the dataset, as well as the speed and acceleration of that leader vehicle for reference.

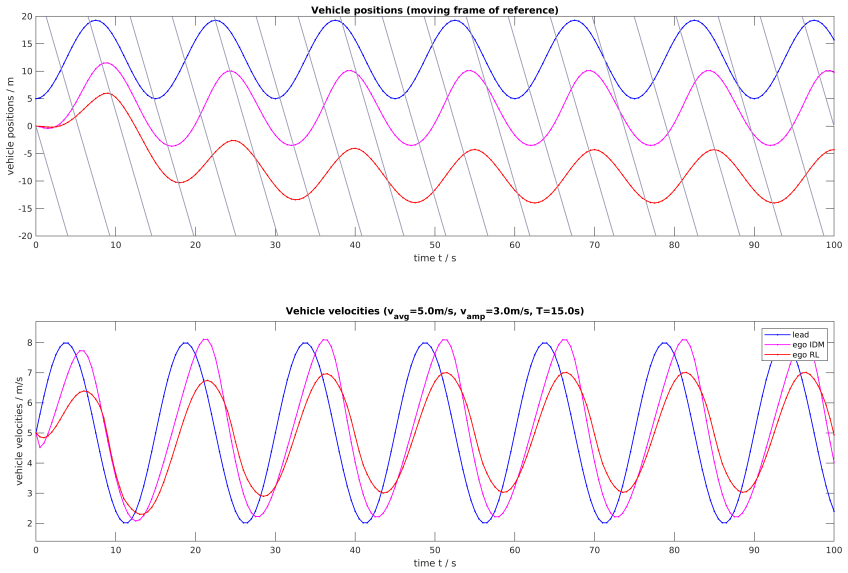

Fig. 12. Position and speed of a lead vehicle whose speed follows a sinusoidal wave of mean $5\frac{\text{m}}{\text{s}}$, amplitude $3\frac{\text{m}}{\text{s}}$ and period 15s, as well as of a CAV following this vehicle, comparing an IDM-controlled CAV to an RL-controlled one. Note that position is represented relative to a linearly-moving frame of reference (represented as the diagonal gray lines) so that the leader position is displayed as a sine wave.

We begin by analyzing whether the RL performs better at smoothing waves than IDM. Fig. 13 shows the growth factor of the RL control over a variety of equilibrium speeds and wave periods. A growth larger than 1 means that the control will amplify the waves, less than 1 attenuate it, and equal to 1 leave it as is. We can observe that RL is stable and has a lower growth factor up to quite higher periods than IDM up to equilibrium speeds above $16\frac{\text{m}}{\text{s}}$, which is reasonable as we are mostly concerned with waves at lower speeds.

Fig. 14 shows the acceleration profile of RL for equal ego and leader velocities, as a function of space gap. We can observe that RL has larger equilibrium gaps than IDM on average (up to speeds above $25\frac{\text{m}}{\text{s}}$). We hypothesize that opening large gaps in order to absorb the potential braking

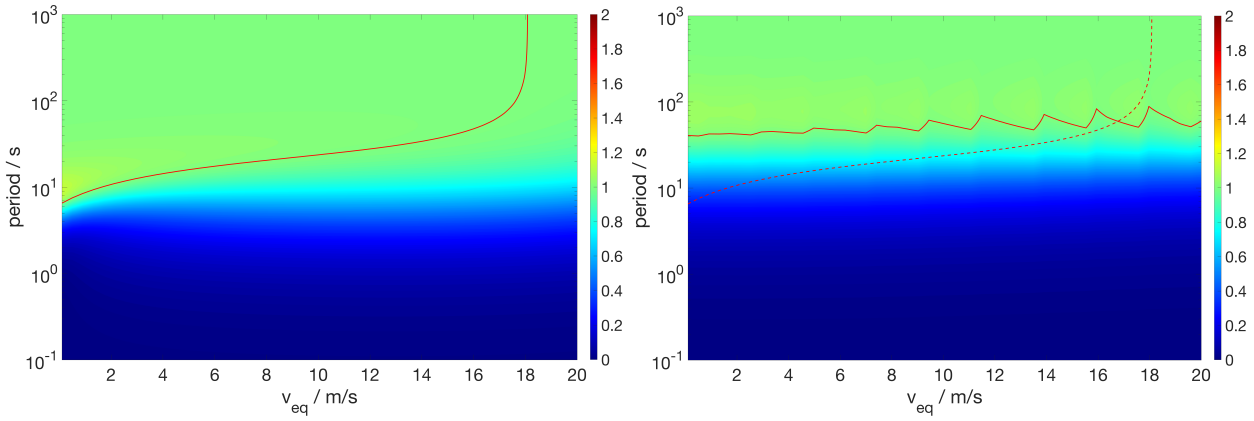

Fig. 13. Regions of instability at equilibrium for IDM (left) and RL (right). The solid red line shows the boundary between stable and unstable for the RL controller, while the dashed red line shows the same for the IDM controller, for comparison. The RL controller stabilizes the system up to lower frequencies at lower speeds (up to periods of 40s) but at higher speeds stabilizes fewer high frequencies than an IDM controller. It is unclear why the RL controller results in this saw-pattern boundary.

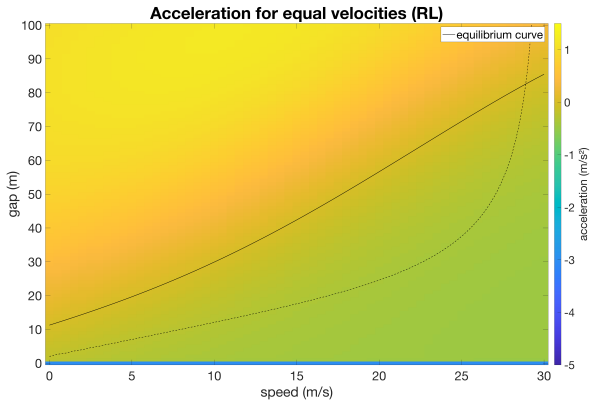

Fig. 14. Acceleration heatmap of the RL control as a function of ego speed, which is assumed equal to leader speed, and space gap. The solid black line shows the acceleration=0 region, i.e., the equilibrium gaps for each speed, assuming the CAV and the leader drive at the same speed. The dashed black line is the corresponding equilibrium curve for an IDM controller.

of the leader is a component of how the RL control achieves wave smoothing.

Finally, Fig. 15 shows acceleration profiles for different ego and leader velocities, where we can observe how the RL control brakes more smoothly and is willing to open larger gaps than IDM.

C. Highway Field Tests

In this section, we describe a set of validation experiments conducted on the segment of I-24 shown in Fig. 2 during August 2021. We assess the success of the controller deployment onto CAVs by showing an accurate match in the maintained AV speed and gap between simulation and reality. Finally, we seek to determine whether our controller improved the energy efficiency of its platoon.

Fig. 19 shows four vehicles from the eleven-vehicle platoon of alternating humans and CAVs that we deployed on I-24. The vehicles are arranged with two human drivers at the front of the platoon to serve as test probes. These vehicles are unaffected

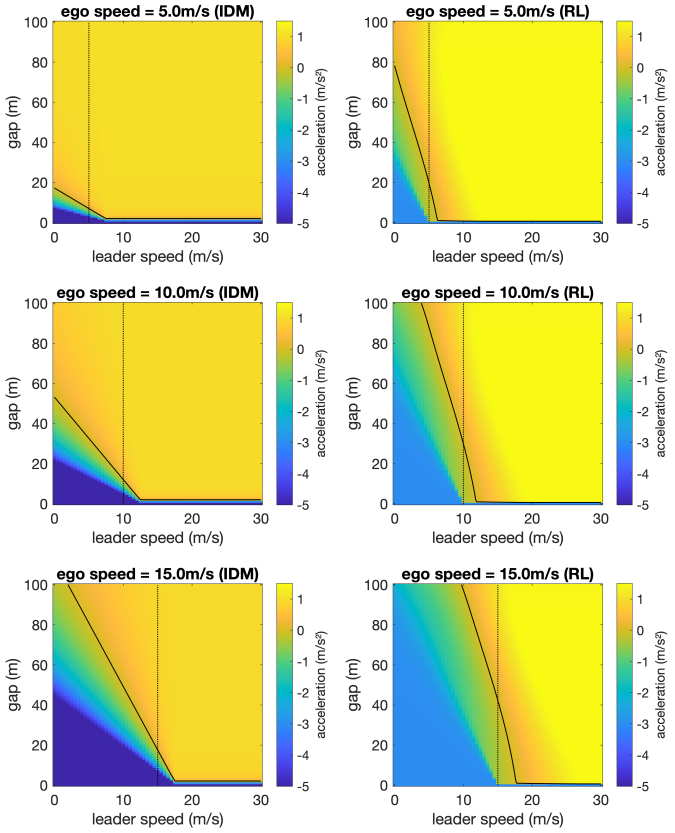

Fig. 15. Acceleration heatmaps (in color) of the IDM (left column) versus our RL control (right column) as a function of leader speed and space gap, for different fixed ego speeds of $5\frac{\text{m}}{\text{s}}$ (first row), $10\frac{\text{m}}{\text{s}}$ (second row) and $15\frac{\text{m}}{\text{s}}$ (third row). The solid black line shows the accel=0 boundary, and the dotted black line shows the x=ego speed, meaning that the intersection of both black lines is the equilibrium gap at equal ego and leader speeds.

by the behavior of the CAVs and can serve as a proxy measure for the MPG of the unsmoothed traffic. We then alternate four CAVs and human drivers going down the platoon. We chose the alternating order rather than a continuous platoon as we expect the platoon to organically spread through interaction with human drivers that change into the platoon's lane. Each

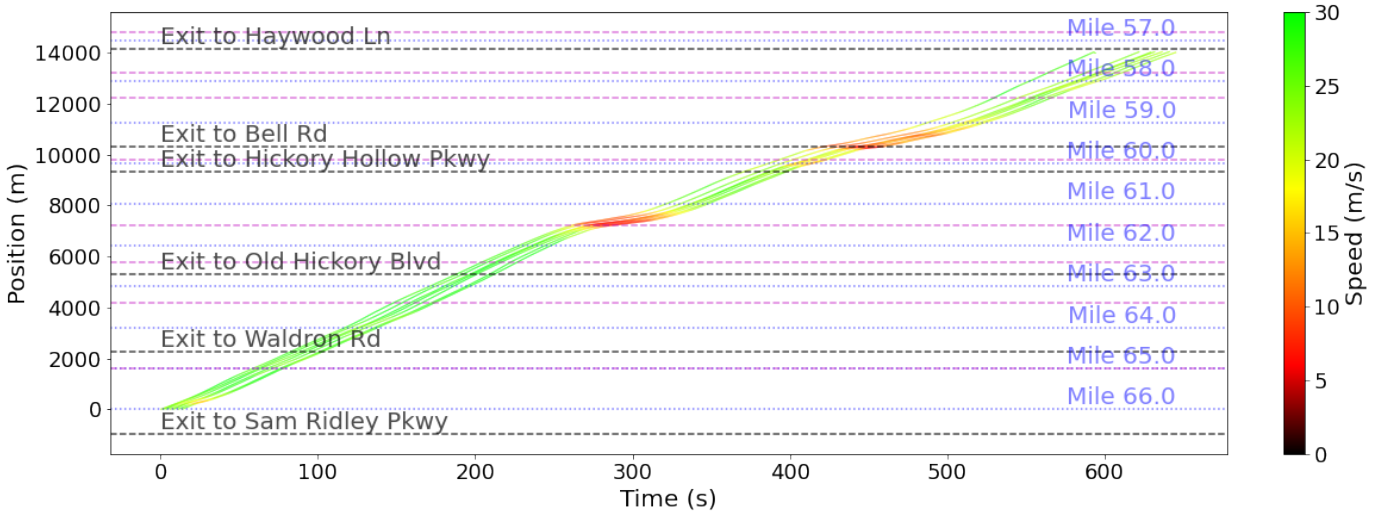

Fig. 16. Time-space diagram showing the trajectories of our platoon of vehicles during the first test. We can observe two regions of congestion (visible in red) where the CAV may have a smoothing effect.

human-CAV pair constitutes a small sub-platoon on which we can measure the influence of the CAV on its following vehicles.

For each test, the platoon merged onto the highway without any non-platoon vehicles lane-changing into it, thanks to the help of the police department. Once on the highway, non-platoon traffic cut in and out of our platoon. Our vehicles were only instrumented to sense the vehicle in front of them, so the number of vehicles that entered our platoon may be unknown but can be established by a lower bound when examining discontinuities in lead vehicle distance, and may be explored in future work. We note that all of our 11 vehicles drove in a single lane throughout each experiment, and our human drivers were instructed to drive as they normally would, so as to not bias the experiment.

We conducted field experiments on August 2nd, 4th, and 6th of 2021, in which many controllers were tested. Each day we launched the platoon of vehicles three times, bringing the vehicles back to the start of the highway section in between each run. The controller presented here was actuated on 08/06, over three tests that started at 6:45, 7:29, and 8:36 AM. Fig. 16 shows individual vehicle trajectories on a time-space diagram from the 6:45 AM test; the two regions of red correspond to potential sources of congestion that the controller may have reduced.

The deployment of the controller from simulation to real vehicles was overall successful – all tests running safely and smoothly. We investigated the effect of the sim-to-real gap induced by the presence of cut-ins and cut-outs, which we did not have when training our controller, as well as imperfect modeling of the transfer function of the CAV.

First, we attempt to compute a counterfactual baseline in which we replay our controller in simulation behind a trajectory collected during the tests, thereby comparing a simulated CAV's behavior to one that has been tested on the highway. However, because the real-world trajectories include lane changes that our simulation cannot perfectly replicate, and because replaying a different controller behind the trajectory might affect the cut-in and cut-out frequencies, we have to make some assumptions. We assume that both the timing and spacing of these lane changes remain the same in both the real and simulated scenarios. Besides, to avoid overly aggressive lane changes in the simulation, namely cut-ins that would be more aggressive than what the real-world CAV experienced, we also adjust the following distance between vehicles when needed, using a specific formula to balance between the simulated and real-world conditions. To that end, at each time-step t where a cut-in would leave the CAV with a space-gap h_t^{sim} while the real-world CAV experienced a space-gap h_t^{real}, we set $h_t^{\text{sim}} = \max(h_t^{\text{sim}}, \min(h_{t-1}^{\text{sim}}, h_t^{\text{real}}))$.

Fig. 17 shows the velocity and time-gap (space-gap divided by velocity) of a CAV from the validation experiment as well as the replay of the trajectory in our simulation using the counterfactual cut-in mechanism mentioned above. The velocity profile of the vehicle closely matches its expected behavior computed in the simulation. Although there are mismatches around cut-ins and cut-outs (regions where the time-gap changes discontinuously), the time-gaps are relatively close and we can observe the vehicle roughly tracking a three-second time-gap in both cases. We observed similar results on the other trajectories we collected during the tests.

Finally, we analyze the potential fuel efficiency improvements from the validation experiment. The third column in Fig. 10 depicts the energy savings obtained when replaying in simulation using the trajectories collected during the experiments (instead of the training data which did not have lane changes) using the counterfactual cut-in mechanism mentioned earlier. We observe that the fuel efficiency of the CAV has improved by 8% with additional small gains for the IDM vehicles. Fig. 18 shows the density of accelerations taken by the IDM vs. the CAV; the higher density of large accelerations of the IDM vehicle is likely the reason for the improved fuel efficiency of the RL CAV over the IDM CAV. Unfortunately, the day of the deployment featured limited congestion so

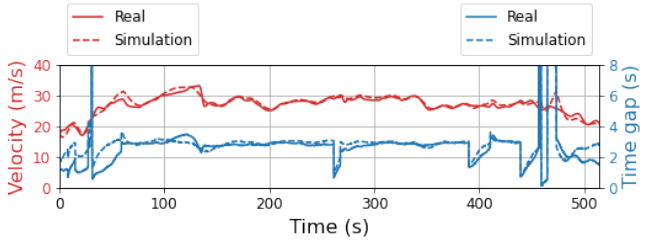

Fig. 17. Comparison of velocity and time-gap between a real (solid) and simulated (dashed) roll-out. There are small divergences that occur around the cut-ins but the car mostly maintains a three-second time-gap in both cases.

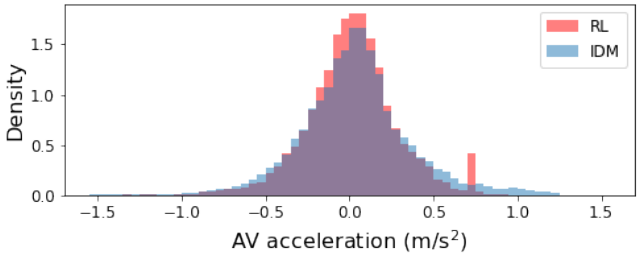

Fig. 18. Histogram showing the density of the CAV acceleration when simulating a CAV or an IDM vehicle behind leader trajectories from the tests. The CAV case places less mass at high, energy-consuming accelerations. The peak observed at 0.75 corresponds to situations in which the lead vehicle is out of range of radar due to a cut-out.

potential improvements are smaller than might be observed in heavier traffic conditions. More experimental testing on a number of days is needed to provide conclusive experimental energy savings results. Here we compute estimates from simulations using our models on experimental trajectory data.

VII. Conclusion

In this work, we proposed and tested a pipeline that allows for effective validation and training of traffic smoothing controllers. The behavior of the vehicle in the validation experiment closely matches its expected simulation behavior, suggesting that our pipeline is an effective mechanism for validating controllers. Besides, even though we did not have the means to measure energy improvement on the highway, evaluating in simulation using real-world trajectories indicates that our controller should have been doing smoothing, given that the sim-to-real gap is low.

We observe that the main feature missing in our environment is the presence of counterfactual lane changes. In future work, this can be addressed using the observed lane changes in the data to build a single-lane lane-changing model that can be used to extend our simulation. Additional field experiments can support the assessment of our approaches in a range of traffic congestion levels.

VIII. Acknowledgments

Eugene Vinitsky is a recipient of an NSF Graduate Research Fellowship and funded by the National Science Foundation under Grant Number CNS-1837244. Computational resources for this work were provided by the Savio cluster at UC Berkeley. This material is also based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) award number CID DE-EE0008872. The views expressed herein do not necessarily represent the views of the U.S. Department of Energy or the United States Government. This material is based upon work supported by the National Science Foundation under Grant Numbers CNS-1837244 (A. Bayen), CNS-1837652 (D. Work), CNS-1446690 (B. Seibold), CNS-1446702 (D. Work). Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

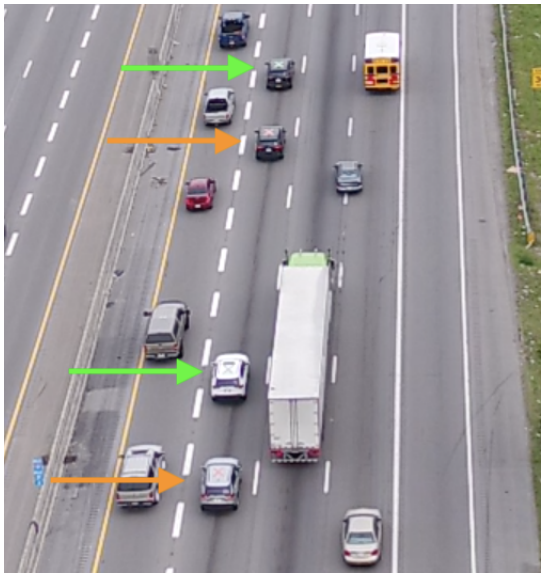

Fig. 19. 4 of 11 vehicles in formation on the roadway. Green arrows and green X on the roof indicate CAV (CAV), and orange arrows and orange X on the roof indicate human-driven sensing vehicle (H), which are instructed to drive as they usually would. During experiments, our platoon was formed in this order: [H, H, CAV, H, CAV, H, CAV, H, CAV, H, H]. As the experiment goes on, traffic flow cuts in and out of our platoon, which we have no control over. Each experiment happens in a single-lane, without any of our vehicles ever cutting in or out.

Appendix
System Identification

Let $x(t), v(t)$, and $a(t)$ be the position, velocity, and acceleration of the ego vehicle at time t. At any moment t, one can control the acceleration $a(t)$ of the ego vehicle through a bounded acceleration command signal $a_{\text{cmd}}(t)$.

For simplicity, we parametrize the acceleration dynamics as the following first-order system

$$\frac{A(s)}{A_{\text{cmd}}(s)} = \frac{k_1}{s + k_2}, \tag{7}$$

where $A(s) = \mathcal{L}[a(t)]$ and $A_{\text{cmd}}(s) = \mathcal{L}[a_{\text{cmd}}(t)]$ for some parameters k_1 and k_2. The transfer function (7) is equivalent in the time domain to the following state space model

$$\dot{a}(t) = -k_2 a(t) + k_1 a_{\text{cmd}}(t). \tag{8}$$

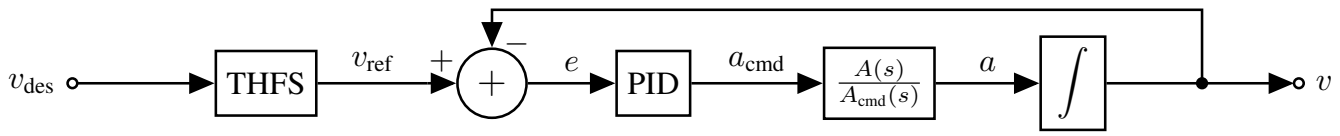

Fig. 20. Block diagram demonstrating our recent hardware design, which features a time headway follower stopper (THFS) as the safety filter and a PID controller with output saturation as the velocity controller.

However, to ensure safety, we do not expose the interface of a_{cmd} to users. Instead, one can only indirectly access it by commanding a desired velocity signal $v_{\text{des}}(t)$. The desired velocity is passed through a safety filter, producing a reference velocity signal $v_{\text{ref}}(t)$, which is then tracked by a velocity controller. An example of our recent hardware design can be seen in Fig. 20.

To illuminate the underlying structure of the system in Fig. 20, we assume that v_{des} is safe, i.e., $v_{\text{ref}} = v_{\text{des}}$, and that the PID is never saturated. Let the PID takes the following form,

$$\frac{A_{\text{cmd}}(s)}{E(s)} = k_p + \frac{k_i}{s} + \frac{k_d \cdot k_n}{1 + k_n/s} \tag{9}$$

$$= \frac{(k_d k_n + k_p)s^2 + (k_p k_n + k_i)s + k_i k_n}{s^2 + k_n s}, \tag{10}$$

where $A_{\text{cmd}}(s) = \mathscr{L}[a_{\text{cmd}}(t)], E(s) = \mathscr{L}[e(t)]$, and $e(t) := v_{\text{ref}}(t) - v(t)$ for some parameters k_p, k_i, k_d and k_n. The controllable canonical form of (10) is

$$\begin{aligned} \begin{bmatrix} \dot{p}(t) \\ \dot{q}(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & -k_n \end{bmatrix} \begin{bmatrix} p(t) \\ q(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (v_{\text{des}}(t) - v(t)), \\ a_{\text{cmd}}(t) &= \begin{bmatrix} k_i k_n & k_i - k_n^2 k_d \end{bmatrix} \begin{bmatrix} p(t) \\ q(t) \end{bmatrix} \\ &\quad + (k_d k_n + k_p)(v_{\text{des}}(t) - v(t)), \end{aligned} \tag{11}$$

where p, q are some internal states of the PID controller.

Combining (8) and (11), we obtain the vehicle dynamics as the following linear time-invariant (LTI) system

$$\begin{aligned} \begin{bmatrix} \dot{p}(t) \\ \dot{q}(t) \\ \dot{v}(t) \\ \dot{a}(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -k_n & -1 & 0 \\ 0 & 0 & 0 & 1 \\ \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 \end{bmatrix} \begin{bmatrix} p(t) \\ q(t) \\ v(t) \\ a(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ -\kappa_3 \end{bmatrix} v_{\text{des}}(t), \\ y(t) &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p(t) & q(t) & v(t) & a(t) \end{bmatrix}^\top, \end{aligned} \tag{12}$$

where y is the output of the system and

$$\begin{bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \\ \kappa_4 \end{bmatrix} = \begin{bmatrix} k_1 k_i k_n \\ k_1 k_i - k_1 k_n^2 k_d \\ -(k_1 k_d k_n + k_1 k_p) \\ -k_2 \end{bmatrix}. \tag{13}$$

The values of system parameters are provided in Table I, where k_1 and k_2 are determined via system identification, whereas the rest are selected during controller design experiments. Specifically, system identification is performed on data collected by recording the inputs and outputs of system (7) under different driving velocities. The PID is first designed in Simulink and then tested on the actual hardware. It can be verified that system (12) with parameters in Table I is fully controllable and observable.

TABLE I
PARAMETERS OF VEHICLE SYSTEM DYNAMICS.

k_1	k_2	k_p	k_i	k_d	k_n
1.745	1.566	2.720	0.0656	1.340	7.549

REFERENCES

[1] Nice, M., Lichtle, N., Gumm, G., Roman, M., Vinitsky, E., Elmadani, S., and Bunting, M., Bhadani, R., Gunter,G., Kumar, M., and McQuade, S., Denaro, C., Delorenzo, R., Piccoli, B., Work, D. Bayen, A. Lee, J., Sprinkle, J. and Seibold, B., "The I-24 trajectory dataset," doi.org/10.5281/zenodo.6366761, 2021.

[2] N. Lichtlé, E. Vinitsky, M. Nice, B. Seibold, D. Work, and A. M. Bayen, "Deploying traffic smoothing cruise controllers learned from trajectory data," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2884–2890.

[3] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[4] P. Ioannou, Z. Xu, S. Eckert, D. Clemons, and T. Sieja, "Intelligent cruise control: theory and experiment," in *Proceedings of 32nd IEEE Conference on Decision and Control*, Dec 1993, pp. 1885–1890 vol.2.

[5] B. Besselink and K. H. Johansson, "String stability and a delay-based spacing policy for vehicle platoons subject to disturbances," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4376–4391, Sep. 2017.

[6] C.-Y. Liang and H. Peng, "Optimal adaptive cruise control with guaranteed string stability," *Vehicle System Dynamics*, vol. 32, no. 4-5, pp. 313–330, 1999. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1076/vesd.32.4.313.2083

[7] D. Swaroop and J. Hedrick, "String stability of interconnected systems," *IEEE Transactions on Automatic Control*, vol. 41, no. 3, pp. 349–357, 1996.

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

[9] R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 205–221, 2018.

[10] M. R. Flynn, A. R. Kasimov, J.-C. Nave, R. R. Rosales, and B. Seibold, "Self-sustained nonlinear waves in traffic flow," *Physical Review E*, vol. 79, no. 5, p. 056113, 2009.

[11] I. G. Jin, S. S. Avedisov, C. R. He, W. B. Qin, M. Sadeghpour, and G. Orosz, "Experimental validation of connected automated vehicle design among human-driven vehicles," *Transportation research part C: emerging technologies*, vol. 91, pp. 335–352, 2018.

[12] J. Ma, X. Li, S. Shladover, H. A. Rakha, X.-Y. Lu, R. Jagannathan, and D. J. Dailey, "Freeway speed harmonization," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 78–89, 2016.

[13] L. Jiang, Y. Xie, N. G. Evans, X. Wen, T. Li, and D. Chen, "Reinforcement learning based cooperative longitudinal control for reducing traffic oscillations and improving platoon stability," *Transportation Research Part C: Emerging Technologies*, vol. 141, p. 103744, 2022.

[14] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2118–2125.

[15] Y. Li, S. Chen, R. Du, P. Y. J. Ha, J. Dong, and S. Labi, "Using empirical trajectory data to design connected autonomous vehicle controllers for traffic stabilization," 2020.

[16] V. Alexiadis, J. Colyar, J. Halkias, R. Hranac, and G. McHale, "The next generation simulation program," *Institute of Transportation Engineers. ITE Journal*, vol. 74, no. 8, p. 22, 2004.

[17] M. Makridis, K. Mattas, B. Ciuffo, F. Re, A. Kriston, F. Minarini, and G. Rognelund, "Empirical study on the properties of adaptive cruise control systems and their impact on traffic flow and string stability," *Transportation research record*, vol. 2674, no. 4, pp. 471–484, 2020.

[18] V. L. Knoop, M. Wang, I. Wilmink, D. M. Hoedemaeker, M. Maaskant, and E.-J. Van der Meer, "Platoon of sae level-2 automated vehicles on public roads: Setup, traffic interactions, and stability," *Transportation Research Record*, vol. 2673, no. 9, pp. 311–322, 2019.

[19] G. Gunter, D. Gloudemans, R. E. Stern, S. McQuade, R. Bhadani, M. Bunting, M. L. Delle Monache, R. Lysecky, B. Seibold, J. Sprinkle *et al.*, "Are commercially implemented adaptive cruise control systems string stable?" *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[20] K.-c. Chu, "Decentralized control of high-speed vehicular strings," *Transportation science*, vol. 8, no. 4, pp. 361–384, 1974.

[21] K. Jang, E. Vinitsky, B. Chalaki, B. Remer, L. Beaver, A. A. Malikopoulos, and A. Bayen, "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 291–300.

[22] C. Wu, A. R. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: A modular learning framework for mixed autonomy traffic," *IEEE Transactions on Robotics*, 2021.

[23] J. Cui, W. Macke, H. Yedidsion, A. Goyal, D. Urielli, and P. Stone, "Scalable multiagent driving policies for reducing traffic congestion," *arXiv preprint arXiv:2103.00058*, 2021.

[24] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani, and Y. P. Fallah, "Altruistic maneuver planning for cooperative autonomous vehicles using multi-agent advantage actor-critic," *arXiv preprint arXiv:2107.05664*, 2021.

[25] D. A. Lazar, E. Bıyık, D. Sadigh, and R. Pedarsani, "Learning how to dynamically route autonomous vehicles on shared roads," *Transportation Research Part C: Emerging Technologies*, vol. 130, p. 103258, 2021.

[26] M. Bunting, R. Bhadani, and J. Sprinkle, "Libpanda: A high performance library for vehicle data collection," in *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*, 2021, pp. 32–40.

[27] R. Bhadani, M. Bunting, M. Nice, N. M. Tran, S. Elmadani, D. Work, and J. Sprinkle, "Strym: A python package for real-time can data logging, analysis and visualization to work with usb-can interface," in *2022 2nd Workshop on Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities Workshop (DI-CPS)*, 2022, pp. 14–23.

[28] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, aug 2000. [Online]. Available: https://doi.org/10.1103%2Fphysreve.62.1805

[29] K. J. Astrom, "Optimal control of markov decision processes with incomplete state estimation," *J. Math. Anal. Applic.*, vol. 10, pp. 174–205, 1965.

[30] S. Elmadani, M. Nice, M. Bunting, J. Sprinkle, and R. Bhadani, "From CAN to ROS: A monitoring and data recording bridge," in *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*, 2021, pp. 17–21.

[31] J. W. Lee, G. Gunter, R. Ramadan, S. Almatrudi, P. Arnold, J. Aquino, W. Barbour, R. Bhadani, J. Carpio, F.-C. Chou, M. Gibson, X. Gong, A. Hayat, N. Khoudari, A. R. Kreidieh, M. Kumar, N. Lichtlé, S. McQuade, B. Nguyen, M. Ross, S. Truong, E. Vinitsky, Y. Zhao, J. Sprinkle, B. Piccoli, A. M. Bayen, D. B. Work, and B. Seibold, "Integrated framework of dynamics, instabilities, energy models, and sparse flow controllers," in *Proceedings of the Workshop on CPS Data for Transportation and Smart cities with Human-in-the-loop*, 2021.

[32] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[33] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.

[34] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[35] M. Treiber and V. Kanagaraj, "Comparing numerical integration schemes for time-continuous car-following models," *Physica A: Statistical Mechanics and its Applications*, vol. 419, pp. 183–195, 2015.

[36] R. K. Bhadani, B. Piccoli, B. Seibold, J. Sprinkle, and D. Work, "Dissipation of emergent traffic waves in stop-and-go traffic using a supervisory controller," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3628–3633.

[37] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[38] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[39] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[40] M. Nice, S. Elmadani, R. Bhadani, M. Bunting, J. Sprinkle, and D. Work, "CAN coach: vehicular control through human cyber-physical systems," in *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, 2021, pp. 132–142.

Nathan Lichtlé is currently working on his Ph.D. in the Department of Electrical Engineering and Computer Sciences at U.C. Berkeley, and in the CERMICS group at École des Ponts ParisTech. His focus is on reinforcement learning and mixed-autonomy traffic.

Eugene Vinitsky is an assistant professor of Civil and Urban Engineering at NYU Tandon and a member of the C2SMARTER center for congestion reduction. He received his PhD in controls engineering from UC Berkeley, his M.A. in physics from UC Santa Barbara, and his B.S. in physics from Caltech. His research interests span reinforcement learning, multi-agent planning and control, and connected and autonomous vehicles.

Matthew Nice is a Research Engineer at the Institute for Software Integrated Systems. He received a M.Eng. in Cyber-Physical Systems, and a PhD from the Department of Civil and Environmental Engineering, in the Institute for Software Integrated Systems, at Vanderbilt University. He has led experimental research on the impact of technology in connected and automated vehicles on transportation systems. His research has been supported by multiple Dwight D. Eisenhower Transportation Fellowship Awards from FHWA.

Rahul Bhadani is an assistant professor of Electrical & Computer Engineering at The University of Alabama in Huntsville. He received his Ph.D. in Electrical & Computer Engineering from the University of Arizona. He completed his postdoctoral appointment at the Institute for Software Integrated Systems, Vanderbilt University, studying the modeling of cyber-physical power systems using SystemC-AMS. His research focuses on model-based system design for cyber-physical systems and intelligent transportation.

Matthew Bunting received a Ph.D. in Electrical and Computer Engineering at the University of Arizona, and is a research scientist in the Institute for Software Integrated Systems, at Vanderbilt University. His focus is on hardware development of autonomous vehicles, democratizing vehicle data collection and control through open source projects.

Fangyu Wu Fangyu Wu is a PhD candidate in the Electrical Engineering and Computer Sciences department at the University of California, Berkeley. He earned his B.S. and M.S. degrees in Civil and Environmental Engineering from the University of Illinois at Urbana-Champaign in 2016 and 2018, and an M.Eng. in EECS from UC Berkeley in 2019. His research interests encompass optimization approaches to multiagent robotic systems, with a focus on planning and control of automated vehicles.

Benedetto Piccoli is University Professor and the Joseph and Loretta Lopez Chair Professor of Mathematics at Rutgers University - Camden. He also served as Vice Chancellor for Research. His research interests span various areas of applied mathematics, including control theory, traffic flow on networks, crowd dynamics, math finance and application to autonomous driving, population health and bio-medical systems. He is author of more than 300 research papers and 7 books and is the founding editor of Networks and Heterogeneous Media. Piccoli is the recipient of the 2009 Fubini Prize, Plenary speaker at ICIAM 2011, and 2012 inaugural Fellow of American Mathematical Society.

Benjamin Seibold is a Professor of Mathematics and Physics, and the Director of the Center for Computational Mathematics and Modeling, at Temple University. His research areas, funded by NSF, DOE, DAC, USACE, USDA, and PDA, are computational mathematics (high-order methods for differential equations, CFD, molecular dynamics) and applied mathematics and modeling (traffic flow, invasive species, many-agent systems, radiative transfer).

Daniel B. Work is a Professor of Civil and Environmental Engineering and the Institute for Software Integrated Systems, at Vanderbilt University. His interests are in transportation cyber-physical systems.

Jonathan W. Lee received the B.S. degree in engineering physics from the University of California, Berkeley and M.S. and Ph.D. degrees in mechanical engineering from Rice University. At Sandia National Laboratories (2011–13), he completed his postdoctoral appointment studying electrical and material properties via molecular dynamics simulations. He subsequently served as a senior data scientist and product manager on various teams at Uber Technologies, Inc. (2014-2019). Since 2019, he has served as an engineering manager at the University of California, Berkeley and the program manager and Chief Engineer of CIRCLES.

Jonathan Sprinkle is a Professor of Computer Science at Vanderbilt University since 2021. Prior to joining Vanderbilt he was the Litton Industries John M. Leonis Distinguished Associate Professor of Electrical and Computer Engineering at the University of Arizona, and the Interim Director of the Transportation Research Institute. From 2017-2019 he served as a Program Director in Cyber-Physical Systems and Smart & Connected Communities at the National Science Foundation in the CISE Directorate.

Alexandre M. Bayen is the Associate Provost for the Berkeley Space Center at UC Berkeley, and the Liao-Cho Professor of Engineering at UC Berkeley. He is a Professor of Electrical Engineering and Computer Science and of Civil and Environmental Engineering (courtesy). He is a visiting Professor at Google. He is also a Faculty Scientist in Mechanical Engineering, at the Lawrence Berkeley National Laboratory (LBNL). From 2014 - 2021, he served as the Director of the Institute of Transportation Studies at UC Berkeley (ITS).