From Simulation to the Real World: Deep Reinforcement Learning for Training Robust, Wave-Smoothing Policies for Autonomous Vehicles

Kathy Jang

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-26
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-26.html

May 1, 2025

From Simulation to the Real World: Deep Reinforcement Learning for Training Robust, Wave-Smoothing Policies for Autonomous Vehicles

by

Kathy Jang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexandre Bayen, Chair
Associate Professor Anca Dragan
Associate Professor Sergey Levine
Assistant Professor Maria Laura Delle Monache

Spring 2024

From Simulation to the Real World: Deep Reinforcement Learning for Training Robust, Wave-Smoothing Policies for Autonomous Vehicles

Abstract

From Simulation to the Real World: Deep Reinforcement Learning for Training Robust, Wave-Smoothing Policies for Autonomous Vehicles

by

Kathy Jang

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Alexandre Bayen, Chair

With the advent of autonomous vehicles (AVs) comes a broad array of possibilities for control. Looking beyond the immediate wave of research that is focused on training models that can drive safely, this work looks into the future, and aims to develop models for AVs that can achieve more than safe driving. Traffic dynamics are notoriously difficult to model and capture on the micro-level, with behaviors ranging from human-observable to ones we are not aware of, happening every second. Reinforcement learning (RL) is a method which is effective in capturing structure from highly complex, heavy, behavioral data. In this work, we use RL and leverage its ability to understand complex human-driver and traffic dynamics in order to develop policies that are able to not only drive, but drive in a way that can smooth traffic.

With the goal of taking these traffic-smoothing algorithms to the real world, the aim of this work takes a path through three parts, from work done purely in simulation to a eventual 100-AV road test. We first explore the concept of using RL as a means of control for wave-smoothing policy control by examining experiments across a variety of traffic scenarios that demonstrate its effectiveness. This portion happens purely in simulation and explores various components of RL design, from environment design to reward shaping. With the goal of deployment always in mind, we also conduct research on how to develop RL policies that are robust enough to survive the transfer from simulation to the real world while sacrificing minimal performance. Lastly, the work comes together to explore the development and deployment of the MegaVanderTest, the deployment of 100 RL-enabled AVs, and to our knowledge, the largest test of AVs designed to smooth traffic.

To my parents

This PhD is dedicated to my mom and dad, who always put their children first, and to whom I owe the opportunities and the life I've been so lucky to live.

Contents

List of Figures

List of Tables

Acknowledgments

I am enormously grateful to a large number of friends, colleagues, and family for their support during my Ph.D. journey. An enormous thank you to my advisor, Alex Bayen, whose support and direction has been invaluable over the past several years. From the beginning to the end, Alex has gone beyond what I could have ever expected from an advisor. From encouraging me to pursue the subjects I was interested in to providing phenomenal leadership in some of the most ambitious projects I've been so lucky to partake in, I have felt extremely supported these past five years and have learned a great deal from his mentorship. The way Alex treats his students is unparalleled – from helping make final edits rushing toward paper deadlines while I was sick with a nasty fever to offering to the entire lab to feel free to call him and join his family on Zoom during dinner time during COVID if they felt lonely, I am beyond grateful to have been his grad student. An additional tremendous thank you goes to my committee, who have provided me with excellent support and feedback along the way.

To the labmates who have come and gone during these past years, I am grateful for the laughs, the friendship, the teamwork, and so much more. Eugene Vinitsky was the graduate student who took a chance on me as an undergraduate student in Berkeley, and provided me with enormous amounts of support from then until now. Though I have expressed my gratitude, I can't do so enough – from always letting me know about workshops and opportunities that he thought I would be a good fit for, to being a wonderful friend with whom I could scream emo punk music with, I'm extremely grateful. Kanaad Parvate was another one of our early lab, whom, along with Eugene, I shared some very fond memories. Playing handball with a yoga ball, meme-ing around and laughing at the dumbest of jokes, I would go into lab daily excited to do research with my dear friends, and have left with some lifelong friends with an appreciation of the absurd who are some of the funniest people that I know. Thank you also to Nathan Lichtlé and Adit Shah (and once again Eugene) for a lot of the work that we did together up until the MegaVanderTest. It was an immense relief to have people I trusted on my team building up to the largest field test I have ever been involved in. Thank you also to Yashar Farid (who I got to know in Bayen Lab and eventually got to work with at Toyoto Infotech Labs) and Cathy Wu for their mentorship.

One of the major parts of this thesis is the work I did in CIRCLES (see Chapters 8 and 9), which consisted of 50+ people, all of whom I'm very thankful for. Big thanks goes to all the PIs who fearlessly took on and led this very ambitious project and for the opportunity for me to see my very own RL algorithms on real vehicles. The coolest moment of my PhD was being able to drive in a car running the RL policy that I trained for the very first time – I owe that to them, my teammates, and the rest of the group working painstakingly on hardware, speed planning, logistics, and so much more. It could not have happened without all of you. In CIRCLES, I especially want to thank Jonny Lee, who was officially the project manager of the group, but in reality, was so much more. Jonny was involved in the technical details of every single subteam and was invaluable to the success of the project in a way no one else was. He also uniquely understood the pressure I felt being dropped into Nashville with less than a week to go before the test. To say I'm grateful is an understatement – thanks for

keeping me sane, for helping me with test drives, for Taiwanese snacks, for merging pieces of code that I couldn't, and so much more.

I've always felt like I have the best friends in the world. To the friends who have been in my life from elementary school to the ones who I met in just the past year, I cherish you all immensely. I'm grateful for late nights struggling with PintOS, trips to Costa Rica, Berlin, Japan, backpacking trips as kids wondering what our lives would be like, Wine Wednesdays, and everything in between. Major thanks to the lifelong friends I've made during my PhD, who understand this chapter of my life more uniquely than anyone. Yuqing Du, Dhruv Shah, Orr Paradise, and so many more – I truly don't know what I would have done without your friendship. I also believe in all of you more than I do in myself, and am so excited to see how you all change the world.

A large thanks goes to my family, who until my defense, did not know what kind of work I did. I'm very thankful for the support and trust I received growing up. I never received an ounce of pressure to go into any particular field, had the freedom to pursue any interests that I had, and always felt and knew how loved I was. I thank them for always being there for me. In particular, this PhD is dedicated to my dad, who I wish could have watched me graduate and I know would have been proud. He was a scientist through and through and I hope to carry his legacy forward through my work and my life. He is very dearly missed and loved. Lastly, I'm grateful to my partner, Shreyas Kapur, for the superhuman amounts of support and love I have received during our time together. He is the best person I know, more talented, passionate, brilliant, funny, kind than anyone I know, and an inspiration in every way. Thank you for everything.

Chapter 1

Introduction

Autonomous vehicles (AV) represent a monumental shift in transportation paradigms and vehicle technology. While AVs have been talked about and implemented in a variety of ways for decades, the 2010s especially saw a significant and rapid breakthroughs of AV technology, from Tesla's semi-autonomous autopilot system, to other automakers. Traditional automakers like Ford or GM have invested heavily in autonomy, focusing on advanced driver-assistance systems, with features such as adaptive cruise control and lane-keeping assist and prevention, which have become common in modern commercial vehicles. Beyond assisted autonomy, a slew of companies have been working toward fully autonomous vehicles, with various test and pilot programs around the world. Waymo, for instance, began testing its level 4 cars in Arizona in 2017. Non-driver supervised Waymos are now available for commercial use to riders in multiple cities such as San Francisco or or Phoenix. While challenges remain in technology and legislation, the progress that has been seen thus far suggests a promising future for AV technology.

With the significant achievements in technology and adoption that have been seen over the past decade, it is compelling to consider the sheer impacts that a fully autonomous roadway might have on environmental and traffic metrics. Hypothetically, a fully autonomous road network, with AVs and automated road infrastructure components like traffic lights or road meters, could easily yield significant gains in efficiency and energy savings. The vast majority of the time between now and that future, however, will consist of mixed-autonomy road settings. Mixed-autonomy describes a scenario in which some percentage of a road's vehicles are autonomous, but not all. The question that begs to be asked is, "Can we still achieve gains in efficiency and energy savings with a mixed-autonomy fleet?" This question is explored throughout my thesis, with experiments running from simulation to the real-world, indicating that mixed-autonomy is, in fact, a valuable scenario, and has considerable impact on a number of metrics, such as throughput, velocity, fuel savings, and more.

In developing specialized controllers for AVs, reinforcement learning (RL) stands out for its ability to capture structure from heavy, complex, data; and for its ability to learn behaviors from high-entropy environments. These characteristics make RL a fitting choice for traffic optimization, which is highly complex and behavior-based. While classical control

methods have been effective, their gains tend to level out past some level of complexity, beyond which RL empirically performs better.

Despite RL's potential for developing effective and efficient controllers in the traffic space, it is challenging to actually port and test these controllers in the real world on vehicles. From expensive equipment, scalability issues, to experiencing a "reality gap", a common problem in RL deployment, there are a wide array of challenges to overcome in order to successfully see the results of such traffic-mitigating policies in action.

This thesis ties in the above components, spanning work from simulation to the real world. The goal of this work is to demonstrate the effects and strength of RL in a real world traffic deployment, discussed in Chapters 9 and 10. The chapters prior discuss the work that inform the final road test, including extensive RL algorithmic development, a wide variety of experiments in simulation, miniature real world tests, and high fidelity testing.

1.1 Chapter Overview

Chapter 2 is a background section that discusses prerequisite information, including technical details on reinforcement learning, policy gradient methods, and traffic models. This background will replace most of the backgrounds that are introduced in the individual chapters.

Chapter 3 discusses simulation work done with FLOW, a framework which combines traffic micro-simulation with RL training libraries to enable modular RL training for traffic systems. We go through several examples in this chapter that demonstrates the flexibility, strengths, and challenges associated with using RL in a traffic scenario.

Chapter 4 discusses a particular study of using RL improve traffic shockwaves that are caused by lane changes. A large amount of perturbation in traffic can be traced back to lane changes. Aggressive lane changes, in particular, have an especially pronounced effect on traffic. This chapter explores how an ego vehicle and a connected vehicle can work together to mitigate the effects of lane changes.

Chapter 5 is the beginning of this thesis's exploration of real world deployment. Given that deployment onto real world vehicles is a costly endeavor, this chapter explores the process of deploying RL models onto a scaled-down "mini-city", and analyzes the effects of these algorithms. The "reality gap" of RL is introduced in this chapter, which describes the discrepancy in efficacy between simulation and deployment. This chapter also introduces a technique used to improve the robustness of sim-to-real transfer.

Chapter 6 explores the shortcomings and challenges of deployment further on the same miniature testbed. In overcoming these challenges, this chapter introduces another method of robustifying the algorithm across the sim-to-real transfer. This technique utilizes adversarial RL to strengthen the policy against the portions of the state and actions spaces that are most vulnerable across the sim-to-real transfer.

Chapter 7 begins the last section of this thesis, which revolves around the MegaVanderTest (MVT), the aforementioned 100-car test of RL algorithms. This chapter discusses

the extensive simulation work that was done in developing an RL controller to later be deployed. We go over the design of the simulator, the experiments that were conducted, as well as a results analysis of the candidate controllers.

Chapter 8 is one of two chapters that discusses the results of the MVT. This chapter discusses the end-to-end pipeline of deploying the RL algorithm developed in chapter 7 onto real vehicles on a real highway in Tennessee. It includes a thorough discussion on algorithm development, details on the software to hardware migration, the manner in which deployment was considered practically during development (e.g. reasonable state spaces that correlated to sensors we actually had access to), results across an entire week of testing, and more.

Chapter 9 is the second of two chapters surrounding the results of the MVT. While Chapter 8 focuses on the RL algorithm, this chapter discusses additional components outside RL that affected the controller, including a framework for lagrangian variable speed limits, server organizations, and more.

Lastly, we conclude with a summary of the contributions of this thesis, including:

- Developed robust RL algorithms that are shown to improve efficiency and energy in traffic across a variety methods and scenarios, including roundabouts and highways.
- Developed methods for creating more robust RL algorithms that can withstand their transfer from simulation to the real world. Two that are discussed in this work include noise injection and adversarial training.
- Demonstrated the effectiveness of mixed autonomy traffic as a mode before full autonomy exists, if it does.
- Used RL to demonstrate can be used at scale to improve traffic in simulation and in reality in the I-24 highwya in Nashville.
- Conducted the largest deployment(to our knowledge) of RL-controlled vehicles designed to smooth traffic.

1.2 Disclaimer on Co-Authored and Published Work

The work in this thesis is derived from publications contributed to by multiple authors. All the results and work from collaborations have been included in this thesis, with focus on work that I personally lead, wrote, ran experiments on, and performed data analysis on. Particularly, Chapter 9 is derived from a publication that includes 50+ authors. In Chapter 9, my personal contribution is focused on the controller development, selection, as well as migration onto vehicles. Other work is included for comprehensiveness and readability.

Specifically, work on energy models in Section 9.2 is work by Nour Khoudari and co-authors [69]; the design of the controller architecture in 9.3 is mostly work by Jonathan Lee; the speed planner described in Sections 9.3 and 9.4 is work by Han Wang and co-authors

[157]; the vehicle-side hardware work is done by Matthew Bunting, Matthew Nice, and co-authors [101, 100]. Work on the I-24 MOTION is done by Derek Gloudemans and co-authors [50]. Server-side implementations are also not contributions. The material is presented here for context to explain the work.

In addition, both Chapters 8 and 9 are published work in IEEE Control System Magazine. Their full citations are Reinforcement Learning Based Oscillation Dampening: Scaling up Single-Agent RL algorithms to a 100 AV highway field operational test , and Traffic Control via Connected and Automated Vehicles: An Open-Road Field Experiment with 100 CAVs .

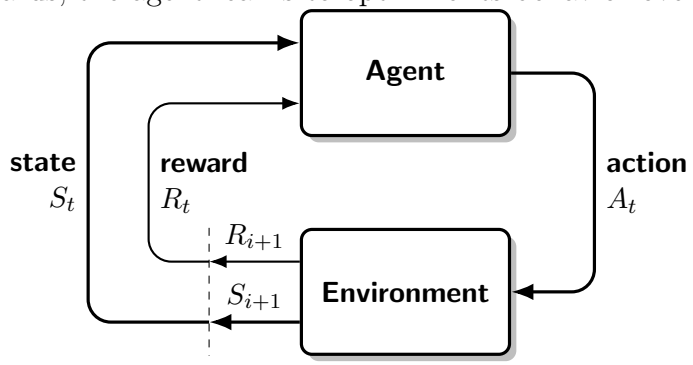
Chapter 2

Background

2.1 Reinforcement Learning

Reinforcement learning has emerged as a powerful paradigm for enabling autonomous systems to learn and optimize their behavior through interaction with their environment. RL offers a unique approach to control system design by incorporating an agent's decision-making process and its impact on the environment. This section provides a concise overview of RL, highlighting its key components and applications in the context of autonomous system control [8].

RL revolves around the concept of an agent learning from its experiences in an environment. The agent takes actions based on its current state and receives feedback in the form of rewards or penalties, which reflect the desirability or undesirability of the agent's actions. By iteratively exploring the environment and adapting its actions based on the received rewards, the agent learns to optimize its behavior over time.



At the core of RL lies the *Markov Decision Process* (MDP) framework, which mathematically formalizes the problem of sequential decision-making under uncertainty. MDPs provide a principled representation of the agent's interaction with the environment, as illustrated in Figure 2.1, encapsulating the states, actions, transition dynamics, and rewards that govern the learning process. Algorithms such as Q-learning [161], SARSA [118], and policy gradient methods [135] leverage MDPs to guide the agent's learning and decision-making

processes. Partially Observable MDPs (POMDPs) are a common variant of MDPs, in which the state is not fully observable. This POMDP can be officially characterized by the tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, T, R, T_0, \gamma, \Omega, \mathcal{O})$, where $\mathcal{S}$ symbolizes the set of states; $\mathcal{A}$ stands for the possible actions; the function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ represents the conditional likelihood of moving to a subsequent state s' given the current state s and the chosen action a; $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function; $T_0 : S \to \mathbb{R}$ is the probability distribution of initial states; and $\gamma \in (0, 1]$ is the discount coefficient applied to the accumulation of rewards. The last two parameters, Ω and $\mathcal{O}$, are included due to the concealed nature of the state: Ω refers to the observations of the hidden state, and $\mathcal{O} : \Omega \times \mathcal{S} \to \mathbb{R}$ stands for the conditional observation probability distribution.

Given a POMDP, one can then sample an initial state $s_0 \sim T_0(\cdot)$ and initial observation of that state $o_0 \sim \mathcal{O}(\cdot \mid s_0)$. The agent, which we denote as a stochastic conditional distribution $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parametrized by θ, samples an action $a_0 \sim \pi_\theta(\cdot \mid s_0)$ from the observation. That action is then used to generate a next state $s_1 \sim T(\cdot \mid s_0, a_0)$ and observation of the state $o_1 \sim \mathcal{O}(\cdot \mid s_1)$, as well as a reward $r_0 = R(s_0, a_0, s_1)$. Iterating this process until termination (which can occur after a fixed time horizon or a certain termination condition) yields a trajectory $\tau = (s_i, o_i, a_i, r_i)_{i \geq 0}$. Let us define the return of a trajectory as the discounted sum of rewards $G(\tau) = \sum_{i \geq 0} \gamma^i r_i$. The goal for the agent is to maximize the expected return over all trajectories, and to learn an optimal policy $\pi^* = \arg\max_\pi \mathbb{E}_{\tau \sim (\pi, \mathcal{P})} G(\tau)$.

The Bellman equation [20], a central element to understanding the structure of the RL problem, is given by the following:

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s')) \tag{2.1}$$

The Bellman equation is used to model the *value function*, which describes the estimated value of a particular state. It operates under a dynamic programming paradigm, meaning that the estimate of the value function is calculated recursively. In practice, RL algorithms leverage other techniques like bootstrapping, function approximation or temporal difference learning such that the value of a state can be estimated without having to calculate all the way to the terminal state.

One of the distinctive features of RL is its ability to handle complex, high-dimensional state and action spaces. Through the utilization of function approximation techniques, such as neural networks, RL algorithms can effectively learn representations of states and policies, enabling the control of systems with large state spaces or continuous actions [40]. Deep RL, which combines deep neural networks with RL, has garnered significant attention in recent years due to its ability to handle complex control tasks, such as autonomous driving and robotics [10].

RL has found diverse applications in the realm of autonomous systems, including automated vehicles, robotics, and game-playing agents. In the context of automated vehicles, RL controllers can learn to navigate complex traffic scenarios, adapt to changing road conditions, and optimize driving strategies based on safety and efficiency objectives [167]. By leveraging

RL, autonomous systems can acquire adaptive and intelligent control policies, making them more capable of handling real-world challenges.

However, RL also presents certain challenges and considerations. The exploration - exploitation trade-off, sample efficiency, and generalization to new environments are areas that continue to be actively researched. Moreover, ensuring the safety, interpretability, and ethical behavior of RL-based controllers remain crucial concerns in their practical deployment.

In conclusion, RL provides a powerful paradigm for developing autonomous system control, enabling agents to learn and optimize their behavior through interaction with the environment. By leveraging the principles of RL, researchers and engineers can advance the capabilities of autonomous systems, paving the way for the realization of intelligent, adaptive, and efficient autonomous systems across various domains.

2.2 Policy Gradient Methods

Policy gradient methods are a foundational class of RL methodologies. By directly optimizing the policy through the gradient of the expected, cumulative, discounted reward with respect to the policy parameters, these algorithms seek the optimal policy $\pi^*(a|s)$, which determines the action a that maximizes the expected reward given a state s. The algorithm updates the policy parameters θ, often represented as the parameters of a neural network in deep RL applications, which ascend the gradient of the expected reward. This process, illustrated in Figure 2.1, is denoted as $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ where α is the learning rate, and $J(\theta)$ represents the expected reward under the policy parameterized by θ.

The gradient of this expected reward with respect to the policy parameters can be distilled into an expectation of the gradient of the log policy multiplied by the cumulative reward (the return). This can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) G_t], \tag{2.2}$$

where $\pi_\theta(a|s)$ represents the policy parameterized by θ, $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ is the return from the time step t, r_t is the reward at time t and $\gamma \in (0, 1]$ is the discount factor. This equation forms the basis of the REINFORCE algorithm [163], a fundamental policy gradient method. Updates in this algorithm occur episodically, with the gradient ascent step typically implemented after each episode. However, more sophisticated policy gradient algorithms, such as Actor-Critic methods [55], offer more frequent updates based on estimates of the expected return, which often accelerates convergence.

More advanced policy gradient algorithms introduce the concept of trust region. These algorithms aim to keep the policy update within a certain "trust region" to ensure stability and prevent harmful policy changes, as well as increase sample efficiency. They optimize a similar objective to the policy gradient methods, but add a constraint to limit the policy update. This leads us to *Proximal Policy Optimization* (PPO) [122], the RL algorithm employed in this work. PPO is a specific implementation of trust region methods that harmoniously combines state-of-the-art performances on standard benchmarks and complex control tasks

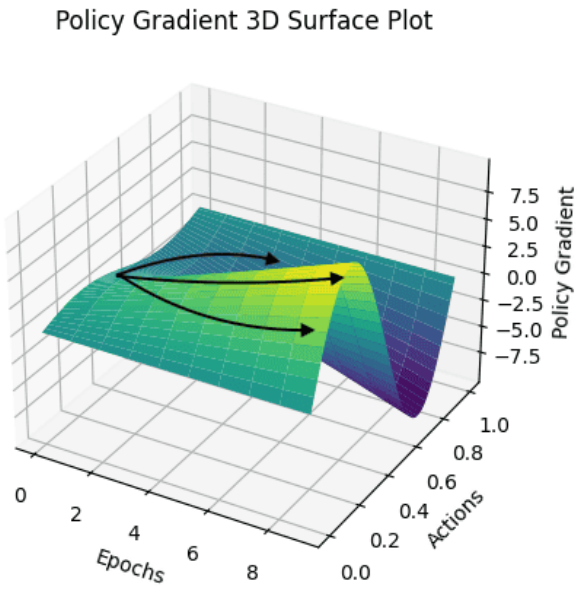


Figure 2.1: Policy Gradient Surface Plot: This 3D surface plot illustrates how policy gradients evolve over epochs and actions, providing insights into the optimization process of a reinforcement learning algorithm. The z-axis represents the magnitude of policy gradients, with colors indicating the strength of gradient values. As training progresses (left to right along the x-axis), the policy gradient landscape changes, guiding the agent's decision-making strategy in reinforcement learning tasks.

with a significant ease in implementation and tuning. PPO aims to compute an update at each step that not only minimizes the cost function but also ensures a modest deviation from the previous policy, optimizing both implementation simplicity and computational efficiency.

2.3 Car Following Models / Intelligent Driver Model (IDM)

In order to train RL controllers, we simulate mixed-autonomy traffic [141] where RL-controlled AVs interact with human-driven vehicles [165, 166, 150]. In order to model human car-following behavior, we use the Intelligent Driver Model (IDM) [68, 143] to govern a vehicle's longitudinal motion according to its leading vehicle. To ensure authenticity, IDM parameters are selected to mimic key non-equilibrium features of real-world traffic, particularly dynamic instabilities (phantom traffic jams) with realistic wave growth and propagating stop-and-go waves caused by human driving behavior [164, 145] or even ACC [58]. To trigger dynamic instabilities that the model possesses for suitably congested densities, zero-mean Gaussian noise is added to the acceleration in each time step. Additional safety measures are included

to maintain acceleration bounds and to avoid collisions in the majority of the experiments in this work—note that pure IDM is mathematically collision-free, but discretized time-stepping and heterogeneous vehicle composition yields potential collision opportunities.

The IDM [143] prescribes the acceleration of vehicle α as a function of its space gap (bumper-to-bumper distance to its leader vehicle) s_α; its speed v_α; and relative speed with the leader, Δv_α:

$$a_{\text{IDM}} = \frac{dv_\alpha}{dt} = a\left[1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right], \tag{2.3}$$

where s^* is the desired space gap and is given by:

$$s^*(v_\alpha, \Delta v_\alpha) = s_0 + \max\left(0, v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}\right), \tag{2.4}$$

where s_0, v_0, T, δ, a, and b are known parameters.

Part I

Early Forays in Deep Reinforcement Learning for Traffic Control

⋆⋆⋆⋆⋆

Part 1 of this thesis discusses some of the fundamental building blocks in exploring the capabilities of reinforcement learning as a tool for traffic optimization. The research in this section paves the way for the latter part of this thesis. Focusing on simulation, we examine use cases and the potential that RL can have in control. We demonstrate the flexibility of such control across a variety of scenarios, including roads, ring roads, and grid networks, and more. We also showcase how connected vehicles and automated vehicles can work cooperatively in order to mitigate the shockwave-inducing characteristics associated with aggressive lane changes.

Chapter 3

FLOW: Connecting Reinforcement Learning with Traffic Optimization

3.1 Introduction

With the advent of autonomous vehicles leading the way for a promising space in the control community, deep RL has emerged as a form of control for future transportation needs. Given the complex, nonlinear dynamics of traffic, deep RL's abilities to capture complicated data makes it a good candidate in the traffic optimization area.

FLOW is a framework that addresses the challenges arising from the rapid growth of automated vehicles in ground traffic [167]. FLOW seamlessly integrates deep RL techniques with the traffic microsimulator SUMO, providing a powerful platform for designing, implementing, and evaluating traffic control strategies. FLOW is open-source and free for download at `https://github.com/flow-project/flow`.

3.2 Background

Reinforcement Learning

We adhere to the conventional reinforcement learning (RL) framework [134, 89], which seeks to optimize the discounted cumulative rewards within a finite time frame for a Partially Observable Markov Decision Process (POMDP) [33]. This is defined by the tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, T, R, T_0, \gamma, \Omega, \mathcal{O})$.

For a formal description and background of RL, refer to Chapter 2.

3.3 Features of FLOW

By leveraging advancements in deep RL, FLOW enables the utilization of RL methods, such as policy gradient, for effective traffic control in the presence of complex nonlinear dynamics.

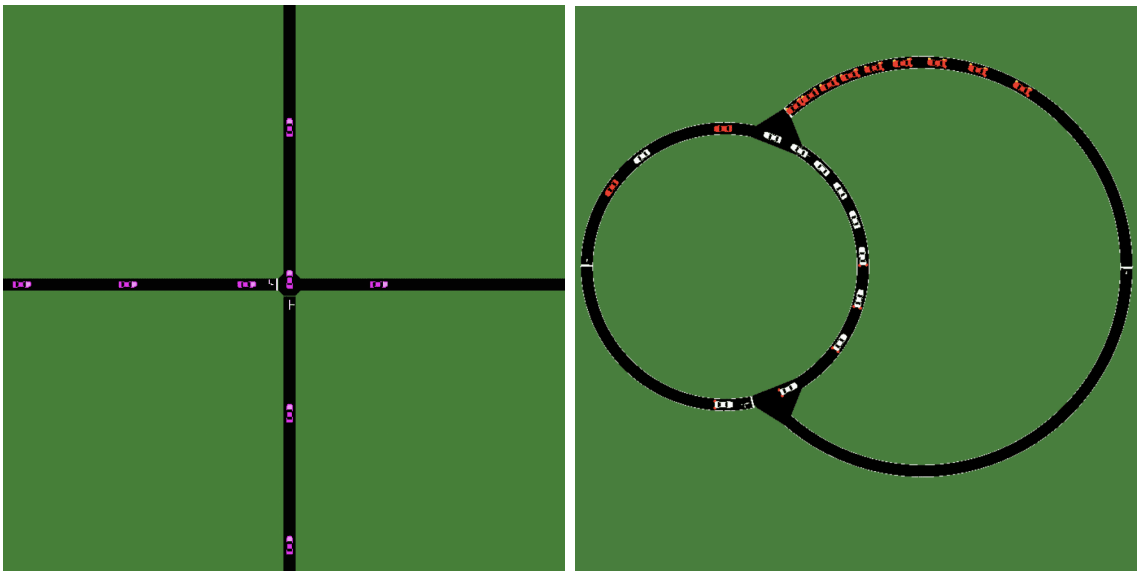


Figure 3.1: Example of network building blocks that be configured and then trained via the FLOW architecture.

It offers a unique opportunity to benchmark the performance of classical controllers, including hand-designed ones, against learned policies or control laws.

FLOW's integration with SUMO and Aimsun as well as a variety of RL libraries such as RLlib and Stable Baselines empowers researchers and practitioners to effortlessly design and simulate a wide range of traffic scenarios. This includes the flexibility to explore different network configurations and vehicle dynamics, allowing for comprehensive evaluation and optimization of traffic control strategies.

FLOW focuses on the challenging problem of controlling mixed-autonomy traffic, involving both automated and human-driven vehicles, in a variety of different traffic settings. FLOW implements some simple scenarios such as an intersection, a ring road, a figure eight, a merge, a bottleneck or a grid available, some of which can be seen in Figure 3.1, while a complex network created from real map data is shown in Figure 3.2. Several benchmarks are available in [150].

3.4 Traffic Control Using Flow

In earlier work, FLOW trains simple neural network policies on a ring road scenario [167]. Through experiments and analysis, they find that while state-of-the-art hand-designed controllers excel within specific conditions, they struggle to generalize to new situations. In contrast, even basic neural network policies trained using FLOW demonstrate remarkable performance in stabilizing traffic across various density settings and exhibit the ability to

Figure 3.2: San Francisco map loaded in FLOW from OpenStreetMap data.

adapt to out-of-distribution scenarios.

Similarly, [73] shows that a low proportion of RL-trained AVs is able to significantly dampen waves in closed and open single-lane networks. Related work trains AVs with a small artificial sensing noise in a roundabout scenario modeled in FLOW and directly transfers the trained policy to a real-world toy network [66]. In another work, authors create a bottleneck model of the San Francisco-Oakland Bay Bridge and significantly improves throughput using a low penetration rate of RL-trained AVs [153].

FLOW highlights the immense potential of RL in advancing traffic control research and its practical application in real-world autonomous vehicle systems. With its ability to seamlessly combine deep RL techniques with traffic micro-simulation, FLOW represents a significant step forward in developing reliable and adaptive controllers for complex traffic environments. By providing a comprehensive and accessible computational framework, FLOW enables researchers and engineers to enhance traffic safety, efficiency, and overall control system performance in the era of automated vehicles.

3.5 Benchmarks

In this work, we focus on a particular scenario: the grid, while drawing comparisons to a number of other examples. The grid is an idealized representation of a city with a grid-like structure such as Manhattan. The purpose of this problem is to highlight issues that arise in coordination of traffic light control, particularly questions of partial observability and the scaling of RL algorithms with action dimension. Solutions to this problem will generate new traffic light control schemes that minimize the average per-vehicle delay while inducing some measure of fairness.

Vehicles enter at the corners of the grid. For simplicity of the problem, vehicles trael straight on their path. Each intersection has a traffic light that allows vehicles to flow either horizontally or vertically. If the light is green, it transitions to yellow for two seconds before switching to red for the purpose of safety.

The gym environment has the following state space, action space, and reward:

- States: Speed, distance to intersection, and edge number of each vehicle. The edges of the grid are uniquely numbered so the travel direction can be inferred. For the traffic lighs we return 0, 1 corresponding to green or red for each light, a number between $[0, t_{switch}]$ indicating how long until a switch can occur, and 0, 1 indicating if the light is currently yellow. Finally, we return the average density and velocity of each edge.
- Actions: A list of numbers $a = [-1, 1]^n$ where n is the number of traffic lights. If $a_i > 0$ for traffic light i, it switches; otherwise no action is taken.
- Reward: The reward present in is once again used for this benchmark. Here, the use of the L2-norm induces fairness, as a more equal distribution of speeds produces a larger reward.

3.6 Experiments

In this section, we present preliminary results from running four reinforcement learning algorithms on the grid. We compare the performance of the grid with three other benchmarks, which are not in the scope of this work: a figure eight scenario, a merge scenario, and a bottleneck scenario. For a more detailed description of these experiments, please refer to [150].

Candidate controllers

Experiments were conducted using gradient-based algorithms *Trust Region Policy Optimization* (TRPO) [123] and *Proximal Policy Optimization* (PPO) [122] as well as the gradient-free method *Evolutionary Strategies* (ES) [120] and an implementation of *Augmented Random Search* (ARS) [95]. For ARS a linear policy is used. For ES we use a deterministic MLP with hidden layers $(100, 50, 25)$ and tanh non-linearity whereas for PPO and TRPO the MLP is diagonal Gaussian. Moreover, in the PPO algorithm, a [256, 256] MLP with tanh

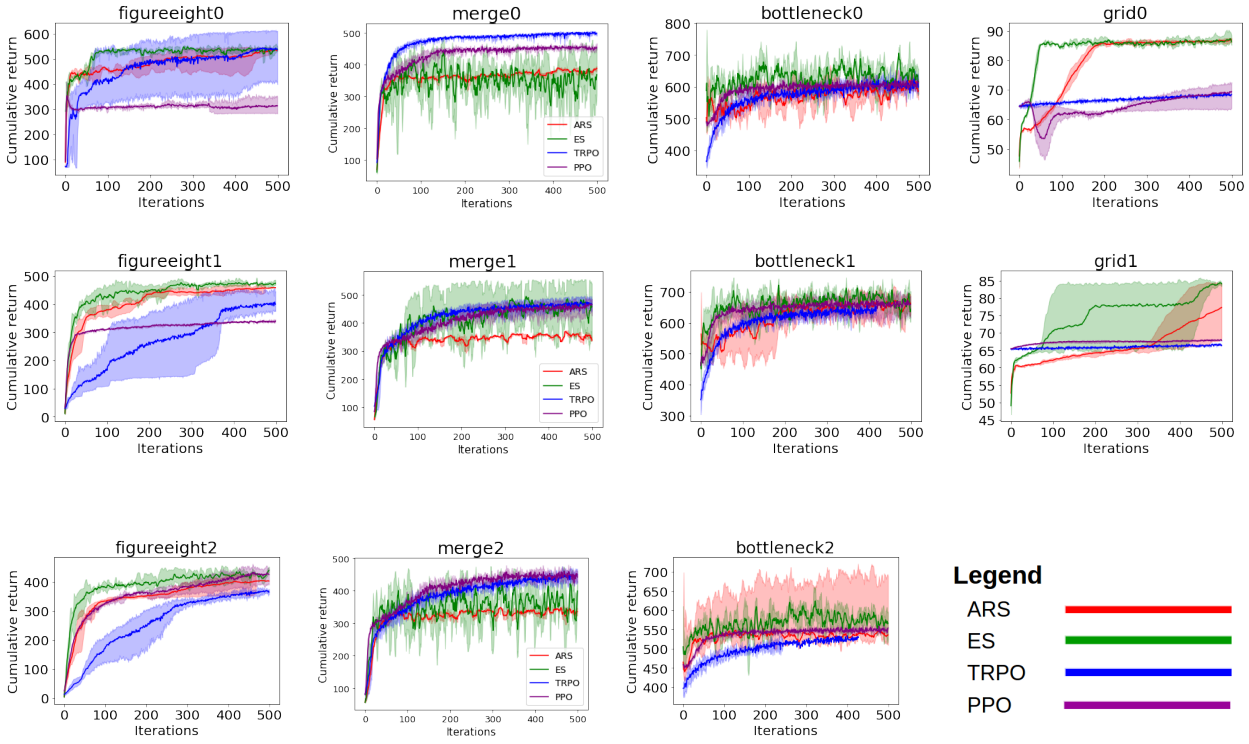


Figure 3.3: Learning curves for each benchmark. A iteration of training in each algorithm consists of 50 rollouts/trajectories. The reported cumulative rewards correspond to undiscounted returns.

non-linearity is used to compute a value function baseline, whereas a linear feature baseline is used for the TRPO algorithm. Results are reported over three random seeds to account for stochasticity and to test training stability. Outside of the reported hyperparamters, we use the default parameters set in rllab and rllib as per the commits specified in section 3.6.

Reproducibility

In the spirit of reproducibility, the controllers used to generate the following results are stored as .pkl files and tensorflow checkpoints at `s3://public.flow.results/corl_exps/exps_final`. We have frozen the code used to generate the results as Flow 0.3.0 and commit number "bc44b21". The commit number of SUMO, available at `https://github.com/eclipse/sumo` used to run the results is "1d4338ab80". The version of RLlib used to run the code is available at `https://github.com/eugenevinitsky/ray` at commit "0f45f80".

Table 3.1: Average optimal return and standard deviation based on performance metrics after 500 training iterations; average is over 40 rollouts. "–" indicates that the experiment did not have a complete number of seeds and thus was not reported.

Benchmark	ARS	ES	TRPO	PPO	Human
Figure Eight 0	7.31 ± 0.54	6.87 ± 0.08	8.26 ± 0.10	–	4.18 ± 0.09
Figure Eight 1	6.43 ± 0.01	–	5.61 ± 0.55	–	4.18 ± 0.09
Figure Eight 2	5.70 ± 0.01	5.96 ± 0.00	5.03 ± 0.23	–	4.18 ± 0.09
Merge 0	11.3 ± 0.31	13.31 ± 0.54	14.95 ± 0.12	13.66 ± 0.40	7.39 ± 0.55
Merge 1	11.06 ± 0.32	17.29 ± 0.40	13.74 ± 0.23	14.61 ± 0.47	7.39 ± 0.55
Merge 2	11.5 ± 0.54	17.36 ± 0.48	14.14 ± 0.24	14.54 ± 0.31	7.39 ± 0.55
Grid 0	270.2 ± 0.2	271.7 ± 0.6	296.2 ± 2.5	296.8 ± 5.3	280.8 ± 1.5
Grid 1	274.7 ± 1.0	274.3 ± 1.1	296.0 ± 2.0	296.2 ± 2.0	276.8 ± 1.7
Bottleneck 0	1265 ± 263	1360 ± 200	1298 ± 268	1167 ± 264	1023 ± 263
Bottleneck 1	1350 ± 162	1378 ± 192	1375 ± 61	1258 ± 200	1135 ± 319
Bottleneck 2	2284 ± 231	2324 ± 264	2131 ± 190	2143 ± 208	1889 ± 252

Algorithm performance

Fig. 3.3 presents the training performance of each of the algorithms presented in sec. 3.6. The results suggest that gradient-based algorithms (TRPO and PPO) are more effective in merge scenarios, while gradient-free algorithms (ES) are better in figure eight, grid, and bottleneck benchmarks. This is likely due to the merge network reward providing richer feedback with regards to which states are good/bad, thereby assisting gradient-based algorithms in computing meaningful gradients. On the other hand, in environments populated with many local extrema associated with poor rewards, more exploration-oriented algorithms (ARS or ES) are less prone to being trapped in local optima.

Controller performance

We highlight the effectiveness of each learned policy on improving traffic performance. In order to do so in terms of understandable values that are not cluttered by the reward design process, simple scenario-specific performance metrics are defined for each benchmark. These metrics are:

- Figure eight: Average speed of vehicles in the network (m/s).
- Merge: Average speed of vehicles in the network (m/s).
- Grid: Average delay of all vehicles in the network (s) relative to traveling at the speed limit. In this case, smaller values are preferable.
- Bottleneck: Outflow over the last 500 seconds of a 1000 second rollout (veh/hr).

In order to get a sense of how well these policies are performing, we use an estimate of human-level performance acquired by running simulations of each environment in the absence of learning agents as a baseline. Note, for the grid we use the actuated traffic light

baseline available in SUMO with the default parameters for minimum duration of green and red time; however, we set the yellow light to imitate our controller and switch every two seconds. Table 3.1 highlights the performance of each algorithm within the context of the above mentioned performance metrics. As we can see, the learned policy outperform human-level dynamics in all task.

3.7 Conclusion

The existing performance of various RL algorithms comes with numerous unresolved questions for previously established benchmarks. Our analysis suggests that the best solutions, particularly in terms of traffic speed and delays, are yet to be discovered, and that current existing benchmarks are not yet optimal. For instance, in a simple model like the grid, connected autonomous vehicles (CAVs) are incapable of coordinating platoons with traffic light phases due to a lack of technology. This observation is reinforced by the discovery of a potentially superior control strategy through reward shaping and design, indicating an exploration challenge within the RL framework.

Furthermore, RL in transportation systems shares challenges with video game environments, including reward sparsity and state space redundancies. The possibility for dimensional reduction through symmetry in networks, such as in the grid and bottleneck scenarios, presents a promising area for exploration. Hierarchical models and methods that consider segmenting the scenario could also enhance the performance of RL applications in complex network scenarios.

Additionally, further directions to consider include issues of fairness, decentralization, generalization across different network conditions, and improving sample efficiency, which is critical for scaling RL applications to real-world transportation networks. The ongoing challenges with large-scale benchmarks like the grid underscore the need for innovative solutions to manage extensive action and state spaces effectively. The release of these benchmarks is intended to drive further research and innovation in this field.

Chapter 4

Cooperative Reinforcement Learning-based Damping of Lane-Change-Induced Waves

4.1 Introduction

Automation in transportation has a unique role in the field of machine learning-based applications due to the sheer reach of the audience it can reach on a regular and tangible basis. Autonomous vehicles are rapidly making their way into the hands of consumers and the public sphere. As of 2022, more than 1,400 self-driving cars, trucks, and other vehicles were in testing by more than 80 companies across 36 U.S. states and Washington DC [44]. Vehicles with level 2 automation capability made up 30% of all new cars sold in the U.S. in 2020, or 11.2 million units in total sales, a 91% increase from the previous year [81].

As automation increasingly hits roadways, the impact of machine learning methods grows correspondingly more profound. Emergent behaviors and significant breakthroughs in machine learning such as the creation of models capable of mastering Starcraft [154] and Go [127] lead the way for learning for autonomous vehicles. Beyond learning how to drive, this article is interested in RL's ability to leverage its knowledge of the environment and human driving behavior in order to not just contribute to, but improve the traffic it is a part of. There is plenty to solve: vision-based methods, traffic light phases [162], and traffic string instability [58]. Stop-and-go waves, also described as traffic shockwaves or oscillations, can form freely without bottlenecks [132]. We use these terms interchangeably in this article.

There has been a rich amount of work in developing systems to mitigate stop-and-go waves under a variety of names: "wave damping or mitigation", "stabilizing traffic flow", "Jam-absorption driving (JAD)", "speed harmonization with vehicles", "dissipating stop-and-go waves" or "vehicle speed limit control". Beatty et al. demonstrated in 1998 that simply maintaining a sufficient gap from the preceding vehicle can play a significant role in wave damping [17]. Since Beatty's work, numerous studies have shown how the control of

longitudinal movements can damp waves and improve performance in a variety of metrics, such as energy consumption, safety, and comfort [38, 85, 112, 30].

Several efforts have been studied to approach the problem of wave damping from a classical optimization perspective [93]. Since stop-and-go waves can occur in a wide variety of scenarios, wave damping studies are often designed for a specific scenario, such as traffic intersections [80, 113, 92] or merge points [115, 106, 178]. Connected, automated, vehicle (CAV) intervention can be applied via longitudinal events like braking events or acceleration, or can exploit the geometry of the road to achieve the same effects. While classical approaches have been the driver for significant results in the space of stop-and-go mitigation and wave damping in the past, the complexity of traffic systems make this a difficult problem to solve.

Meanwhile, deep RL has emerged as an effective means of control for traffic-based systems. RL's ability to capture rich, complex, data-driven behavior, combined with deep learning's ability for flexibility and breadth, makes it an apt tool for traffic scenarios. RL has been demonstrated to be an effective tool in the traffic domain as a controller for vehicles [167, 152, 73, 111]. Its applications are varied: RL has challenged classical control in a variety of scenarios, including highways, bottlenecks, traffic intersections; exhibiting its ability to mitigate traffic shockwaves.

Cooperative lane change maneuvers have been extensively studied in the literature [133, 99]. These studies aim to establish cooperation between an ego vehicle and CAV(s) in the target lane to create a gap and facilitate efficient, safe, and comfortable lane change actions. This article is focused on analyzing the impacts of lane-changing on target lane stability and providing RL methods to mitigate those instabilities, in particular when cooperative vehicles can provide richer information. This paper is not exploring a cooperative lane change action, but rather investigates cooperative maneuvers to mitigate lane-change induced shockwaves.

The contributions of this article are:

- Demonstration and study of how a single lane change can cause significant traffic-destabilizing effects in single-lane traffic.
- The introduction of a single-agent RL policy applied to a CAV, designed to damp the waves caused by a connected, lane-changing ego vehicle.
- Numerical results over a number of inflows evaluating the RL-enabled CAV's ability to damp waves.

The remainder of this article is organized as follows. In Section 6.2, we provide background information on RL, car following models, and the simulator. In Section 6.3, we introduce the problem formulation, including details on the traffic scenario, RL structure, and simulation parameters. In Section 4.4, we present and discuss the simulation results. Finally, we draw concluding remarks in Section 6.6.

4.2 Background

Deep Reinforcement Learning

We adhere to the conventional reinforcement learning (RL) framework [134, 89], which seeks to optimize the discounted cumulative rewards within a finite time frame for a Partially Observable Markov Decision Process (POMDP) [33]. This is defined by the tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, T, R, T_0, \gamma, \Omega, \mathcal{O})$.

For a formal description and background of RL, refer to Chapter 2.

Policy Gradient Algorithms

We use *Proximal Policy Optimization* (PPO) [122] with a multilayer perceptron policy in this article. For a formal description and background of policy gradient algorithms, refer to Chapter 2.

Car Following Models

We use the Intelligent Driver Model (IDM) [143] to model the dynamics of human drivers.

We describe the IDM parameters and the values used in our simulation in Section 6.3 and Table 8.2. For a formal description and background of car following algorithms, refer to Chapter 2.

Highway Simulation

For the training of the RL policies in this article, we used a private Toyota-developed simulation framework, compatible with RL training frameworks such as Stable Baselines 3 [62], which was used for training, and h-baselines [74]. It provides a convenient interface for developing traffic scenarios such as highways, multi-lane roads, and on-and-off ramps. A variety of driver models such as IDM, FollowerStopper [112], and also lane-changing models are built into the repository. All the environments developed with this simulation framework are compatible with OpenAI Gym [26]. As such, support for most RL algorithms is readily available. For further information on this and related work, we refer to readers to view the article listed here: [72].

4.3 Problem Formulation

In this section, we define the problem we are attempting to solve, as well as discuss details of the RL formulation, traffic network configuration, and simulation details.

We demonstrate the effectiveness of cooperative, RL-enabled CAVs in traffic smoothing and wave damping by examining a particular case: the effects of lane-changing on traffic. In certain inflows and densities of traffic, lane changes can cause mild to severe perturbations

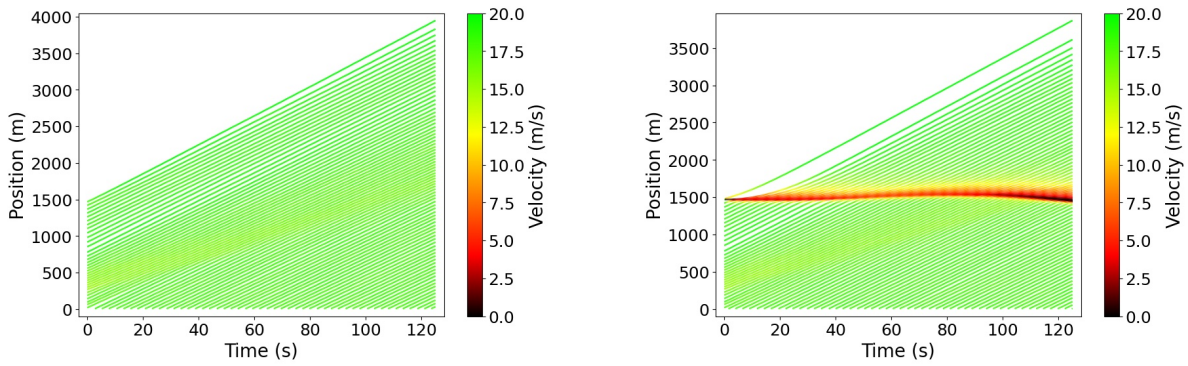


Figure 4.1: Left: Time Space Diagram of vehicles in one-lane free flowing traffic. Right: Time Space Diagram of vehicles after a lane change.

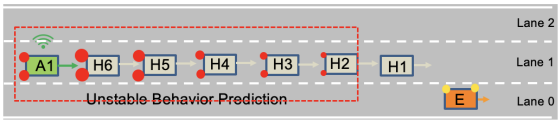


Figure 4.2: A diagram of the experimental setup. An ego vehicle, E, lane changes from lane 0 to lane 1 ahead of the main platoon of 5 human-driven vehicles (H_1-H_6) and a CAV (A_1). E is connected to $A1$.

in the state of traffic, causing oscillations and waves to form. This is affected by a number of factors, such as the distance to the follower vehicle in the target lane, the density of vehicles in the target lane, or the speeds of the vehicles.

As shown in Fig. 4.1, the occurrence of a lane change can result in worsening of traffic conditions. On the left of Fig. 4.1, observe a time space diagram depicting free-flowing traffic at 1400 vehicles per hour. On the right of Fig. 4.1, the identical scenario is affected by a lane change at the first time step, resulting in significant waves that propagate and deepen at 120 seconds into the simulation. Other work has depicted the effects of lane changing in terms of energy savings. Zheng et al. shows that the wavelet-based energy readings of vehicles increase by 3.6 % after a lane change [179]. In our baseline scenario, as portrayed in Fig. 4.2, a connected, ego vehicle lane changes into the mainline lane of vehicles in lane 1, causing unstable behavior. In intelligent transport systems, an ego vehicle refers to a vehicle equipped with sensing capabilities that can observe its surrounding environment.

Solving this worsening of traffic, depicted in this case as a traffic oscillation or wave, is the goal of our RL agent. In the context of the problem, the CAV and the ego vehicle are working together, cooperating to mitigate the effects of the lane change. Having access to information from the ego vehicle is valuable, as it gives the CAV additional time to prepare before it hits the lane change-induced slowdown, as well as gives the CAV important insight

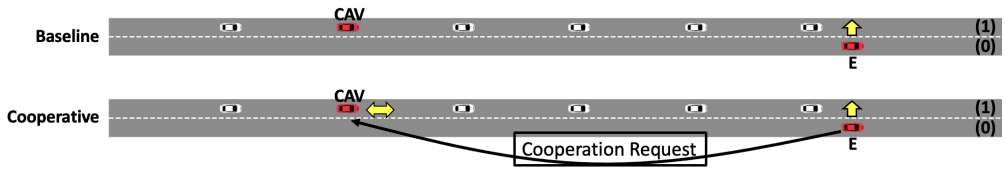


Figure 4.3: Experiment setup: a) Baseline scenario where E changes lane and CAV follows an IDM acceleration profile, b) the cooperative scenario where E changes lane and CAV follows the the Deep RL policy to mitigate the disturbance.

on downstream conditions, including velocities and densities of vehicles. In this article, we assume an unlimited sensing range between the CAV and ego vehicle.

Network Configuration

In all of the experiments presented, simulation is done on a two-lane, two-kilometer highway over a length of 4000 meters, depicted in Fig. 4.3. The ego vehicle lane changes to the front of the platoon of vehicles in lane 1. The CAV is placed 5 vehicles behind the ego vehicle, which begins the simulation traveling at a velocity of 10 m/s and lane changes in at 11 meters ahead of lane 1's leader vehicle. Behind the CAV, a constant inflow of vehicles (1400 veh/hr) is being supplied. The highway includes three types of vehicles:

1. Ego vehicle: Changes from lane 0 to lane 1, shown as E in Fig 4.2.
2. CAV: In lane 1, follows an RL policy, shown as A_1 in Fig 4.2.
3. Human-driven vehicles: Both lanes, unconnected, follows IDM controller, shown as H_{1-6} in Fig 4.2.

The traffic state and inflow settings for each lane are initiated as follows:

- **Lane 0**: The ego vehicle is placed at a distance $x_{\mathrm{E}}(0)$ from the beginning of the highway section. The magnitude of the lane change-induced disturbance can be changed by adjusting the ego vehicle's parameters, such as the initial velocity $v_{\mathrm{E}}(0)$ or position $x_{\mathrm{E}}(0)$. A constant flow of vehicles populates the lane after the ego vehicle.
- **Lane 1**: Initially, the lane is populated with 5 human vehicles, followed by a CAV. For each traffic flow rate Q, we obtain equilibrium traffic state (e.g., gaps and speeds of the vehicles) through a warm up period and store them as initial states. We clip vehicles in lane 1 at point $x_{\mathrm{E}}(0)$, since the vehicles traveling in front of the ego vehicle will not be affected by the lane-change action. Beyond the initial states, we prescribe a constant flow of vehicles to this lane.

We consider two scenarios:

- **Baseline**: At the first simulation step, vehicle E changes lanes from lane 0 to lane 1. The lane-change maneuver disturbs the traffic in the target lane (lane 1). The CAV and human-driven vehicles follow acceleration profiles generated by the IDM car-following model. Further lane-changes are not allowed.
- **Cooperative**: Like the baseline scenario, E changes lanes; however, the CAV follows acceleration instructions generated by the RL policy. Human-driven vehicles follow IDM acceleration profiles. Like the baseline, further lane-change maneuvers are not allowed.

Reinforcement Learning Structure

In this section, we discuss the details and parameters of the RL problem, including the action space, observation space, and reward function.

Action Space

The actions are applied from a 1-dimensional vector of length 1, which is used to control the acceleration of the CAV at any given timestep. Upon evaluation time, all accelerations are wrapped in a failsafe that prevents vehicles from executing an acceleration that would result in a crash. The acceleration space is bounded between $[-3, 3]$ to mimic reasonable safe acceleration and deceleration bounds.

Observation Space

We will reference the naming scheme from Fig. 4.2 in order to describe the observation space for the remainder of the article. The observation space of the CAV, A_1 is listed below and conveys the following information about the environment to the agent. In addition, the timestep before the ego vehicle E lane changes into lane 1, it communicates the first five bullets of the below observation space to the CAV:

- Intent to lane change as a boolean.
- Longitudinal distance to its future follower vehicle, H_1.
- Longitudinal distance to the CAV, A_1.
- Speed of ego vehicle, v_E.
- Speed of future follower vehicle, v_{h_1}.
- Speed of CAV, v_E.
- Headway of CAV, H_E.
- Density of vehicles between A_1 and ego vehicle (E).

All of the elements of the observation space are normalized by dividing against the maximum value of each state. The observation space was designed such that the observation space values could be replicated in real world evaluation. With modern sensing capabilities, all of these could be collected with sensors like Lidar, radar, or cameras.

Reward Function

The reward function is a velocity-based function with a variety of penalties designed to coax smoothing.

$$r_t = \frac{\sum_{i=0}^{p} v_i}{p} - \text{pen}_a - \text{pen}_{\text{drac}} - 2(\text{pen}_{lh} \mid \text{pen}_{sh} \mid \text{pen}_{zero}) \tag{4.1}$$

The main component of this reward function is the average velocity of the first p vehicles in the system. We use p to denote the size of a *platoon* of vehicles. The choice of measuring the average velocity of a smaller platoon of vehicles as opposed to the entire system speed is to focus the agent's efforts on the vehicles which are most affected by the lane change. Too small of a platoon size would cause the agent to optimize greedily rather than for other vehicles. While further vehicles are also affected, it aids training to localize the reward to a platoon. For these experiments, the platoon size is 20, or $p = 20$.

In this reward function, pen_a is a penalty on square acceleration, which discourages overly high accelerations: $c_a * a^2$, where c_a is a coefficient. pen_{drac} is a penalty on the Deceleration Rate to Avoid a Crash (DRAC) [156, 94]. This is a safety index in traffic oscillation scenarios which measures the minimum deceleration required to avoid a collision, defined as:

$$DRAC = \begin{cases} \frac{(V_2 - V_1)^2}{2D_{1-2}} & \text{if } V_2 > V_1 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

, where V_2 is the follower vehicle, V_1 is the leader vehicle, and D_{1-2} is the follower vehicle's headway. pen_{lh} is a penalty on large headways that is set to 1 if $h_E > 200$. pen_{sh} is a penalty on small headways that is set to 1 if $h_E < 2$. Lastly, pen_{zero} discourages standstill behavior by being set to 1 for small accelerations, $a < 0.1$.

Algorithm/Simulation Details

We ran PPO experiments with a discount factor of 0.99, a GAE lambda value of 0.99, a mini-batch size of 64, a learning rate of $3e-4$, a step size of 1536, and trained over iterations up to 6000. The controller is a neural network with a network architecture of $(64, 64)$ and a tanh non-linearity activation function.

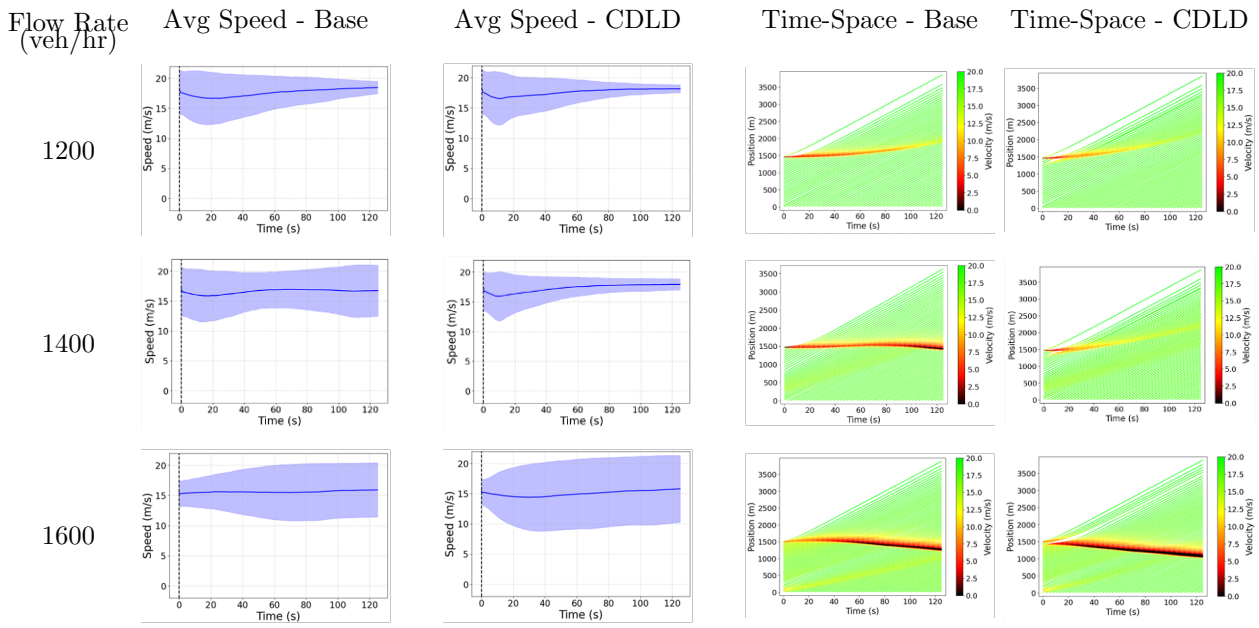


Figure 4.4: Simulation results for both baseline and cooperative (implementing RL) scenarios: a) average velocity plots show both mean value (solid blue curve) as well as the standard deviation (blue shading), b) Time-Space plots are colored by velocity (m/s). The disturbance occurs at time $t = 1$ s and position $x_H(0) = 1500$ m.

Metrics	**1200**			**1400**			**1600**		
	Base	Coop.	%	Base	Coop.	%	Base	Coop.	%
Throughput	41	41	0	41.4	48	15.9	46.7	43.6	-6.6
Average Velocity (m/s)	17.17	17.67	2.9	16.12	16.97	5.3	14.87	14.04	-5.6
TTC (s)	8.02	10.45	30.3	5.96	10.67	79.0	6.96	5.88	-15.5
Energy Consumption (kJ/kg)	1.87	1.73	-7.5	1.79	1.64	-8.4	1.70	1.69	-0.6

Table 4.1: Throughput (veh/hr) past 1600m on the highway, Average Velocity (m/s), and Energy (kJ/kg) for the simulation scenarios corresponding to Figure 4.4. Results for the base case and the cooperative case are presented.

	Intelligent Driver Model						
Parameter	v_0	T	a	b	δ	s_0	ϵ
Value	20	1.2	0.7	4.0	4	2	$\mathcal{N}(0, 0.2)$

Table 4.2: Selected IDM parameters

Simulation Parameters

We simulate three traffic inflow rates $Q = 1200$, 1400, and 1600 vehicles per hour per lane. For the initial state, we place the ego vehicle at a gap distance $s0 = 11$ m from the leading vehicle in lane 2 (see Figure 4.3) and we initialize the ego vehicle's speed at $v_{\mathrm{E}}(0) = 10$ m/s.

The simulation step size is 0.25 seconds. The horizon is 500 timesteps, or 125 seconds. The car-following parameters used in the simulation are listed in Table 8.2.

4.4 Experimental Results

Metrics

We use the following metrics to evaluate the performance of the proposed controller:

- **Energy**: We use power request as a proxy for fuel consumption. The power request for each vehicle at each time step is [84]:

$$P := \begin{cases} m_{eff}\dot{x}(\ddot{x} + f(\dot{x})) & \text{if } \ddot{x} > 0 \\ 0 & \text{if } \ddot{x} \leq 0 \end{cases} \tag{4.3}$$

  where m_{eff} is the effective mass of a vehicle, $\dot{x}$ is the speed of the vehicle, $\ddot{x}$ is the acceleration of the vehicle, and $f(\dot{x})$ is the drag force. For simplicity, we assume that $f(\dot{x}) = 0$ and we consider unit mass as $m_{eff} = 1$. We estimate the normalized consumed energy by integrating the power request over the simulation time for all vehicles.

- **Time-to-Collision (TTC)**: We use TTC as a surrogate measure for safety to evaluate the controller's impact on traffic safety. We use the following definition:

$$TTC := \frac{x_{i-1} - x_i - l_{i-1}}{\dot{x}_i - \dot{x}_{i-1}} \qquad \text{if } \dot{x}_i > \dot{x}_{i-1} \tag{4.4}$$

  where $x_{i-1} - x_i - l_{i-1}$ is the space gap between vehicle i and its immediate predecessor, and $\dot{x}_i - \dot{x}_{i-1}$ is the speed difference between vehicle i and its predecessor.

- **Throughput**: We compute the total number of vehicles that reach 1600 meters in the network during the simulation period.

- **Average Speed**: Average speed of all vehicles during the simulation time period.

Results

The results indicate that the simulated traffic has a metastable condition. In metastable traffic regime, there is a density range ($\rho_{c1} \leq \rho \leq \rho_{c2}$) in which small perturbations disappear while perturbations of sufficiently large amplitude grow [144]. In Figure 4.4, the time space diagrams show that at low traffic flow rates ($Q = 1200$), the lane-change perturbation disappears in the base scenario. However, at higher flow rates ($Q = 1400$ and $Q = 1600$), the perturbation leads to traffic breakdown.

Overall, the RL policy performs well in all metrics across the $Q = [1200, 1400]$ cases, increasing throughput, velocity, and TTC; and decreasing energy consumption. This is depicted numerically in Table 4.1, which includes data from $Q = [1200, 1400, 1600]$. Each value is averaged over 30 evaluation runs. The RL policy performs the best in the $Q = 1400$ scenario with which it was trained, yielding a 5.3% increase in average velocity and a 15.9% increase in throughput, shown in Table 4.1.

The wave damping effects are shown clearly on the time space diagrams on the right of Fig. 4.4, which shows a complete mitigation of the wave by 120 seconds in, such that the vehicles are in free flow. Meanwhile, the baseline has not only failed to clear up, but has in fact worsened by the same time, featuring a considerable traffic jam at around 1500 meters. Qualitatively, the RL agent learns to create a gap at the moment of lane change indication, and slowly closes the gap, allowing for a less dramatic slowdown for the platoon and other vehicles approaching the slowdown. This in turn prevents other vehicles from executing any major braking events that could further propagate the wave backward. The RL policy also improves upon the $Q = 1200$ case in every measure except throughput (for which it maintains performance), yielding a 2.9% increase in velocity.

The TTC values for both $Q = [1200, 1400]$ increase drastically. TTC is a good proxy for driver safety, so this drastic increase indicates that the performance of the CAV allows for vehicles and drivers to be in safer positions, with improved time buffers on potential collisions.

The average velocity plots from the left of Fig. 4.4, depicting mean velocity as well as its standard deviation, also confirm the wave damping behavior: a small dip in speed in the RL-enabled average velocity plots shows the CAV slowing down to create a gap. At this point, the standard deviation of the average velocity widens briefly because the CAV in fact creates minor waves in order to mitigate the larger wave. Then, the standard deviation of the average velocity narrows out significantly compared to that of the baseline, indicating that vehicles are traveling close to the same speed, a marker of free flow traffic. The higher standard deviations indicate the presence of waves, given that varying vehicles have very low or high average velocity.

Our current iteration of the RL policy underperforms for the $Q = 1600$ scenario, however. While this could be due to the increased "difficulty" of the problem with higher inflows and

therefore deeper waves, we are interested to evaluate the performance for a policy trained explicitly for $Q = 1600$ or for a policy exposed to a variety of different Q.

Discussion

Cooperative RL has shown to be an effective method for wave damping caused by lane changes. While high performance is evident for the $Q = 1400$ scenario and improvements are also clear to see in the $Q = 1200$ case, future work should focus on robustness of the policy. While $Q = [1200, 1400]$ has not reached a critical bottleneck, the qualitative behavior that the RL policy has learned to exhibit happens to work for both. For $Q = 1600$, however, which is given to a deeper wave and traffic jam, it is likely that the same emergent behavior that worked for $Q = [1200, 1400]$ does not in fact work for $Q = 1600$.

Interestingly, the time space diagrams generated for RL policies in Fig 4.4 between $Q = 1200$ and $Q = 1400$ are quite similar in appearance. This would indicate that, at least at the local maxima of this policy, there is a certain minimal amount of time that it takes to damp a wave; "rushing" wave damping or mitigation may not be an option.

4.5 Conclusion

In this article, we studied the ability of RL to mitigate wave propagation caused by lane changing. Specifically, we are interested in the scenario where lane changes can be anticipated via vehicle-to-vehicle communication between the lane-changing ego vehicle and the CAV that is being trained to respond to the lane change. We developed an RL-based controller for the CAV responding to the lane change. In evaluation, we exhibited how the RL-enabled CAV is capable of significantly increasing overall system velocity and throughput. In addition, it is also effective in wave mitigation in never-before-seen lower inflow rate scenarios.

As we continue to research RL-based methods in wave damping, some directions that we hope to explore include:

- Increasing the number of single-agent CAVs involved in wave damping on the mainline.
- Multi-agent solutions across multiple CAVs.
- Training across different inflows and traffic parameters (e.g. initial speeds, number of vehicles between the ego vehicle and the CAV) to improve performance over different scenarios.
- Examining and improving the robustness of the policy.
- Considering the impact of the position of the CAV on wave dampening ability.

Another potential direction for future research in this area would be to consider a more general approach than the one we have presented. For example, considering a general mixed autonomy highway setting in which a certain percentage of vehicles are CAVs, a lane-changing ego vehicle should be able to signal to any of the CAVs within reasonable range to mitigate a potential disturbance. There could be multi-agent approaches to this, where CAVs cooperate to do this, or a centralized controller could learn an algorithm for ranking which CAV should be assigned to damp the disturbance.

ACKNOWLEDGMENT

The authors would also like to thank Aboudy Kredieh for their contributions to the code base.

In this section, we examined some particular use cases of the extents of reinforcement learning in traffic optimization. From control-enabled AVs to traffic infrastructure components such as traffic lights, a rich tapestry exists for RL to reside on. With real-world challenges in mind, 4 demonstrates how RL can significantly mitigate lane-change induced shockwaves in highway traffic, ranging from complete jam absorption to mild to moderate mitigation. While the work in simulation is promising, the aim of this work is to test the efficacy of RL on the real world. With the foundation in simulation set, the next part of this thesis aims to explore the deployment of RL algorithms on toy scenarios. We will also discuss the "reality gap" that is common with RL algorithms, as well as explores techniques for creating more robust algorithms that can survive the transfer from simulation to reality.

⋆ ⋆ ⋆ ⋆ ⋆

Part II

sim2real: A Study in Robustness in Moving Reinforcement Learning Algorithms from Simulation to the Real World

⋆ ⋆ ⋆ ⋆ ⋆

With the aim of this work being an eventual test of reinforcement learning algorithms in the real world, this part seeks to bridge the gap between simulation and the final 100-vehicle deployment. This area, sometimes referred to as "sim2real" describes the class of problems associated with transferring a policy from simulation to the real world. Here, we are introduced to the "reality gap", a common problem in RL that describes the discrepancies between the model that is trained in simulation, which tends to be constructed under the more perfect conditions of simulation; and deployment, to which noise and previously unmodeled circumstances are common. This deterioration of algorithms is noted in many robotics-adjacent RL tasks. As a first step in developing techniques in creating more robust algorithms, we test our deployment on a scaled-down miniature city, consisting of all the standard fixtures of traffic system, from roads, to houses, to pedestrians. Chapter 5 introduces a technique for improving the robustness of policy transfer via a method called "noise injection". Chapter 6 improves the noise injection method with an adversarial method. In both chapters, we analyze in real-time the effects of deployment, in this scaled-down precursor to a real-car test.

Chapter 5

Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles

5.1 Introduction

Control of mixed-autonomy traffic: Transportation is a major source of US energy consumption and greenhouse gas emissions, accounting for 28% and 26% respectively. According to the bureau of transportation statistics, total road miles traveled is continuously increasing, growing at 2 to 3% per year between 2010 and 2014 while over the same period the total road length of the US transportation network remained unchanged. The increased road usage is coupled with an increase in congestion. Overall congestion delay in 2014 was 6.9 billion hours, an increase of 33% since 2000; the problem is even worse in metropolitan areas where travelers needed to allocate an additional 150% more travel time during peak periods to arrive on time. The congestion also has significant economic cost, totaling 160 billion dollars in 2014 [39]. Depending on their usage, automated vehicles have the potential to alleviate system level metrics such as *congestion*, *accident rates*, and *greenhouse gas emissions* through a combination of intelligent routing, smoother driving behavior, and faster reaction time [155].

Partially automated systems are predicted to increasingly populate roadways between 2020 and 2025 but will primarily be usable in high driving or high speed operations in light traffic. Hazard detection technology is not expected to be mature enough for full automation in the presence of general vehicles and pedestrians (i.e. heterogeneous fleets, manned/unmanned, bicycles, pedestrians, mixed use road-space etc.) until at least 2030. It takes 20 years for a vehicle fleet to turn over sufficiently which makes it likely that vehicles will be partially manned at least until 2050 [125].

Recently, the steady increase in usage of cruise control systems on the roadway offers an opportunity to study the optimization of traffic in the framework of *mixed-autonomy*

traffic: traffic that is partially automated but mostly still consists of human driven vehicles. However, the control problems posed in this framework are notoriously difficult to solve. Traffic problems, which often exhibit features such as time-delay, non-linear dynamics, and hybrid behavior, are challenging for classical control approaches, as microscopic traffic models are high complexity: discrete events (lane changes, traffic light switches), continuous states (position, speed, acceleration), and non-linear driving models. These complexities make analytical solutions often intractable. The variety and non-linearity of traffic often leads to difficult trade-offs between the fidelity of the dynamics model and tractability of the approach.

Classical control approaches: Classical control approaches have successfully solved situations in which the complexity of the problem can be reduced without throwing away key aspects of the dynamics. For example, there is a variety of analytical work on control of autonomous intersections with simple geometries. For mixed-autonomy problems, there have been significant classical controls based results for simple scenarios like vehicles on a ring [38] or a single lane of traffic whose stability can be characterized [108, 136]. A thorough literature review on coordinating autonomous vehicles in intersections, merging roadways, and roundabouts can be found in [115]. The classical control approaches described in this review can be broken down into reservation methods, scheduling, optimization with safety constraints, and safety maximization. Other approaches discussed in the review involve applications of queuing theory, game theory, and mechanism design.

However, as the complexity of the problem statement increases, classical techniques become increasingly difficult to apply. Shifting focus from simple scenarios to, for example, hybrid systems with coexisting continuous and discrete controllers, explicit guarantees for hand-designed controllers can become harder to find. Ultimately, when the complexity of the problem becomes too high, optimization-based approaches have been shown to be a successful approach in a wide variety of domains from robotics [75] to control of transportation infrastructure [83].

Deep reinforcement learning: *Deep reinforcement learning* (deep RL) has recently emerged as an effective technique for control in high dimensional, complex CPS systems. Deep RL has shown promise for the control of complex, unstructured problems as varied as robotic skills learning [56], playing games such as Go [127], and traffic light ramp metering [19]. Of particular relevance to this work, deep RL has been successful in training a single autonomous vehicle to optimize traffic flow in the presence of human drivers [167].

One key distinction in RL is whether the algorithm is model-free or model-based, referring to whether the algorithm is able to query a dynamics model in the computation of the control or the policy update. Model-free RL tends to outperform model-based RL if given sufficient optimization time, but requires longer training times. Thus, model-free techniques are most effective when samples can be cheaply and rapidly generated. This often means that model-free RL works best in simulated settings where a simulation step can be made faster than real-time and simulation can be distributed across multiple CPUs or GPUs. A long-standing goal is to be able to train a controller in simulation, where model-free techniques can be used, and then use the trained controller to control the actual system.

Policy Transfer: Transfer of a controller from a training domain to a new domain is referred to as *policy transfer*. The case where the policy is directly transferred without any fine-tuning is referred to as *zero-shot policy transfer*. Zero-shot policy transfer is a difficult problem in RL, as the true dynamics of the system may be quite different from the simulated dynamics, an issue referred to as *model mismatch*. Techniques used to overcome this include adversarial training, in which the policy is trained in the presence of an adversary that can modify the dynamics and controller outputs and the policy must subsequently become robust to perturbations [110]. Other techniques to overcome *model mismatch* include re-learning a portion of the controller[119], adding noise to the dynamics model [109], and learning a model of the true dynamics that can be used to correctly execute the desired trajectory of the simulation-trained controller [35]. Efforts to overcome the reality gap have been explored in vision-based reinforcement learning [96] and in single AV systems [172].

Other challenges with policy transfer include *domain mismatch*, where the true environment contains states that are unobserved or different from simulation. For example, an autonomous vehicle might see a car color that is unobserved in its simulations and subsequently react incorrectly. Essentially, the controller overfits to its observed states and does not generalize. Domain mismatch can also occur as a result of imperfect sensing or discrepancies between the simulation and deployment environment. While in simulation it is possible to obtain perfect observations, this is not always the case in the real world. Small differences between domains can lead to drastic differences in output. For example, a slight geometric difference between simulation and real world could result in a vehicle being registered as being on one road segment, when it is on another. This could affect the control scheme in a number of ways, such as a premature traffic light phase change. Techniques used to tackle this problem include domain randomization [140], in which noise is injected into the state space to enforce robustness with respect to unobserved states.

Contributions and organization of the article: In this work we use deep RL to train two autonomous vehicles to learn a classic form of control: ramp metering, in which traffic flow is regulated such that one flow of vehicles is slowed such that another flow can travel faster. While in the real world, ramp metering is controlled via metering lights, we demonstrate the same behavior using AVs instead of lights. Each RL vehicle interacts with sensors at each of the entrance ramps and is additionally able to acquire state information about vehicles on the roundabout, as well as state information about the other RL vehicle. By incorporating this additional sensor information, we attempt to learn a policy that can time the merges of the RL vehicles and their platoons to learn ramp metering behavior, which prevents energy-inefficient decelerations and accelerations. Being positioned at the front of a platoon of vehicles, each RL vehicle has the ability to control the behavior of the platoon of human-driven vehicles following it. The RL vehicle, also referred to in this paper as an autonomous vehicle (AV), is trained with the goal of minimizing the average delay of all the vehicles in simulation.

Next, we show how we overcome the RL to real world reality gap and demonstrate RL's real world relevance by transferring the controllers to the University of Delaware's Scaled Smart City (UDSSC), a reduced-scale city whose dynamics, which include sensor delays,

friction, and actuation, are likely closer to true vehicle dynamics. RL trained policies, which are learned in a simulation environment, can overfit to the dynamics and observed states of the simulator and can then fare poorly when transferred to the real world. The combination of model and domain mismatch contributes to this problem. We combine the ideas of domain randomization with adversarial perturbations to the dynamics and train a controller in the presence of noise in both its observations and actions. For reasons discussed in Sec. 5.4, we expect the addition of noise in both state and action to help account for both model and domain mismatch.

In this work we present the following results:

- The use of deep RL in simulation to learn an emergent metering policy.
- A demonstration that direct policy transfer to UDSSC leads to poor performance.
- A successful zero-shot policy transfer of the simulated policy to the UDSSC vehicles via injection of noise into both the state and action space.
- An analysis of the improvements that the autonomous vehicles bring to the congested roundabout.

The remainder of the article is organized as follows:

1. Section 6.2 provides an introduction to deep RL and the algorithms used in this work.
2. Section 5.3 describes the setup we use to learn control policies via RL, followed by the policy transfer process from simulation to the physical world.
3. Section 5.4 discusses the results of our experiments and provides intuition for the effectiveness of the state and action noise.
4. Section 6.6 summarizes our work and future directions.

5.2 Background

Reinforcement Learning

We adhere to the conventional reinforcement learning (RL) framework [134, 89], which seeks to optimize the discounted cumulative rewards within a finite time frame for a Partially Observable Markov Decision Process (POMDP) [33]. This is defined by the tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, T, R, T_0, \gamma, \Omega, \mathcal{O})$.

For a formal description and background of RL, refer to Chapter 2.

Policy Gradient Methods

The particular policy gradient method used in this paper is *Trust Region Policy Optimization* (TRPO) [124]. TRPO is a monotonic policy improvement algorithm, whose update step provides guarantees of an increase in the expected total reward. However, the exact expression for the policy update leads to excessively small steps so implementations of TRPO take larger steps by using a trust region. In this case, the trust region is a bound on the KL divergence between the old policy and the policy update. While not a true distance measure, a small KL divergence between the two policies suggests that the policies do not act too differently over the observed set of states, preventing the policy update step from sharply shifting the policy behavior.

For a formal description and background on policy gradients, refer to Chapter 2.

Car Following Models

For our model of the driving dynamics, we used the *Intelligent Driver Model* [143] (IDM) that is built into the traffic microsimulator SUMO [71]. IDM is a microscopic car-following model commonly used to model realistic driver behavior. The IDM paramter values used in our simulations are given in Sec. 5.3. To better model the natural variability in driving behavior, we induce stochasticity in the desired driving speed v_0. For a given vehicle, the value of v_0 is sampled from a Gaussian whose mean is the speed limit of the lane and whose standard deviation is 20% of the speed limit.

Car following models are not inherently collision-free, we supplement them with a safe following rule: a vehicle is not allowed to take on velocity values that might lead to a crash if its lead vehicle starts braking at maximum deceleration. However, due to some uncertainty in merging behavior, there are still rare crashes that can occur in the system.

For a formal description and background on car following models, refer to Chapter 2.

Flow

We run our experiments in *Flow* [167], a library that provides an interface between a traffic microsimulator, SUMO [71], and two RL libraries, rllab [40] and RLlib [86], which are centralized and distributed RL libraries respectively. *Flow* enables users to create new traffic networks via a Python interface, introduce autonomous controllers into the networks, and then train the controllers in a distributed system on the cloud via AWS EC2. To make it easier to reproduce our experiments or try to improve on our benchmarks, the code for *Flow*, scripts for running our experiments, and tutorials can be found at https://github.com/flow-project/flow.

Fig. 5.1 describes the process of training the policy in *Flow*. The controller, here represented by policy π_θ, receives a state and reward from the environment and uses the state to compute an action. The action is taken in by the traffic microsimulator, which outputs the next state and a reward. The (state, next state, action, reward) tuple are stored as a sample

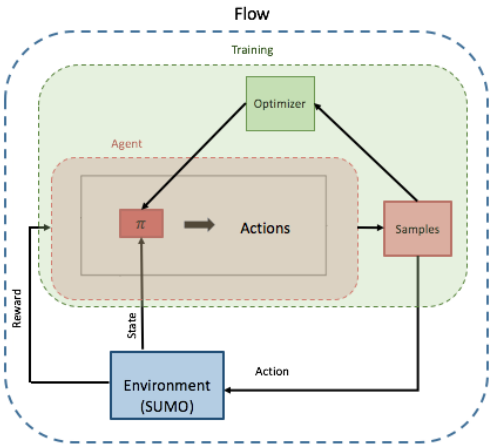


Figure 5.1: Diagram of the iterative process in *Flow*. Portions in red correspond to the controller and rollout process, green to the training process, and blue to traffic simulation.

to be used in the optimization step. After accumulating enough samples, the states, actions, and rewards are passed to the optimizer to compute a new policy.

University of Delaware's Scaled Smart City (UDSSC)

The *University of Delaware's Scaled Smart City (UDSSC)* was used to validate the performance of the RL control system. UDSSC is a testbed (1:25 scale) that can help prove concepts beyond the simulation level and can replicate real-world traffic scenarios in a small and controlled environment. UDSSC uses a VICON camera system to track the position of each vehicle with sub-millimeter accuracy, which is used both for control and data collection. The controller for each vehicle is offloaded to a mainframe computer and runs on an independent thread which is continuously fed data from the VICON system. Each controller uses the global VICON data to generate a speed reference for the vehicles allowing for precise independent closed-loop feedback control. A detailed description of UDSSC can be found in [128]. To validate the effectiveness of the proposed RL approach in a physical environment, the southeast roundabout of the UDSSC was used (Fig. 5.2).

5.3 Experimental Deployment

Experimental setup

Simulation Details

To derive the RL policy, we developed a model of the roundabout highlighted in red in Fig. 5.2 in SUMO. The training of the model, shown in Fig. 5.3, included a single-lane

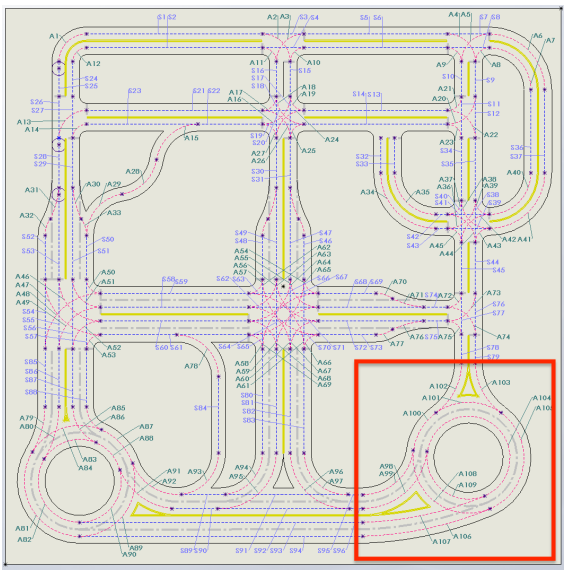


Figure 5.2: Diagram of the UDSSC road map, with the experimental zone highlighted in red.

roundabout with entry points at the northern and western ends. Throughout this paper we will refer to vehicles entering from the western end as the *western platoon* and the north entrance as the *northern platoon*. The entry points of the model are angled slightly different as can be seen in Figs. 5.2 and 5.3.

The human-controlled vehicles operate using SUMO's built-in IDM controller, with several modified parameters. In these experiments, the vehicles operating with the IDM controller are run with $T = 1$, $a = 1$, $b = 1.5$, $\delta = 4$, $s_0 = 2$, $v_0 = 30$, and noise$= 0.1$, where T is a safe time headway, a is a comfortable acceleration in m/s^2, b is a comfortable deceleration, δ is an acceleration exponent, $s0$ is the linear jam distance, v_0 is a desired driving velocity, and noise is the standard deviation of a zero-mean normal perturbation to the acceleration or deceleration. Details of the physical interpretation of these parameters can be found in [143]. Environment parameters in simulation were set to match the physical constraints of UDSSC. These include: a maximum acceleration of $1\frac{\mathrm{m}}{\mathrm{s}^2}$, a maximum deceleration of $-1\frac{\mathrm{m}}{\mathrm{s}^2}$, and a maximum velocity of $15\frac{\mathrm{m}}{\mathrm{s}}$. The timestep of the system is set to 1.0 seconds.

Simulations on this scenario in the roundabout were executed across a range of different settings in terms of volume and stochasticity of inflows. In the RL policy implemented in UDSSC and discussed in 5.3, vehicles are introduced to the system via deterministic inflows from the northern and western ends of the roundabout using two routes: (1) the northern platoon enters the system from the northern inflow, merges into the roundabout, and exits through the western outflow and (2) the western platoon enters the system from the western inflow, U-turns through the roundabout, and exits through the western outflow. The western platoon consists of four vehicles total: three vehicles controlled with the IDM controller led

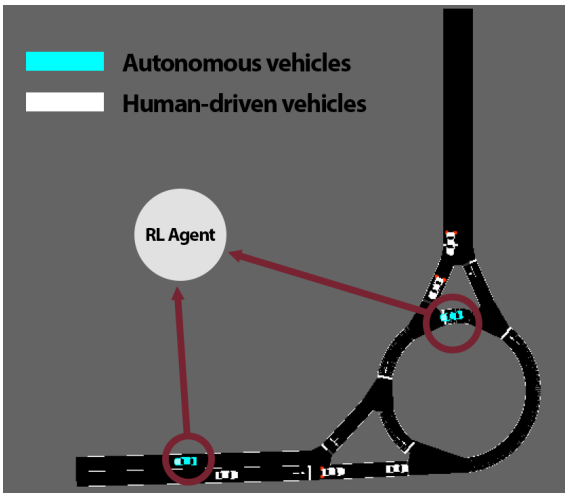


Figure 5.3: SUMO-generated network in UDSSC's roundabout. The blue vehicles are the AVs; they are both controlled by the RL policy. Videos of this policy in simulation are available at https://sites.google.com/view/iccps-policy-transfer.

by a vehicle running with the RL policy. The northern platoon consists of of three vehicles total: two vehicles controlled with the IDM controller led by a vehicle running with the RL policy. New platoons enter the system every 1.2 minutes, the rate of which is significantly sped up in simulation. These inflow settings are designed to showcase the scenario where routes clash (Fig. 5.3).

UDSSC

Each vehicle in UDSSC uses a saturated IDM controller to (1) avoid negative speeds, (2) ensure that the rear-end collision constraints do not become active, and (3) maintain the behavior of Eq. (8.1). Both the IDM and RL controllers provide a desired acceleration for the vehicles, which is numerically integrated to calculate each vehicle's reference speed.

Merging at the northern entrance of the roundabout is achieved by an appropriate yielding function. Using this function, the car entering the roundabout proceeds only if no other vehicle is on the roundabout at a distance from which a potential lateral collision may occur. Otherwise, the vehicle stops at the entry of the roundabout waiting to find a safe space to proceed.

To match the SUMO training environment, only one vehicle per path was allowed to use the RL policy. The paths taken by each vehicle are shown in Fig. 5.4. For each path, the first vehicle to enter the experimental zone was controlled by the RL policy; every subsequent vehicle runs with the saturated IDM controller. Once an active vehicle running with the RL policy exits the experimental zone, it reverts back to the IDM controller.

In the experiments, the vehicles operated in a predefined deterministic order, as described

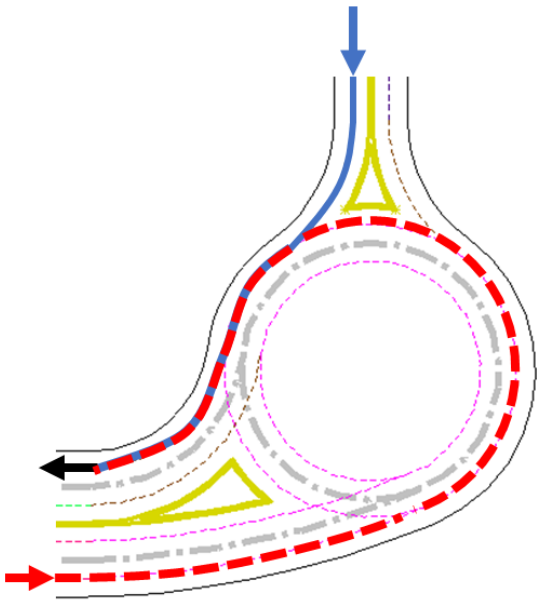


Figure 5.4: Visualization of the path taken on the UDSSC roundabout. The red route enters going east and exits going west; the blue route enters north and exits via the same west entrance as the red route.

in 5.3. Four vehicles were placed just outside the experimental zone on the western loop, and three vehicles were placed in the same fashion near the northern entrance. Each vehicle platoon was led by a vehicle running with the RL policy, except in the baseline case where all vehicles used the saturated IDM controller. The experiment was executed with three variations: (1) the baseline case with all vehicles running with the IDM controller, (2) the case with a leader vehicle running with an RL policy trained in SUMO, and (3) the case where the leader vehicles running with an RL policy were trained in simulation with noise injected into their observations and accelerations.

Reinforcement Learning Structure

Action space

We parametrize the controller as a neural net mapping the observations to a mean and diagonal covariance matrix of a Gaussian. The actions are sampled from the Gaussian; this is a standard controller parametrization [82]. The actions are a two-dimensional vector of accelerations in which the first element corresponds to vehicles on the north route and the second element to the west route. Because the dimension of the action vector is fixed, there can only ever be 1 AV from the northern entry and 1 AV from the western entry. Two queues, one for either entryway, maintain a list of the RL-capable vehicles that are currently in the system. It should be noted that the inflow rates are chosen such that the trained

policy never contains more than a queue of length 2. Platoons are given ample time to enter and exit the system before the next platoon arrives. This queue mechanism is designed to support the earlier stages of training, when RL vehicles are learning how to drive, which can result in multiple sets of platoons and thus more than 2 RL vehicles being in the system at the same time. Control is given to vehicles at the front of both queues. When a vehicle completes its route and exits the experimental zone, its ID is popped from the queue. All other RL-capable vehicles are passed IDM actions until they reach the front of the queue. If there are fewer than two AVs in the system, the extra actions are simply unused.

The dynamics model of the autonomous vehicles are given by the IDM described in sec. 5.2 subject to a minimum and maximum speed i.e.

$$v_j^{\text{IDM}}(t+\Delta t) = \max\left(\min\left(v_{AV}(t) + a_{IDM}\Delta t, v_j^{\max}(t)\right), 0\right) \tag{5.1}$$

where $v_j^{AV}(t)$ is the velocity of autonomous vehicle j at time t, a_{IDM} is the acceleration given by an IDM controller, Δt is the time-step, and $v_j^{\max}(t)$ is the maximum speed set by the city j. For the AVs, the acceleration a_t is straightforwardly added to the velocity via a first-order Euler integration step

$$v_j^{\text{AV}}(t+\Delta t) = \max\left(\min\left(v_j^{\text{AV}}(t) + a_t\Delta t, v_j^{\max}(t)\right), 0\right) \tag{5.2}$$

Observation space

For the purposes of keeping in mind physical sensing constraints, the state space of the MDP is partially observable. It is normalized to ± 1 and includes the following:

- The positions of the AVs.
- The velocities of the AVs.
- The distances from the roundabout of the 6 closest vehicles to the roundabout for both roundabout entryways.
- The velocities of the 6 closest vehicles to the roundabout for both roundabout entryways.
- Tailway and headway (i.e. distances to the leading and following vehicles) of vehicles from both AVs.
- Length of the number of vehicles waiting to enter the roundabout for both roundabout entryways.
- The distances and velocities of all vehicles in the roundabout.

This state space was designed with real-world implementation in mind, and could conceivably be implemented on existing roadways equipped with loop detectors, sensing tubes, and

vehicle-to-vehicle communication between the AVs. For a sufficiently small roundabout, it is possible that an AV equipped with enough cameras could identify the relevant positions and velocities of roundabout vehicles. Similarly, the queue lengths can be accomplished with loop detectors, and the local information of the AVs (its own position and velocity, as well as the position and velocity of its leader and follower) are already necessarily implemented in distance-keeping cruise control systems.

Action and State Noise

The action and state spaces are where we introduce noise with the purpose of training a more generalizable policy that is more resistant to the difficulties of cross-domain transfer. We train the policies in two scenarios, a scenario where both the action and state space are perturbed with noise and a scenario with no noise. This former setting corresponds to a type of *domain randomization*. In the noisy case, we draw unique perturbations for each element of the action and state space from a Gaussian distribution with zero mean and a standard deviation of 0.1. In the action space, which is composed of just accelerations, this corresponds to a standard deviation of $0.1\frac{\text{m}}{\text{s}^2}$. In the state space, which is normalized to 1, this corresponds to a standard deviation of 1.5 m/s for velocity-based measures. The real-life deviations of each distance-based state space element are described here: AV positions, and the tailways and headways of the AVs, deviate the most at 44.3 m, the large uncertainty of which results in a policy that plays it safe. The distance from the northern and western entryways respectively deviate by 7.43m and 8.66 m. The length of the number of vehicles waiting to enter the roundabout from the northern and western entryway respectively deviate by 1.6 and 1.9 vehicles.

These perturbations are added to each element of the action and state space. The elements of the action space are clipped to the maximum acceleration and deceleration of ± 1, while the elements of the state space are clipped to ± 1 to maintain normalized boundaries. Noisy action and state spaces introduce uncertainty to the training process. The trained policy must still be effective even in the presence of uncertainty in its state as well as uncertainty that its requested actions will be faithfully implemented.

Reward function

For our reward function we use a combination of the L2-norm of the velocity of all vehicles in the system and penalties discouraging standstills or low velocity travel.

$$r_t = \frac{\max\left(v_{\max}\sqrt{n} - \sqrt{\sum_{i=1}^{n}(v_{i,t} - v_{\max})^2}, 0\right)}{v_{\max}\sqrt{n}} - 1.5 \cdot \text{pen}_\text{s} - \text{pen}_\text{p} \tag{5.3}$$

where n is the number of all vehicles in the system, $v_{\max}$ is the maximum velocity of $15\frac{\text{m}}{\text{s}}$, $v_{i,t}$ is the velocity that vehicle v_i is travelling at at time t. The first term incentives vehicles to travel near speed $v_{\max}$ but also encourages the system to prefer a mixture of low and

high velocities versus a mixture of mostly equal velocities. The preference for low and high velocities is intended to induce a platooning behavior. RL algorithms are sensitive to the scale of the reward functions; to remove this effect the reward is normalized by $v_{\max}\sqrt{n}$ so that the maximum reward of a time-step is 1.

This reward function also introduces 2 penalty functions, pen_s and pen_p. pen_s returns the number of vehicles that are traveling at a velocity of 0, and pen_p is the number of vehicles that are traveling below a velocity of 0.3 m/s. They are defined as:

$$\text{pen}_\text{s} = \sum_{i=1}^{n} g(i) \quad \text{where} \quad g(x) = \begin{cases} 0, & v_x \neq 0, \\ 1, & v_x = 0. \end{cases} \tag{5.4}$$

$$\text{pen}_\text{p} = \sum_{i=1}^{n} h(i) \quad \text{where} \quad h(x) = \begin{cases} 0, & v_x \geq 0.3, \\ 1, & v_x \leq 0.3 \end{cases} \tag{5.5}$$

These penalty functions are added to discourage the autonomous vehicle from fully stopping or adopting near-zero speeds. In the absence of these rewards, the RL policy learns to game the simulator by blocking vehicles from entering the simulator on one of the routes, which allows for extremely high velocities on the other route. This occurs because velocities of vehicles that have not yet emerged from an inflow do not register, so no penalties are incurred when the AV blocks further vehicles from entering the inflow.

Algorithm/simulation details

We ran the RL experiments with a discount factor of .999, a trust-region size of .01, a batch size of 20000, a horizon of 500 seconds, and trained over 100 iterations. The controller is a neural network, a *Gaussian multi-layer perceptron* (MLP), with hidden sizes of $(100, 50, 25)$ and a tanh non-linearity. The choice of neural network non-linearities, size, and type were picked based on traffic controllers developed in [150]. The states are normalized so that they are between 0 and 1 by dividing each states by its maximum possible value. The actions are clipped to be between -1 and 1. Both normalization and clipping occur after the noise is added to the system so that the bounds are properly respected.

Code reproducibility

In line with open, reproducible science, the following codebases are needed to reproduce the results of our work. *Flow* can be found at `https://github.com/flow-project/flow`. The version of *rllab* used for the RL algorithms is available at `https://github.com/cathywu/rllab-multiagent` at commit number **4b5758f**. *SUMO* can be found at `https://github.com/eclipse/sumo` at commit number **1d4338ab80**.

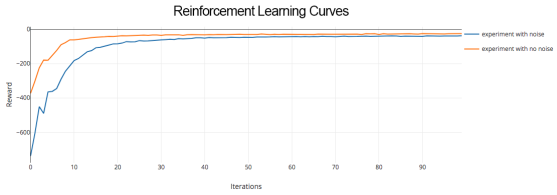


Figure 5.5: Convergence of the RL reward curve of an experiment with noised IDM, RL accelerations, and noisy state space

Policy Transfer

The RL policy learned through Flow was encoded as the weights of a neural network. These weights were extracted from a serialized file and accessed via a Python function which maps inputs and outputs identical to those used in training. Separating these weights from rllab enables an interface for state space information from the UDSSC to be piped straight into the Python function, returning the accelerations to be used on the UDSSC vehicles. The Python function behaves as a control module within the UDSSC, replacing the IDM control module in vehicles operating under the RL policy.

The inputs to the RL neural network were captured by the VICON system and mainframe. The global 2D positions of each vehicle were captured at each time step. These positions were numerically derived to get each vehicle's speed and were compared to the physical bounds on the roadways to get the number of vehicles in each queue at the entry points. Finally, the 2D positions were mapped into the 1D absolute coordinate frame used during training. This array was passed into the RL control module as the inputs of the neural network.

Results

In this section we present our results from a) training vehicular control policies via RL in simulation, and b) transferring the successful policy to UDSSC. Extended work and videos of the policies in action are available at https://sites.google.com/view/iccps-policy-transfer.

Simulation results

Fig. 5.5 depicts the reward curve. The noise-injected RL policy takes longer to train than the noise-free policy and fares much worse during initial training, but converges to an almost identical final reward. In the simulations, videos of which are on the website, a ramp metering behavior emerges in which the incoming western vehicle learns to slow down to allow the vehicles on the north ramp to smoothly merge.

This ramp metering behavior can also be seen in the space-time diagrams in Fig. 5.6, which portrays the vehicle trajectories and velocities of each vehicle in the system. Western

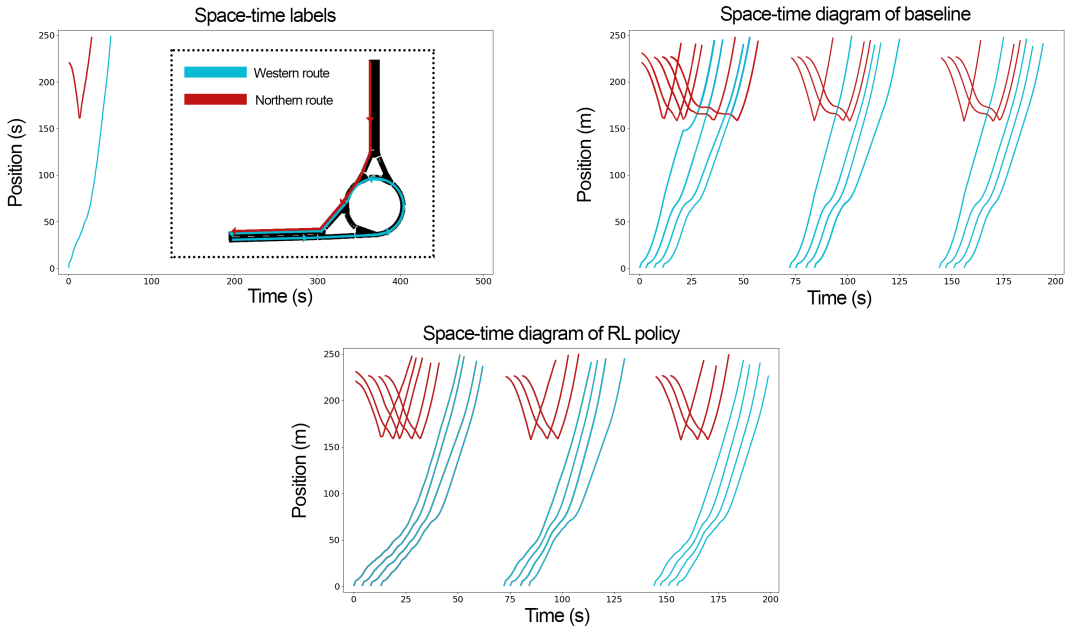


Figure 5.6: Space-time diagrams of the simulated baseline and RL policy. Each line corresponds to a vehicle in the system. Top: a guide to the color-scheme of the space time diagrams. The northern route is in red, the western route in blue. Middle: illustrates the overlap between the merging northern platoon and the western platoon. Bottom: The RL policy, depicted at the bottom successfully removes this overlap. Videos of this policy in simulation are available at https://sites.google.com/view/iccps-policy-transfer.

vehicles are depicted in blue and northern in red. Due to the overlapping routes, visible in Fig. 5.4, it was necessary to put a kink in the diagram for purposes of clarity; the kink is at the point where the northern and western routes meet. As can be seen in the middle figure, in the baseline case the two routes conflict as the northern vehicles aggressively merge onto the ramp and cut off the western platoon. Once the RL policy controls the autonomous vehicles, it slows down the western platoon so that no overlap occurs and the merge conflict is removed.

Transfer to UDSSC

The RL policies were tested under three cases in UDSSC: (1) the baseline case with only vehicles running with the IDM controller, (2) the case with a leader vehicle running with the RL policy trained in sumo without additional noise, and (3) the case where the leader vehicles running with the RL policy were trained with noise actively injected into their observations and accelerations. The outcomes of these trials are presented in Table 5.1.

During the congestion experiment, the third case, in which noise was actively injected into the action and state space during training, successfully exhibits the expected behavior

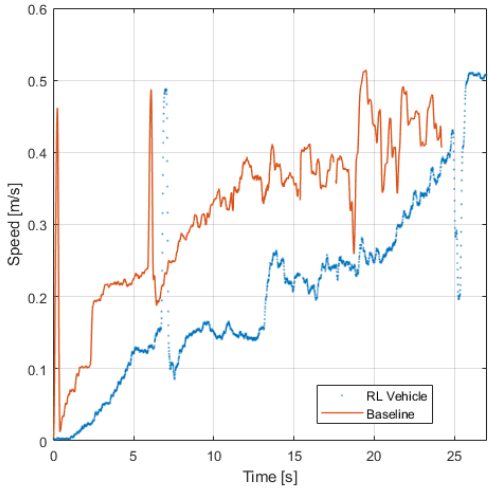


Figure 5.7: Comparison of the first vehicle on the southern loop for the baseline (IDM) and RL experiments. The RL vehicle starts off slower but eventually accelerates sharply once the northern platoon has passed.

Figure 5.8: Experiment with two platoons being led by RL vehicles (blue, circled).

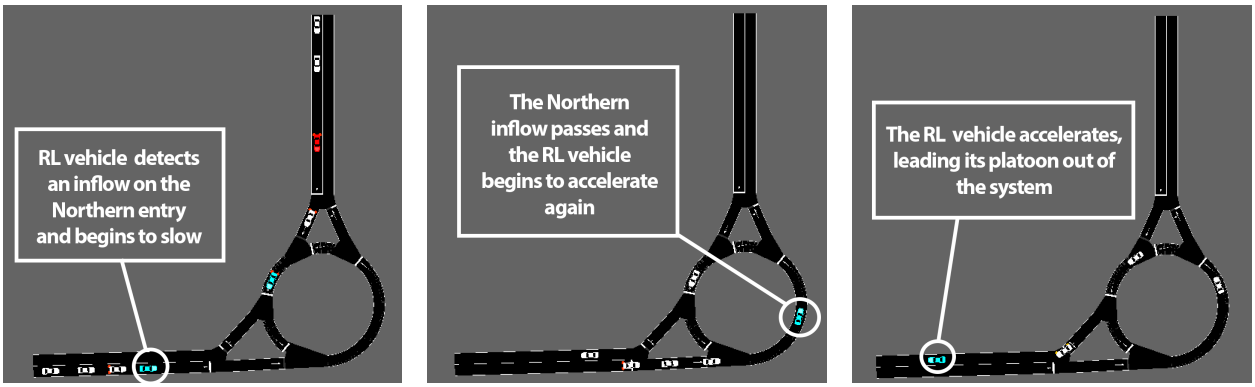


Figure 5.9: RL-controlled vehicle demonstrating smoothing behavior in this series of images. **First**: RL vehicle slows down in anticipation of a sufficiently short inflow from the north. **Second**: The northern inflow passes through the roundabout at high velocity. **Fourth**:The RL vehicle accelerates and leads its platoon away from the roundabout. Videos of this policy in simulation are available at https://sites.google.com/view/iccps-policy-transfer.

it demonstrated in simulation. In this RL controlled case, the transfer consistently showed successful ramp metering: the western platoon adopted a lower speed than the baseline IDM controller, as can be seen in the lower velocity of the RL vehicle in Fig. 5.7. This allowed the northern queue to merge before the western platoon arrived, increasing the overall throughput of the roundabout. This is closer to a socially optimal behavior, leading to a lower average travel time than the greedy behavior shown in the baseline scenario. No unexpected or dangerous driving behavior occurred.

This is in comparison with the second case, a policy trained on noiseless observations. In the second case, undesirable and unexpected deployment behavior suggests problems with the transfer process. Collisions occurred, sometimes leading to pile-ups, and platooning would frequently be timed incorrectly, such that, for example, only part of the Western platoon makes it through the roundabout before the Northern platoon cuts the Western platoon off. This indicates that the noise-injected policy is robust to transfer and resistant to domain and model mismatch.

Furthermore, the platoons led by RL vehicles trained with injected noise outperformed the baseline and noise-free cases. The results of these experiments, averaged over three trials with the RL vehicles, are presented in Table 5.1. This improvement was the outcome of a metering behavior learned by the western RL platoon leader. In the baseline case, the north and western platoons meet and lead to a merge conflict that slows the incoming western vehicles down. This sudden decrease in speed can be seen in the drop in velocity at 20 seconds of the baseline in Fig. 5.7.

Fig. 5.8 shows the experiment in progress, with the blue (circled) vehicles being RL vehicles trained under noisy conditions. The RL vehicle entering from the western (lower)

	Avg. Vel.[m/s]	Avg. Time[s]	Max Time[s]
Baseline	0.26	15.71	23.99
RL	0.22	15.68	20.62
RL with Noise	0.23	14.81	18.68

Table 5.1: Results for the congestion experiment, the average and maximum times are averaged between three RL trials and a single baseline trial.

entrance has performed it's metering behavior, allowing vehicles from the northern (upper) queue to pass into the roundabout before the western RL vehicle speeds up again. Videos of the emergent behavior can be found on the website.

5.4 Discussion

For the simulated environment, the choice of reward function, specifically, using the L2-norm rather than the L1-norm, encourages a more stable, less sparse solution. This makes evaluating the success of policy transfer more straightforward. Table 5.1 reports the results of the UDSSC experiments on three metrics:

- The average velocity of the vehicles in the system
- The average time spent in the system
- The maximum time that any vehicle spent in the system

Note, the system is defined as the entire area of the experiments, including the entrances to the roundabouts. The layer of uncertainty in the noise-injected policy aided with overcoming the domain and model mismatch between the simulation system and the UDSSC system. Thus, the noised policy was able to successfully transfer from simulation to UDSSC and also improved the average travel time by 5% and the maximum travel time by 22%. The noise-free policy did not improve on the average travel time and only improved the maximum travel time by 14%. Although we did not perform an ablation study to check whether both state space noise and action space noise were necessary, this does confirm that the randomization improved the policy transfer process.

The UDSSC consistently reproduced the moderate metering behavior for the noise-injected policy, but did not do so for the policy that was trained without noise. As can be seen in the videos, the noise-free policy was not consistent and would only irregularly reproduce the desired behavior, or meter dramatically to the point that average travel time increased. Overall the noised policy significantly outperformed the noise-free version.

However, we caution that in our testing of the policy transfer process on the UDSSC, we performed a relatively limited test of the effectiveness of the policy. The vehicles were

all lined up outside the system and then let loose; thus, the tests were mostly deterministic. Any randomness in the tests would be due solely to randomness in the dynamics of the UDSSC vehicles and stochasticity in the transferred policy. In training, the acceleration of IDM vehicles are noised and can account for some stochasticity in the initial distribution of vehicles on the UDSSC. However, the trained policy was not directly given this inflow distribution at train time, so this does correspond to a separation between train and test sets.

There may be several reasons why the noise and action injection may have allowed for a successful zero shot transfer. First, because the action is noisy, the learned policy will have to learn to account for *model mismatch* in its dynamics: it cannot assume that the model is exactly the double-integrator that is used in the simulator. Subsequently, when the policy is transferred to an environment with both delay, friction, and mass, the policy sees the mismatch as just another form of noise and accounts for it successfully. The addition of state noise helps with domain randomization; although the observed state distributions of the simulator and the scaled city may not initially overlap, the addition of noise expands the volume of observed state space in the simulator which may cause the two state spaces to overlap. Finally, the addition of noise forces the policy to learn to appropriately filter noise, which may help in the noisier scaled city environment.

5.5 Conclusions

In this paper, we demonstrated the real-world relevance of deep RL AV controllers for traffic control by overcoming the gap between simulation and the real world. Using RL policies, AVs in the UDSSC testbed successfully coordinated at a roundabout to ensure a smooth merge. We trained two policies, one in the presence of state and action space noise, and one without, demonstrating that the addition of noise led to a successful transfer of the emergent metering behavior, while the noise-free policy often over-metered, or failed, to meter at all. This implies that for non-vision based robotic systems with small action-dimension, small amounts of noise in state and action space may be sufficient for effective zero-shot policy transfer. As a side benefit, we also demonstrate that the emergent behavior leads to a reduction of 5% in average travel time and 22% max-travel time on the transferred network.

Ongoing work includes characterizing this result more extensively, evaluating the effectiveness of the efficiency of the policy against a wide range of vehicle spacing, platoon sizes, and inflow rates. In this context, there are still several questions we hope to address, as for example:

- Are both state and action space noise needed for effective policy transfer?
- What scale and type of noise is most helpful in making the policy transfer?
- Would selective domain randomization yield a less lossy, more robust transfer?
- Would adversarial noise lead to a more robust policy?

- Can we theoretically characterize the types of noise that lead to zero-shot policy transfer?

Finally, we plan to generalize this result to more complex roundabouts including many lanes, many entrances, and the ability of vehicles to change lanes. We also plan to evaluate this method of noise-injected transfer on a variety of more complex scenarios, such as intersections using stochastic inflows of vehicles.

Chapter 6

Zero-shot autonomous vehicle policy transfer: From simulation to real-world via adversarial learning

6.1 Introduction

In 2015, commuters in the US spent an estimated 6.9 billion additional hours waiting in congestion, resulting in an extra 3.1 billion gallons of fuel, costing an estimated $160 billion [121]. An automated transportation system [177] can alleviate congestion, reduce energy use and emissions, and improve safety by increasing traffic flow. The use of connected and automated vehicles (CAVs) can transition our current transportation networks into energy-efficient mobility systems. Introducing CAVs into the transportation system allows vehicles to make better operational decisions, leading to significant reductions of energy consumption, greenhouse gas emissions, and travel delays along with improvements to passenger safety [155].

Several efforts have been reported in the literature towards coordinating CAVs to reduce spatial and temporal speed variations of individual vehicles. These variations can be introduced by breaking events, or due to the structure of the road network, e.g., intersections [80, 113, 92], cooperative merging, and speed reduction zones [93]. One of the earliest efforts in this direction was proposed by Athans [11] to efficiently and safely coordinate merging behaviors as a step to avoid congestion. With an eye toward near-future CAV deployment, several recent studies have explored the traffic and energy implications of partial penetration of CAVs under different transportation scenarios, e.g., [116, 117, 57, 160].

While classical control is an effective method for some traffic control tasks, the complexity and sheer problem size of autonomous driving in mixed-traffic scenarios makes it a notoriously difficult problem to address. In recent years, deep reinforcement learning (RL) has emerged as an alternative method for traffic control. RL is recognized for its ability to solve data-rich, complex problems such as robotic skills learning [56], to larger and more

complicated problems such as learning winning strategies for Go [127] or StarCraft II [7]. Deep RL is capable of handling highly complex behavior-based problems, and thus naturally extends to traffic control. The results from the ring road experiments that Stern and Sugiyama [130, 132] demonstrated their policies have also been achieved via RL methods [165]. AVs controlled with RL-trained policies have been further used to demonstrate their traffic-smoothing capabilities in simple traffic scenarios such as figure eight road networks [165], intersections [150] and roundabouts [67], and can also replicate traffic light ramp metering behavior [19]. Indeed, real-world evaluation and validation of control techniques under a variety of traffic scenarios is a necessary task.

The contributions of this article are: (1) the introduction of Gaussian single-agent noise and adversarial multi-agent noise to learn traffic control behavior for an automated vehicle; (2) a comparison performance with noise injected into the action space, state space, and both; (3) the demonstration of real-world disturbances leading to poor performance and crashes for some training methods, and (4) experimental demonstration of how autonomous vehicles can improve performance in a mixed-traffic system.

The remainder of this article is organized as follows. In Section 6.2, we provide background information on reinforcement learning, car-following models, the *Flow* framework, and the experimental testbed. In Section 6.3, we introduce the mixed-traffic roundabout problem and the implementation of the RL framework. In Section 6.4, we present the simulation results. In Section 6.5, we discuss the policy transfer process along with the experimental results. Finally, we draw concluding remarks in Section 6.6.

6.2 Background

Deep Reinforcement Learning

We adhere to the conventional reinforcement learning (RL) framework [134, 89], which seeks to optimize the discounted cumulative rewards within a finite time frame for a Partially Observable Markov Decision Process (POMDP) [33]. This is defined by the tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, T, R, T_0, \gamma, \Omega, \mathcal{O})$.

For a formal description and background of RL, refer to Chapter 2.

Policy Gradient Algorithms

There are a number of algorithms that exist for deriving an optimal RL policy π^*. For the experiments in this article, π^* is learned via proximal policy optimization (PPO) [122], a widely-used policy gradient algorithm.

In this article, we use PPO as the algorithm for the two types of experiments described in Sec. 6.3, Gaussian single-agent and adversarial multi-agent.

For a formal description and background of policy gradient algorithms, refer to Chapter 2.

Car Following Models

We use the intelligent driver model (IDM) [143] to model human driving dynamics. The IDM parameters and the values used in our simulation are described in Section 6.4.

For a formal description and background of car following models, including IDM, refer to Chapter 2.

Flow

For training the RL policies in this article, we use *Flow* [167], an open-source framework for interfacing RL libraries such as RLlib [86], Stable Baselines, and rllab [40] with traffic simulators such as SUMO [71] or Aimsun. *Flow* enables the ability to design and implement RL solutions for a flexible, wide variety of traffic-oriented scenarios. RL environments built using *Flow* are compatible with OpenAI Gym [26] and as such, support training with most RL algorithms. *Flow* also supports large-scale, distributed computing solutions via AWS EC2 [1].

6.3 Problem Formulation

To demonstrate the viability of autonomous RL vehicles in reducing congestion in mixed traffic, we implemented the scenario shown in Fig. 6.1. In this scenario, two groups of vehicles enter the roundabout stochastically, one at the northern end and one at the western end. In what follows, we refer to the vehicles entering from the north entry as the *northern group*, and to the vehicles entering from the west entry as the *western group*.

The baseline scenario consists of homogeneous human-driven vehicles using the IDM controller (8.1). The baseline is designed such that vehicles approaching the roundabout from either direction will clash at the roundabout. This results in vehicles at the northern entrance yielding to roundabout traffic, resulting in significant travel delays. The RL scenario puts an autonomous vehicle at the head of each group, which can be used to control and smooth the interaction between vehicles; these mixed experiments correspond to a $15\% - 50\%$ mixture of autonomous and human-driven vehicles.

Reinforcement Learning Structure

We categorize two sets of RL experiments that are used and compared in this article. We will refer to them as: *Gaussian single-agent*: A single-agent policy trained with Gaussian noise injected into the state and action space. *Adversarial multi-agent*: A multi-agent policy trained wherein a second agent provides selective adversarial noise to the learning agent.

[1]For further information on Flow, we refer readers to view the *Flow* Github page, website, or article, respectively listed here. Github: https://github.com/flow-project/flow, website: https://flow-project.github.io/, paper: [167].

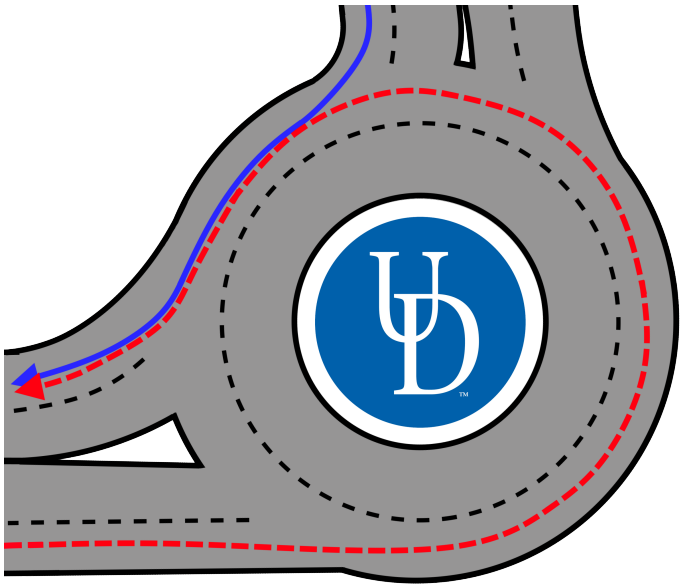


Figure 6.1: The routes taken by the northern (solid blue) and western (dashed red) groups through the roundabout.

We discuss the particulars of these two methods in Sections 6.3 and 6.3. In this article, we deploy seven RL-trained policies, one of which is single-agent with no noise, three of which are Gaussian single-agent and the other three of which are adversarial multi-agent. All experiments follow the same setup. Inflows of stochastic length emerge at the northern and western ends of the roundabout. The size of the northern group will range from 2 to 5 cars, while the size of the western group ranges from 2 to 8. The length of these inflows will remain static across each rollout, and are randomly selected from a uniform distribution at the beginning of each new rollout.

Action Space

The actions are applied from a 2-dimensional acceleration vector, in which the first element is used to control the AV leading the northern group, and the second is used to control the AV leading the western group. If the AV has left the experiment, that element of the action vector is discarded.

State Space

The state space conveys the following information about the environment to the agent: the position, velocity, tailway, and headway of each AV and each vehicle in the roundabout, the distance from the roundabout entrances to the 6 closest vehicles, the velocities of the 6 closest

vehicles to each roundabout entrance, the number of vehicles queued at each entrance, and the lengths of each inflow. All elements of the state space are normalized. The state space was designed with real-world implementation in mind and could contain any environmental factors that the simulation supports. As such, it is partially-observable to support modern sensing capabilities. All of these observations are reasonably selected and could be emulated in the physical world using sensing tools such as induction loops, camera sensing systems, and speedometers.

Reward Function

The reward function used for all experiments minimizes delay and applies penalties for standstill velocities, near-standstill velocities, jerky driving, and speeding, i.e.,

$$r_t = 2 \cdot \frac{\max\left(v_{\max}\sqrt{n} - \sqrt{\sum_{i=1}^{n}(v_{i,t} - v_{\max})^2}, 0\right)}{v_{\max}\sqrt{n}} - p, \tag{6.1}$$

$$p = p_s + p_p + p_j + p_v. \tag{6.2}$$

where n is the total number of vehicles, p is the sum of four different penalty functions, p_s is a penalty for vehicles traveling at zero velocity, designed to discourage standstill; p_p penalizes vehicles traveling below 0.2 m/s, which discourages the algorithm from learning an RL policy which substitutes extremely low velocities to circumvent the zero-velocity penalty; p_j discourages jerky driving by maintaining a dynamic queue containing the last 10 actions and penalizing the variance of these actions; and p_v penalizes speeding.

Gaussian single-agent noise

Injecting noise directly to the state and action space has been shown to aid with transfer from simulation to real-world testbeds [67, 140]. In this method, which applies to three of the policies we deployed, each element of the state space was perturbed by a random number selected from a Gaussian distribution. Only two elements describing the length of the inflows approaching the merge were left unperturbed. Elements of the state space corresponding to positioning on the merge edge were perturbed from a Gaussian distribution with a standard deviation of 0.05. For elements corresponding to absolute positioning, the standard deviation was 0.02. All other elements used a standard deviation of 0.1. These values were selected to set reasonable bounds for the degree of perturbation in the real world. Each element of the action space was perturbed by a random number selected from a zero mean Gaussian distribution with 0.5 standard deviation.

Adversarial multi-agent noise

For the other three policies, we use a form of adversarial training to yield a policy resistant to noise [110]. This is a form of multi-agent RL, in which two policies are learned. Adversarial

training pits two agents against each other in a zero-sum game. The first is structurally the same as the agent which is trained in the previous four policies. The second, *adversarial* agent has a reward function that is the negative of the first agent's reward; in other words, it is incentivized by the first agent's failure. The adversarial agent can attempt to lower the agent reward by perturbing elements of the action and state space of the first agent.

The adversarial agent's action space is a 1-dimensional vector of length 22, composed of perturbation values bound by $[-1, 1]$. The first two elements of the adversarial action space are used to perturb the action space of the original agent's action space. Adversarial action perturbations are scaled by 0.1. Combining adversarial training with selective randomization, the adversarial agent has access to perturb a subset of the original agent's state space. The remaining 20 elements of the adversarial agent's action space are used to perturb 20 selective elements of the original agent's state space. Both the adversarial action and state perturbations are scaled down by 0.1. The selected elements that the adversary can perturb are the observed positions and velocities of both controlled AVs in the system and the observed distances of vehicles from the merge points.

6.4 Simulation Framework

Car Following Parameters

As introduced in Section 6.2, the human-driven vehicles in these simulations are controlled via IDM. Accelerations are provided to the vehicles via (8.1) and (8.2). Within these equations, s_0 is the minimum spacing or minimum desired net distance from the vehicle in front of it, v_0 is the desired velocity, T is the desired time headway, δ is an acceleration exponent, a is the maximum vehicle acceleration, and b is the comfortable braking deceleration.

Human-driven vehicles in the system operate using SUMO's built-in IDM controllers, which allows customization to the parameters described above. Standard values for these parameters as well as a detailed discussion on the experiments producing these values can be found in [143]. In these experiments, the parameters of the IDM controllers are defined to be $T = 1$ s , $a = 1$ m/s^2 , $b = 1.5$ m/s^2, $\delta = 4$, $s_0 = 2$ m, $v_0 = 30$ m/s. A noise parameter 0.1 was used to perturb the acceleration of the IDM vehicles.

Environment parameters in the simulation were set to match the physical constraints of the experimental testbed. These include: a maximum acceleration of 1 m/s^2, a maximum deceleration of -3 m/s^2, and a maximum velocity of 8 m/s. The timestep of the system is set to 1 s.

Algorithm/Simulation Details

We ran experiments with a discount factor of 0.999, a trust-region size of 0.01, a batch size of 20000, a horizon of 500 seconds, and trained over 100 iterations. The controller is a neural network, a *Gaussian multi-layer perceptron* with a tanh non-linearity, and hidden sizes of

$(100, 50, 25)$. The choice of neural network non-linearities, size, and type were picked based on traffic controllers developed in [150]. The states are normalized so that they are between 0 and 1 by dividing each state by its maximum possible value. The agent actions are clipped to be between -3 and 1. Both normalization and clipping occur after the noise is added to the system so that the bounds are properly respected. The following codebases are needed to reproduce the results of our work. *Flow*[2], *SUMO*[3] and the version of *RLlib*[4] used for the RL algorithms is available on GitHub.

Simulation Results

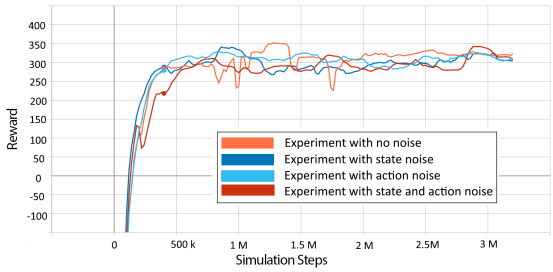


Figure 6.2: Convergence of the RL reward curves of the 3 Gaussian experiments and the noiseless policy.

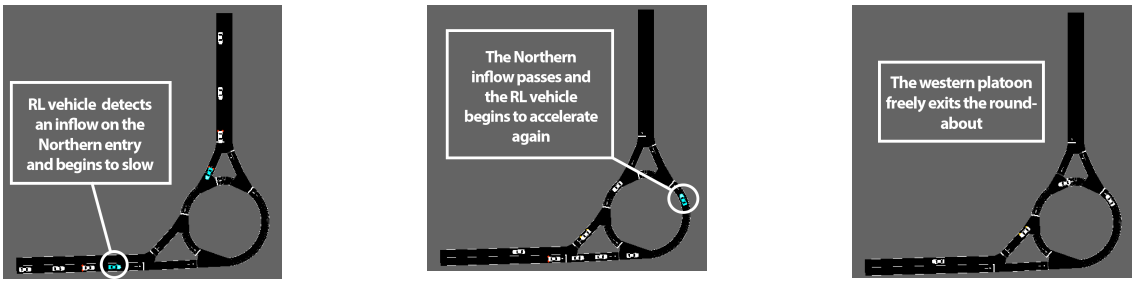


Figure 6.3: Two RL-controlled AVs trained with adversarial multi-agent noise demonstrate emergent ramp metering behavior.

The reward curves of the Gaussian single-agent experiments are displayed in Fig. 6.2. These include the curves of the 3 experiments, which are trained with Gaussian noise injection, as well as one trained without any noise. In both the Gaussian single-agent and

[2] https://github.com/flow-project/flow.
[3] https://github.com/eclipse/sumo at commit number **1d4338ab80**.
[4] https://github.com/flow-project/ray/tree/ray_master at commit number **ce606a9**.

adversarial multi-agent experiments, the policy learns a classic form of traffic control: ramp-metering, in which one group of vehicles slows down to allow for another group of vehicles to pass. Despite the varying length of inflows from the two entries, policies consistently converge to demonstrate ramp-metering.

6.5 Experimental Deployment

The University of Delaware's Scaled Smart City

University of Delaware's Scaled Smart City (UDSSC) is a 1:25 scale testbed designed to replicate real-world traffic scenarios and implement cutting-edge control technologies in a safe and scaled environment. UDSSC is a fully integrated smart city, which can be used to validate the efficiency of control and learning algorithms, including their performance on physical hardware. UDSSC utilizes high-end computers and a VICON motion capture system to simulate a variety of control strategies with as many as 35 scaled CAVs. For further information on the capabilities and features of the UDSSC, see [18].

To implement the RL policy in UDSSC, the weights of the network generated by *Flow* were exported into a data file. This file was accessed through a Python script on the main-frame, which uses a ROS service to map the current state of the experiment into a control action for each RL vehicle. During the experiment, the RL vehicles took commands from this script as opposed to the IDM controller.

To generate a disturbance on the roundabout system, a random delay for when each group was released was introduced. This delay was uniformly distributed between 0 and 1 seconds for the western group and between 0 and 4 seconds for the northern group during UDSSC experiments. The size of each vehicle group was randomly selected from a uniform distribution for each trial.

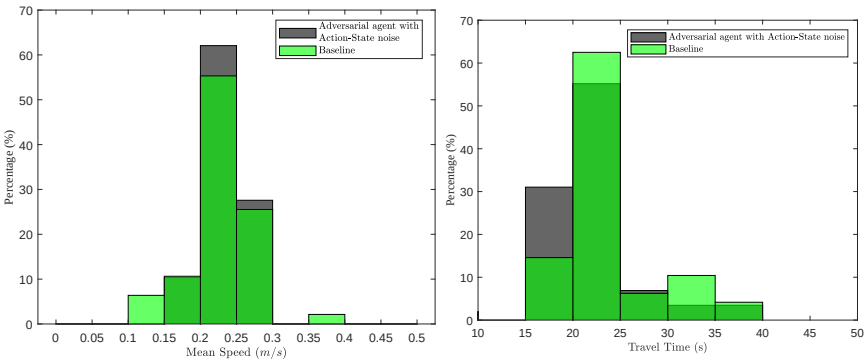
Experimental Results

The data for each vehicle was collected through the VICON motion capture system and is presented in Table 6.1. The position of each car was tracked for the duration of each experiment, and the velocity of each car was numerically derived with a first order finite difference method.

For all trials, the RL vehicle exhibited the learned ramp metering behavior, where the western leader reduced its speed to avoid yielding by the northern group. The metering behavior was extreme for the Gaussian single-agent noise case, especially when noise was added to the action and state together. This excessive metering significantly reduced the average speed and increased travel delay, as seen in Table 6.1. The adversarial multi-agent training significantly outperformed the Gaussian single-agent and tended to leave only a single vehicle yielding at the northern entrance. This strategy led to a travel time reduction for the northern group without a significant delay in western vehicles. The adversarial

Table 6.1: Experimental results for the baseline (no RL) case and each training method.

Training	Mean Time (s)	Mean Speed (m/s)	Trials	% Time Saved	Crashes
Baseline	23.6	0.23	47	-	0
Adversarial Multi-Agent					
Action-State	22.1	0.24	29	+6.3	0
Action	22.1	0.23	23	+6.4	0
State	21.4	0.24	26	+9.6	10
Gaussian Single-Agent					
Action-State	25.8	0.21	26	-9.2	0
State	23.0	0.23	18	+2.6	0
Action	23.1	0.22	37	+2.4	0
Noiseless	22.8	0.23	32	+3.5	0



(a) (b)

Figure 6.4: A relative frequency histogram for (a) the mean speed and (b) travel time of each vehicle for the baseline and adversarial multi-agent scenarios with noise injected in action and state.

multi-agent case with noise injected only into the state accelerated especially fast and led to several catastrophic accidents between the two RL vehicles. Finally, for small numbers of vehicles, the adversarial multi-agent trained controllers appeared to exhibit an emergent zipper merging behavior.[5]

Relative frequency histograms of average travel time and mean speed for the adversarial multi-agent case with noise injected into both action and state versus baseline scenarios are

[5]Videos of the experiment and supplemental information can be found at: `https://sites.google.com/view/ud-ids-lab/arlv`.

overlaid in Fig. 6.4. Over all trials, the adversarial case had a higher relative frequency of shorter travel time compared to the baseline scenario. Average travel time for 30% of adversarial scenarios, lies in the range [15s, 20s] comparing to 15% for the baseline scenarios.

From Table 6.1, we can see the average speed for baseline and the adversarial multi-agent with noise in action-state are nearly the same. However, in Fig. 6.4, we can see that approximately 8% of trials in the baseline scenarios have an average speed between 0.1 m/s and 0.15 m/s. Furthermore, there are some trials that the average speed of the baseline scenario is between 0.35 m/s and 0.4 m/s. On the other hand, the average speed for the adversarial multi-agent with noise in action and state varies less, and near 65% of trials have the average speed between 0.2 m/s and 0.25 m/s compared to 55% in the baseline scenarios.

6.6 Conclusion

In this article, we developed a zero-shot transfer of an autonomous driving policy directly from simulation to the UDSSC testbed. Even under stochastic, real-world disturbances, the adversarial multi-agent policy improved system efficiency by reducing travel time and average speed for most vehicles.

As we continue to investigate approaches for policy transfer, some potential directions for future research include: multi-agent adversarial noise with multiple adversaries, tuning to determine which elements of the state space are most suitable for perturbations, tuning injected noise to maximize policy robustness, larger, more complex interactions, such as intersections, or merging at highway on-ramps, and longer tests involving corridors with multiple bottlenecks.

ACKNOWLEDGMENT

The authors would like to thank Yiming Wan and Ishtiaque Mahbub for their contributions to UDSSC hardware and software.

In this section, we became acquainted with some of the challenges associated with policy transfer. In the case of traffic, this can consist of unpredictable collisions are possible, as well as a decrease in efficacy of the policy itself. We discussed some techniques to minimize this deterioration, including noise injection and strengthening via adversarial methods. With a better understanding of the roadblocks associated with real world deployment, the next part takes what we have learned to examine the opus of this work: developing an RL algorithm for a real world road test and deploying that algorithm onto 100 automated vehicles.

⋆ ⋆ ⋆ ⋆ ⋆

Part III

The MegaVanderTest: the Development and Deployment of Reinforcement Learning Policies in 100-AVs in a Real-World Highway

⋆ ⋆ ⋆ ⋆ ⋆

In the final part of this thesis, we move to cover the real world deployment of 100 automated vehicles running reinforcement learning controllers, to our knowledge the largest deployment of AVs designed to smooth traffic flow. We discuss the reinforcement learning algorithm development and end-to-end engineering involved in depth. A three-year long project, the next few chapters explore the progression of the work and the problem statement over those years. The first year focused solely on simulation, the second year ended in an initial 10-car test, and the final year resulted in the 100-AV test. Chapter 7 discusses the algorithm development of the originally intended reinforcement learning algorithm, with a detailed analysis of its performance in simulation. Chapter 8 contains the bulk of the work surrounding the reinforcement learning-related aspects of deployment and the work leading up to it. It continues the discussion on creating the RL policy, including changes made since the controller freeze of the algorithm described in the prior chapter. Lastly, Chapter 9 discusses the engineering work surrounding the reinforcement learning development.

Chapter 7

Traffic Smoothing Controllers for Autonomous Vehicles Using Deep Reinforcement Learning and Real-World Trajectory Data

7.1 INTRODUCTION

Transportation accounts for a large share of energy usage worldwide, with the U.S. alone attributing 28% of its total energy consumption in 2021 to moving people and goods [148]. It is the single largest sector of energy consumption, ranking over other major contributors such as the industrial sector [146]. Advances in technology have paved the way for improvements in CO_2 emissions and fuel economy via consumer shifts toward hybrid and electric vehicles (EVs), as well as innovation in vehicle technology such as turbocharged engines or cylinder deactivation [61]. Meanwhile, as autonomous vehicles (AVs) become increasingly more available in roadways, with the Insurance Institute for Highway Safety predicting there to be 3.5 million vehicles with autonomous capabilities on U.S. roads by 2025 [12], so too do their potential to yield a pronounced effect on the state of traffic, addressing the problem of energy usage from another angle.

This paper focuses on the problem of energy usage in transportation and explores the potential of AVs to alleviate this issue. In this era of mixed autonomy traffic, in which a percentage of vehicles are AVs with special control capabilities, a rich amount of work has been produced that shows that even a small percentage of intelligent agents in traffic are capable of achieving energy savings and traffic reduction via wave dampening or jam-absorption driving of stop-and-go waves and bottlenecks [17, 170, 66, 152]. Many studies have shown how longitudinal control of an AV can significantly impact global and local metrics such as total velocity or fuel economy; for instance, controlled vehicles are capable of completely dissipating traffic waves in a ring setting [112, 38]. To our knowledge, there

have no been no large-scale tests conducted with the goal of smoothing traffic flow.

However, finding an effective way to design controllers for these settings remains an open question due to the partially observed, hybrid, multi-agent nature of traffic flow. In recent years, reinforcement learning (RL) has emerged as a powerful approach for traffic control, leveraging its ability to capture patterns from unstructured data. RL is responsible for producing quality controllers across a wide range of domains, from robotics [56] to mastering gameplay over human experts such as with Starcraft or Go [154, 127]. Within the domain of traffic, RL has been used to derive a variety of state-of-the-art controllers for improving traffic flow metrics such as throughput and energy efficiency [167, 150, 66].

With a particular focus on the real-world impacts of RL on traffic, we discuss the RL-based controller we developed for a stretch of the I-24 highway near Nashville, Tennessee. Our controllers take into account the requirements of real-world deployment, utilizing observations that are accessible via radar and cameras. Despite limited access to the full state space, our RL-based approach achieves notable fuel savings even with low penetration rates.

The contributions of this article are:

- The introduction of a single-agent RL-based controller developed using real traffic trajectories with advanced telemetry-enabled downstream information,
- Numerical results of the controller's performance demonstrating significant fuel savings of over 15% in scenarios exhibiting large-amplitude stop-and-go waves.

7.2 Problem Formulation

In this paper, we consider the problem of decreasing energy consumption in highway traffic by attempting to smooth out stop-and-go waves. These waves are a phenomenon where high vehicle density causes vehicles to start and stop intermittently, creating a wave-like pattern that can propagate upstream and be highly energy-inefficient due to frequent accelerating and braking [132]. We insert a small percentage of autonomous vehicles (AVs) equipped with reinforcement learning-based controllers into the flux of traffic. Leveraging a data-driven, one-lane simulator previously introduced in [87], we simulate real-world highway trajectories. This approach is considerably more time-efficient than comprehensive traffic micro-simulations and is able to partially model the intricate stop-and-go behaviors that occur in traffic, although it overlooks complex dynamics such as lane-changing, which we rectify by incorporating a lane-changing model that we calibrate on data. The following subsections introduce the simulation as well as the different modules that it integrates.

Dataset and Simulation

We use the I-24 Trajectory Dataset [103] introduced in [87] along with a one-lane simulator that replays collected trajectory data. A vehicle, which we call the *trajectory leader*, is placed at the front of the platoon and replays the real-world I-24 trajectory. Behind it, we

place an arbitrary number of AVs and human vehicles (introduced in Sec. 7.2). Typically during training, the platoon behind the trajectory leader consists of one RL-controlled AV, followed by 24 human vehicles. The goal of the AV is to absorb the perturbations in the leader trajectory, so that the energy consumption of the following human vehicles improves compared to the case where the AV is not present.

Instead of training on the whole dataset, we only train on a selected set of four trajectories containing different patterns of waves alternating between low and high speeds. This allows us to optimize directly on the dynamics we are interested in improving, without having training diluted by free-flow trajectories containing no waves for the controller to smooth. We observed that this made training faster while still yielding a controller able to generalize to unseen trajectories. As the trajectories are quite long (between 5000 and 12000 time steps, where a time step is 0.1s), each simulation randomly samples a chunk of size 500 (or 50s) within a trajectory. At evaluation time, we consider a set of six trajectories distinct from the training trajectories and simulate the whole trajectories.

Speed planner

We use real-time data about the state of traffic all along the I-24 in order to equip the RL control with some knowledge about the downstream state of traffic. The data is provided to us by INRIX, which broadcasts it in real time with an approximately 1-minute update interval and a 3-minute delay. The highway is divided into segments of 0.5 miles on average (with significant variance), and the data consists of the average speed of traffic in each segment. For training, we retrieved the INRIX data matching the time when the trajectories were collected on the highway (which includes delay). In the absence of such historical data, one could also generate synthetic traffic data for each dataset trajectory by artificially and averaging the trajectory speeds accordingly.

On top of this raw INRIX data, we use a *speed planner* developed in [45] that provides a profile that the controller should approximately track. The speed planner takes in the data and interpolates on the individual data points to create a continuous and less noisy speed profile of the whole highway. It then uses kernel smoothing to create a *target speed profile*, which is intended as an estimate of the speed to drive at in order to smooth out the traffic waves. However, due to delay and noisy estimates, driving exactly at this speed is insufficient to guarantee smoothing or reasonable gaps. This target speed profile, sampled at different points, is finally fed as an input to our controller, which can be used as an indication of where the free-flow and the congested regions might be.

Energy function

Since we aim to optimize energy consumption, we need a model that can be used to compute consumed energy in our cost functions. We use a model of instantaneous fuel consumption, a polynomial fitted on a Toyota RAV4 model that is similar to the model in [77], but with

updated coefficients. The function depends on the speed of the AV and its instantaneous acceleration, as well as road grade, which we assume to be zero in this work.

Human Driver Behavior

To model human drivers in simulation, we use the Intelligent Driver Model (IDM) [68], a time-continuous car-following model which is widely used in traffic applications. We pick the IDM parameters such that the model is unstable below $18\frac{\text{m}}{\text{s}}$, meaning that stop-and-go waves will propagate backward in the flux of traffic and grow instead of just dissipating. Numerous results demonstrate the string-unstable qualities of human-driver behavior, both via real human drivers in real life and via models such as IDM in simulation [38].

Lane-changing model

We use a lane-changing model to enable more complex multi-lane dynamics for evaluation only. The model consists of a cut-in probability function $P_{\text{in}}(h, v_{\text{lead}})$ and a cut-out probability function $P_{\text{out}}(v_{\text{lead}})$, where h is the space gap and v_{lead} is the speed of the leading vehicle. Both are piecewise second-order polynomials whose coefficients were calibrated using data collected on the I-24 highway, which has not been published yet. At each time step t, and for each ego vehicle in the simulation, P_{in} gives the probability that a vehicle cuts in front of the ego vehicle, while P_{out} gives the probability that the leading vehicle cuts out. If a cut-in happens, a vehicle is inserted such that the ratio between the space gap of the inserted vehicle and the space gap of the ego vehicle after the cut-in follows a normal distribution (also fit to data), clipped to ensure safety after insertion. This model lets us measure the robustness of the control, allowing human vehicles to cut in front of the AV as it tries to open larger gaps to smooth out traffic waves.

7.3 Controller Design

In this section, we formally define the problem in the context of reinforcement learning and discuss the structure and design of the controller.

Defining the POMDP

We use the standard RL formalism of maximizing the discounted sum of rewards for a finite-horizon partially-observed Markov decision process (POMDP). We can formally define this POMDP as the tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma, \Omega, \mathcal{O})$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ represents the set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ represents the conditional probability distribution of transitioning to state s' given state s and action a, $R : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1]$ is the discount factor used when calculating the sum of rewards. The final

two terms are included since the state is hidden: Ω is the set of observations of the hidden state, and $\mathcal{O} : S \times \Omega \to \mathbb{R}$ represents the conditional observation probability distribution.

Observation space

The primary observation space (at time t) consists of the ego vehicle speed v_t^{av}, the leader vehicle speed v_t^{lead}, and the space gap (bumper-to-bumper distance) h_t between the two vehicles. (Note that all distances are in m, velocities in $\frac{\text{m}}{\text{s}}$, and accelerations in $\frac{\text{m}}{\text{s}^2}$.) Two gap thresholds are also included: $h_t^{\min}$, which is the failsafe threshold below which the vehicle will always brake, and $h_t^{\max}$, the gap-closing threshold above which the vehicle will always accelerate. We also include the history of the ego vehicle's speed over the last 0.5 seconds. Finally, the observation space includes traffic information from the speed planner. This consists of the current target speed v_t^{sp}, as well as the target speeds 200m, 500m, and 1km downstream of the vehicle's current position. Note that the AV only observes its leading vehicle and that there is no explicit communication between AVs. All observations provided to the RL agent are rescaled to the range $[-1, 1]$.

Action space

The action space consists of an instantaneous acceleration $a_t \in [-3, 1.5]$. After the RL output, gap-closing and failsafe wrappers are then enforced. We define the gap-closing wrapper such that $h_t^{\max} = \max(120, 6v_t^{\text{av}})$, meaning that the AV will be forced to accelerate if its space gap becomes larger than 120m or its time gap larger than 6s. This result is then wrapped within a failsafe, which enforces safe following distances and prevents collisions. We define the time to collision Δ_t^{TTC} of the ego and lead vehicles as:

$$\Delta_t^{\text{TTC}} = \begin{cases} \dfrac{h_t}{v_t^{\text{diff}}} & \text{if } v_t^{\text{diff}} > 0 \\ +\infty & \text{otherwise} \end{cases}$$

$$v_t^{\text{diff}} = \left[v_t^{\text{av}} \left(1 + \frac{4}{30} \right) + 1 \right] - v_t^{\text{lead}}$$

where the AV velocity is slightly exaggerated to ensure robustness at both low and high speeds. The actual numbers are chosen heuristically: for instance, if both AV and leader drive at $30\frac{\text{m}}{\text{s}}$, the failsafe will ensure a minimum gap of 30m. The failsafe triggers if the time to collision ever goes below 6 seconds, in which case the RL output is overridden and the vehicle will brake at its maximum allowed deceleration. We thus have $h_t^{\min} = 6v_t^{\text{diff}}$. The final RL acceleration is given by:

$$a_t^{\text{out}} = \begin{cases} -3 & \text{if } \Delta_t^{\text{TTC}} \leq 6 \quad (\Leftrightarrow h_t \leq h_t^{\min}) \\ 1.5 & \text{if } \Delta_t^{\text{TTC}} > 6 \text{ and } h_t \geq h_t^{\max} \\ a_t & \text{otherwise} \end{cases}$$

which is further clipped to ensure that the speed v_t^{av} remains within the boundaries $[0, 35]\frac{\text{m}}{\text{s}}$. Note that the free-flow behavior due to the gap-closing wrapper will be to drive at the speed limit in the absence of a leader.

Optimization criterion

At the core, we aim to minimize the overall energy consumption of the traffic. However, as sparse rewards are harder to optimize, we employ proxies that can be minimized at each time step. We mainly aim to minimize the instantaneous energy consumption of the AV and a platoon of vehicles behind it. For comfort, and as another proxy for energy savings, we also minimize squared acceleration amplitudes. Since optimizing for energy and acceleration can be done by stopping or maintaining unreasonably large or small gaps for comfort, we penalize gaps outside of a certain range; this also penalizes the use of failsafe and gap-closing interventions. To further discourage large gaps within this allowed range, the final term adds a penalty proportional to the time gap (space gap divided by ego speed). This is formalized as the reward function r_t, which is given by:

$$\begin{aligned} r_t = &- c_1 \frac{1}{n} \sum_{i=1}^{n} E_t^i - c_2 (a_t^{\text{out}})^2 - c_3 \mathbb{1}\left[h_t \notin [h_t^{\min}, h_t^{\max}]\right] \\ &- c_4 \frac{h_t}{v_t^{\text{av}}} \mathbb{1}\left[h_t > 10 \wedge v_t^{\text{av}} > 1\right] \end{aligned}$$

where E_t^i is the instantaneous energy consumption of vehicle i at time t, where index $i = 1$ corresponds to the AV, and indexes $i = 2$ to $i = n$ correspond to the following $n - 1$ IDM vehicles. $\mathbb{1}$ is defined such that $\mathbb{1}[\mathcal{P}] = 1$ if $\mathcal{P}$ is true, 0 otherwise.

Training algorithm

We use single-agent Proximal Policy Optimization [122] (PPO) with an augmented value function as our training algorithm. PPO is a policy gradient algorithm, a class of RL techniques that optimize for cumulative, discounted reward via shifting the parameters of the neural net directly. More explicitly, policy gradient methods represent the policy as $\pi_\theta(a|s)$, where θ are the parameters of the neural net.

During training, we give additional observations that are available in simulation as an input to the value network. This includes the cumulative miles traveled and gallons of gas consumed, which allows for estimating energy efficiency. This augmented observation space also includes the following metadata: the size of the finite horizon, an identifier for the specific trajectory chunk being trained on, and the vehicle's progress (in space and time) within this chunk. The additional information removes some of the partial observability from the system, allowing the value function to more accurately predict the factors that influence reward.

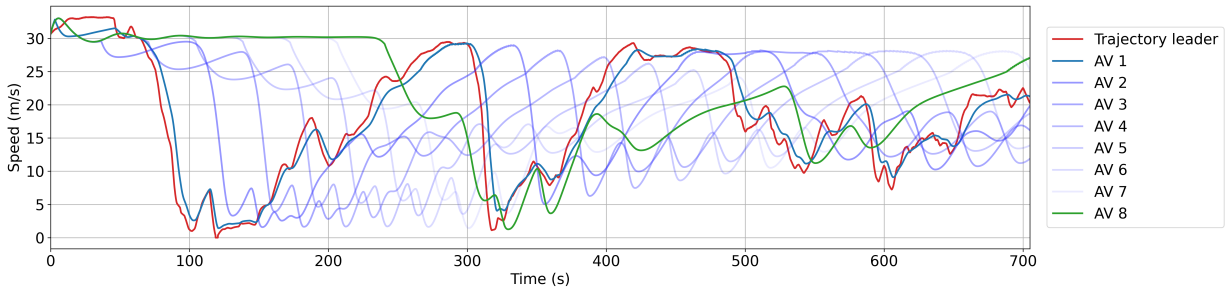


Figure 7.1: Evolution of the speed of the trajectory leader and the AVs in a platoon of 200 vehicles. The 8 AVs are equally spaced at a 4% penetration rate. The first AV in the platoon is shown in blue, the following ones are displayed by decreasing opacity and the last one is in green, demonstrating the smoothing effect of the AVs on the leader trajectory. In particular, one can see how the first AV (in blue) already smoothes the trajectory leader (in red), doesn't slow down as much or accelerate as fast, and thus saves energy.

Experiment details

We run the PPO experiments using the implementation provided in Stable Baselines 3[1] version 1.6.2. We train the model for 2500 iterations, which takes about 6 hours on 90 CPUs. We use a training batch size of 9000, a batch size of 3000, a learning rate of $3 \cdot 10^{-4}$ and do 5 epochs per iteration. The simulation horizon is set to 500, and we do 10 simulation steps per environment step, ie. each action is repeated 10 times. Given that the simulation time step is $dt = 0.1$s, this means that the action changes only every second during training. This allows us to artificially reduce the horizon so that it only is 50, meaning that each training batch contains 100 simulation rollouts. The agent's policy is modeled as a fully-connected neural network with 4 hidden layers of 64 neurons each and tanh linearities, with continuous actions. The value network has the same architecture as the policy network. We set the discount factor γ to 0.999, the GAE value λ to 0.99, and the other training and PPO parameters are left to their default values.

We train with a platoon of $n = 25$ vehicles (not including the leading trajectory vehicle) consisting of one AV followed by 24 IDM vehicles. For our reward function, we used coefficients $c_1 = 0.06$, $c_2 = 0.02$, $c_3 = 0.6$, $c_4 = 0.005$. Both the model parameters and training hyperparameters are determined through grid search, with each experiment conducted using 4 to 8 distinct random seeds to overcome instances of the agent getting trapped in local optima.

[1] `https://github.com/DLR-RM/stable-baselines3`

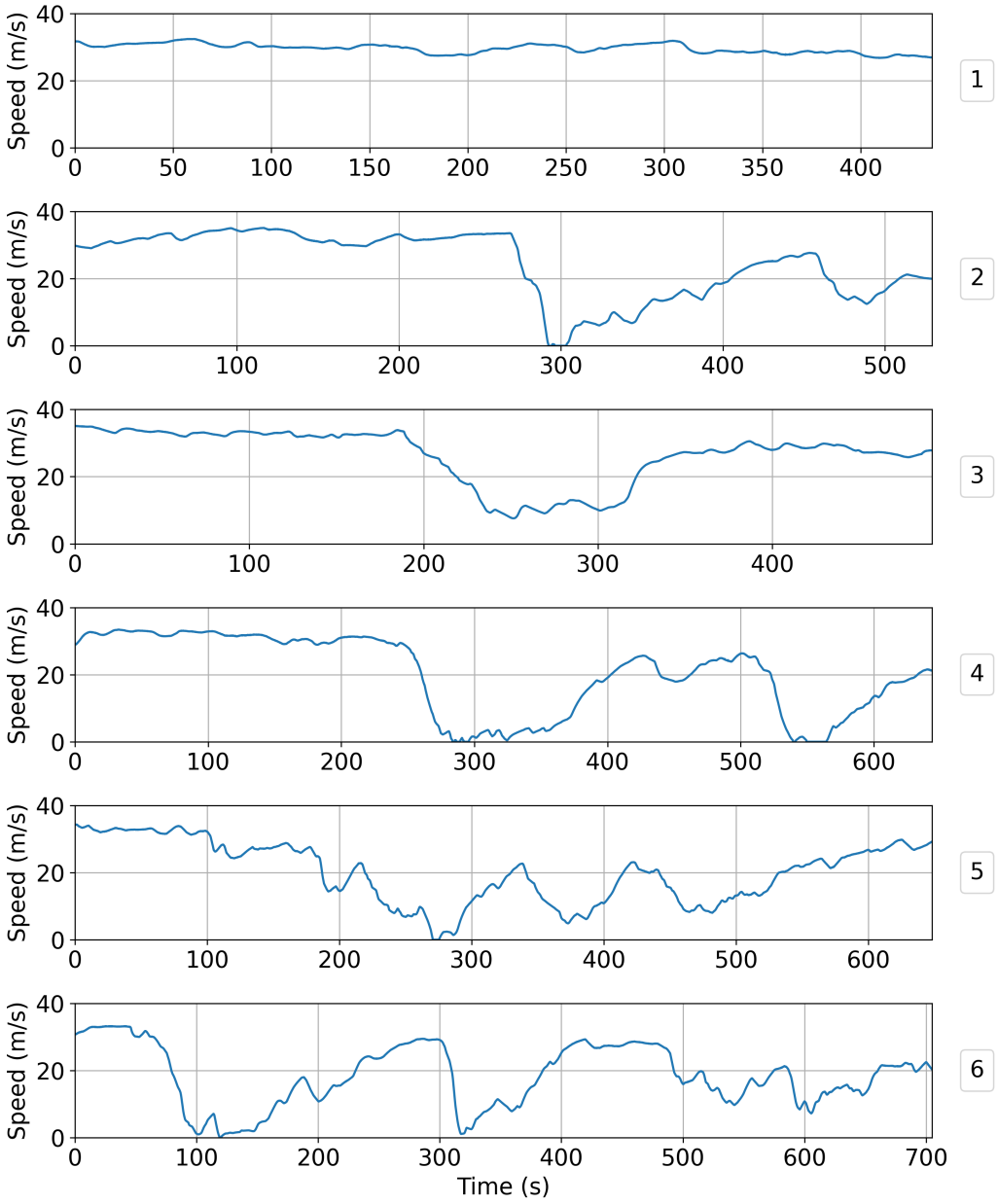


Figure 7.2: Trajectories used for evaluation, numbered from 1 to 6. The first one corresponds to free flow, while the five others contain both low and high speeds, including sharp breaking, sharp accelerating or stop-and-go behaviors.

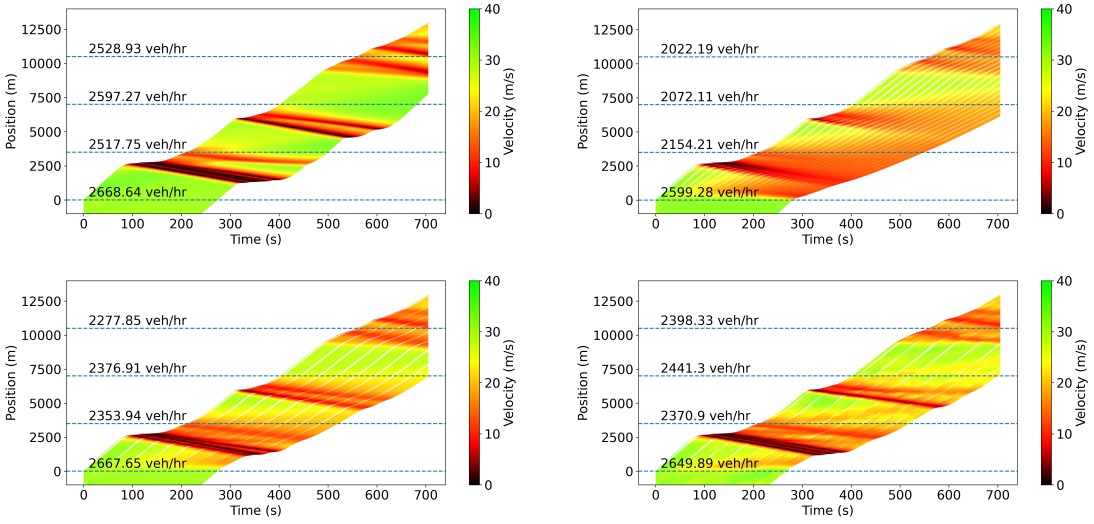


Figure 7.3: Time-space diagrams, each representing a simulation of 200 vehicles. Each vehicle trajectory is plotted as a line in position-by-time space, with color representing the speed of that vehicle. One can observe the wave-smoothing effect of the RL-controlled AVs over time. Horizontal lines display the throughput of the traffic flow at that particular position. Also note that as a warm-up, all AVs behave as humans when their position is negative. **Top left:** all 200 vehicles are IDMs. Traffic waves are illustrated by the red and black colors, while bright green represents free flow. **Top right:** 20 equally-spaced RL-controlled AVs (10% penetration rate, 1 AV every 10 vehicles). One can see the larger gaps opened by the AVs as the white lines between platoons. **Bottom left:** 8 equally-spaced RL-controlled AVs (4% penetration rate, 1 AV every 25 vehicles). **Bottom right:** 8 equally-spaced RL-controlled AVs (4% penetration rate) with the lane-changing model enabled (note that it is disabled in all 3 other subfigures).

7.4 Results

In this section, we analyze the performances of our RL controller in simulation, in terms of energy savings, wave-smoothing, behavior and robustness.

The smoothing effect of the AVs is illustrated in Fig. 7.1, where one can see the speeds of all the AVs in a platoon of 200 vehicles as a function of time on trajectory 6 (see Fig. 7.2). One can observe how the speed profiles become smoother and smoother after each AV.

Fig. 7.3 illustrates the smoothing performed by the RL agents on trajectory 6 in a different way. The top-left time-space diagram shows that the humans don't smooth any waves; on the contrary, they even create some due to the string-unstable nature of the IDM we use. At a 10% penetration rate, most of the waves get smoothed out, less at 4%, and even less with lane-changing. However, in all 3 cases, the diagrams clearly demonstrate the improvement over the baseline. As expected, AVs opening larger gaps also leads to decreased throughput,

Index	10% w/o LC	10% w/ LC	4% w/o LC	4% w/ LC
1	+7.33%	+8.39%	+4.29%	+6.12%
2	+10.87%	+12.63%	+7.04%	+9.42%
3	+14.65%	+14.48%	+9.02%	+7.23%
4	+15.02%	+13.19%	+9.23%	+8.54%
5	+22.58%	+15.77%	+17.05%	+8.37%
6	+28.98%	+18.55%	+19.96%	+15.40%

Table 7.1: We run simulations with 200 vehicles, and show the improvement in system MPG when we control 10% (Left) or 4% (Right) of the vehicles, compared to when all vehicles behave as IDMs, with and without lane-changing in both cases. The trajectories used for evaluation can be seen in Fig. 7.2, with corresponding indexes.

and the best throughput is achieved when the lane-changing model is enabled and human vehicles fill in the gaps. This comes down to a trade-off between throughput reduction and energy savings, which can be tuned by varying the penetration rate.

Table 7.1 shows the energy savings that our controller achieves on the evaluation trajectories (shown in Fig. 7.2) when deployed on AVs at two different penetration rates, with and without lane-changing enabled. The percentages correspond to how much the average system miles-per-gallon (MPG) value increases when AVs use our RL controller, compared to when they all behave as humans. The average MPG is defined as the sum of the distances traveled by all the vehicles in the simulation, divided by the sum of their energy consumptions.

We can observe the energy savings varying a lot depending on the trajectories, which is expected since trajectories that are mostly free flow (like trajectory 1) cannot be improved much, while one with a lot of stop-and-go waves (like trajectory 6) has a lot of potential to be smoothed. As one can expect, energy savings decrease as the AV penetration rate decreases or as lane-changing is enabled, but even at 4% penetration and with lane-changing, the controller reduces the energy consumption on trajectory 6 by over 15%, while only reducing throughput by 5%.

We also note that the controller appears robust to not having access to the speed planner. For example at a 4% penetration rate and without lane-changing, the control achieves +16.87% energy improvement on trajectory 6 without the speed planner (compared to +19.96% with the speed planner), and the trend is similar on the other trajectories.

Finally, in Fig. 7.4, we illustrate the gaps opened by the AV on trajectory 6, along with the failsafe and gap-closing thresholds. The gap-closing threshold allows the AV to open larger gaps and consequently absorb abrupt braking from its leader while ensuring that these gaps are not overly large. As can be expected, we have observed that the larger the maximum gap we allow, the better the AV performs in terms of energy savings. However, a larger maximum gap is usually accompanied by a decrease in throughput, which is again a trade-off. The failsafe threshold mostly ensures safety and comfort for the driver, although

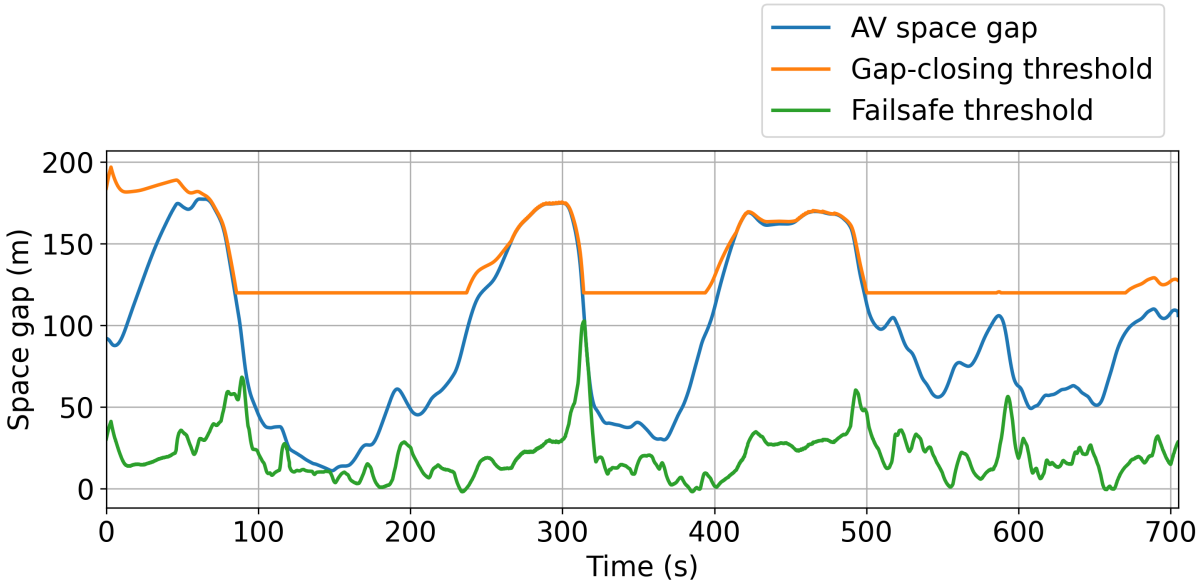


Figure 7.4: Space gap of the first AV in the platoon (in blue) by time, on the same scenario as in Fig. 7.1. The orange line shows the gap-closing threshold $h_t^{\max}$ and the green line shows the failsafe threshold $h_t^{\min}$, introduced in Sec. 7.3.

it is worth noting that when deploying the controller, we integrate an additional explicit safety wrapper, as detailed in [87].

7.5 Conclusion

In this work, we developed RL policies that incorporate both accessible local observations and downstream traffic information, achieving substantial energy savings in simulation. There are several avenues for future research. Given that all the training is conducted in a simulated environment, it would be beneficial to train the agent with more realistic dynamics by enhancing the accuracy of the various models that make up this simulation, like human driving behavior, lane-changing dynamics, and energy consumption metrics. Additionally, it would be interesting to explore multi-agent RL to help the model be robust to interactions between AVs and potentially enable cooperation between them.

While our simulation process has a distinct speed advantage over large micro-simulators, it could benefit significantly from vectorization. Moreover, despite our technical capacity to deploy our controller safely onto a real-world vehicle, gathering results is challenging and necessitates further field tests. Another direction of research we are exploring consists of training and deploying adaptive cruise control (ACC)-based controllers, where the policy outputs a desired set-speed instead of an acceleration. By design of the ACC, the control

would be safe and smooth, and easily deployable at a large scale simply by augmenting the onboard ACC algorithm to use the RL control as a set-speed actuator.

Chapter 8

Reinforcement Learning Based Oscillation Dampening: Scaling up Single-Agent RL algorithms to a 100 AV highway field operational test

8.1 Introduction

As the automotive industry continues to evolve, the quest for safer and more efficient self-driving cars has become a top priority. To achieve this, engineers and researchers are turning to state-of-the-art techniques like reinforcement learning to create intelligent control systems that can navigate complex environments with unprecedented precision and adaptability. Reinforcement learning (RL), a subfield of machine learning, offers a promising avenue for training automated vehicles to make optimal decisions and actions in real-time scenarios. Unlike traditional rule-based approaches, RL-based controllers learn through trial and error, progressively refining their behavior based on feedback from the environment. This ability to adapt and improve over time makes RL an invaluable tool for enhancing the performance and reliability of automated vehicles.

Together with the other articles presented in this special issue, this article constitutes an element of the technical work involved in bringing together objective of the CIRCLES project [59]: the MegaVanderTest (MVT), the largest deployment of automated vehicles (AVs) designed to smooth traffic flow in history as of 2023. These AVs are partially automated and limited to longitudinal control. They do not communicate between each other, but communicate with a central server to get information about the downstream state of the highway. This article discusses the following:

- Background material, including a short overview of reinforcement learning, human driver models, and policy gradient algorithms.

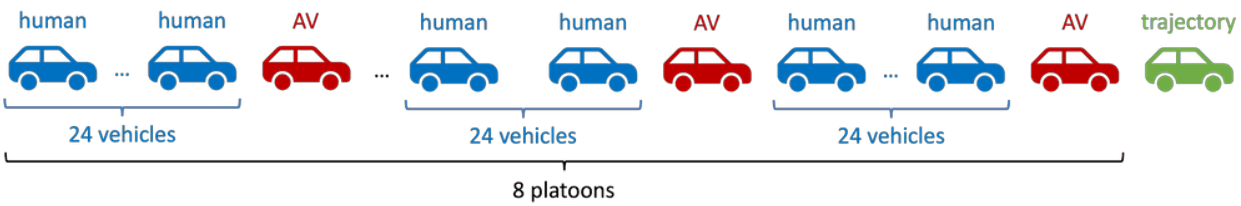


Figure 8.1: Example setup of the trajectory simulator for evaluation. One vehicle, which we name *trajectory leader*, replays a velocity trajectory from the I-24 Trajectory Dataset [104]. Following it are a combination of IDM-controlled human vehicles and automated vehicles (AVs) all on a single lane. This evaluation setup contains 8 platoons, each consisting of one AV followed by 24 human vehicles. The human vehicles are used to assess the smoothing performances of the AV. During training, only one platoon is simulated. © 2024 IEEE.

- Problem formulation of the two classes of RL controllers designed for this experiment: (1) Acceleration-based control and (2) Adaptive Cruise Control (ACC)-based control.
- Description of the work moving from the software to hardware platform, deployment onto real roadways, and the MVT field test week.
- Result analysis of all controllers in simulation and in deployment

8.2 Background

Reinforcement Learning

We adhere to the conventional reinforcement learning (RL) framework [134, 89], which seeks to optimize the discounted cumulative rewards within a finite time frame for a Partially Observable Markov Decision Process (POMDP) [33].

For a formal description and background of RL, refer to Chapter 2.

Human Driver Models

In order to train RL controllers, we simulate mixed-autonomy traffic [141] where RL-controlled AVs interact with human-driven vehicles [165, 166, 150]. In order to model human car-following behavior, we use the Intelligent Driver Model (IDM) [68, 143] to govern a vehicle's longitudinal motion according to its leading vehicle. To ensure authenticity, IDM parameters are selected to mimic key non-equilibrium features of real-world traffic, particularly dynamic instabilities (phantom traffic jams) with realistic wave growth and propagating stop-and-go waves caused by human dri ving behavior [164, 145] or even ACC [58]. To trigger dynamic instabilities that the model possesses for suitably congested densities, zero-mean Gaussian

noise is added to the acceleration in each time step. Additional safety measures are included to maintain acceleration bounds and to avoid collisions—note that pure IDM is mathematically collision-free, but discretized time-stepping and heterogeneous vehicle composition yields potential collision opportunities.

The IDM [143] prescribes the acceleration of vehicle α as a function of its space gap (bumper-to-bumper distance to its leader vehicle) s_α; its speed v_α; and relative speed with the leader, Δv_α:

$$a_{\text{IDM}} = \frac{dv_\alpha}{dt} = a\left[1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right], \tag{8.1}$$

where s^* is the desired space gap and is given by:

$$s^*(v_\alpha, \Delta v_\alpha) = s_0 + \max\left(0, v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}\right), \tag{8.2}$$

where s_0, v_0, T, δ, a, and b are known parameters. For further details on car following models and IDM, refer to Chapter 2. Parameter values are in Table 8.2.

MegaController / Speed Planner

The *MegaController* (see [79]) is the outermost framework that addresses the challenge of smoothing traffic flow despite a limited penetration rate of automated vehicles by introducing two key components: the algorithm and the speed planner. The *Speed Planner*, discussed in further detail in its own standalone article [157], is a control framework for dynamic speed advisories in a traffic environment in which both automated and human-driven vehicles coexist. The speed planner is a centralized unit hosted on a server which incorporates various algorithms to handle computationally intensive tasks. The algorithms deployed on individual vehicles serve as operators, following the instructions provided by the centralized planner. The primary function of the Speed Planner is to design target speed profiles with the objective of minimizing vehicle energy consumption and maximizing overall traffic flow efficiency. The implemented Speed Planner is dependent on INRIX, a data source which provides the realtime average speed of road segments globally, including the segments of the *Interstate 24* (I-24) that the RL algorithm is deployed on. INRIX data are typically represented in 500-800m segments, with high variability in segment size; approximately three minutes of latency; and is aggregated over all lanes across 60 seconds. Thus, while representing imperfect data, INRIX data provides a solid foundation upon which to develop prediction modules.

Leveraging Field Operational Test Data to Build a Fast Traffic Simulator

Creating a robust and representative simulation environment is crucial for the successful implementation of wave-smoothing controllers in automated vehicles, ensuring their efficient

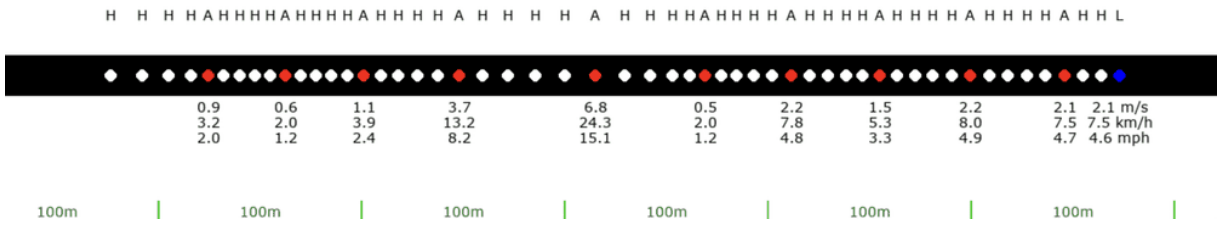


Figure 8.2: A render from the simulator. Each circle represents one vehicle. The leading blue vehicle (L) replays a trajectory from the dataset [105], the white vehicles (H) use IDM behavior and the red vehicles (A) are AVs. In this screenshot we can observe two large waves propagating through the traffic. © 2024 IEEE.

operation across diverse traffic conditions. While we have previously employed large traffic micro-simulations in the past (see Chapter 3, based on SUMO), they require careful and extensive calibration of factors such as network parameters, road conditions, and both lane-changing and longitudinal driving models. This is essential to capture the intricate dynamics of the targeted highway, especially the nuances of wave dynamics. One other major limitation is that these simulations can be time-consuming, given the hundreds or thousands of vehicles interacting.

Prioritizing speed and efficiency for fast turnover in training new RL algorithms, we built a new simulator for CIRCLES. This simulator is built on real human driver data collected from the I-24 highway [104] [105], enabling it to mirror real-world, albeit simplified, wave dynamics. This approach, assuming the existence of trajectory data for the highway of interest, not only does not require any calibration but also allows for very fast simulations. Consequently, it has since been extensively leveraged for the design, training and evaluation of the different types of controls in CIRCLES.

Data Acquisition

Trajectory data is recorded [29] on a 14.5-kilometer segment of I-24, near Nashville, Tennessee, using an instrumented vehicle. The data includes vehicle Controller Area Network (CAN) data and GPS information, capturing speed, relative speed of the lead vehicle, acceleration, and space gap (distance from front bumper to rear bumper of leading vehicle).

Dataset Analysis

The trajectory dataset covers a wide range of traffic conditions, including stationary congested traffic and high-speed free-flow traffic, with diverse acceleration patterns. Training data retain both low and high-speed instances to ensure competent controller behavior during

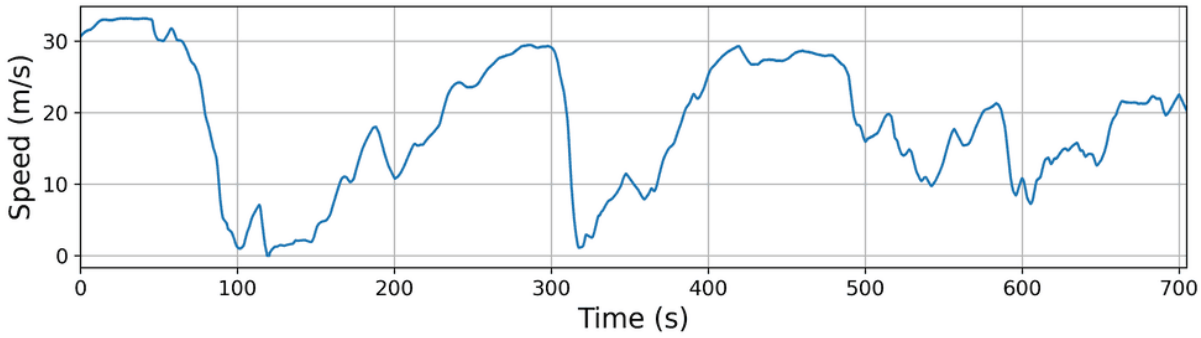


Figure 8.3: Speed vs. time for one of the dataset trajectories (corresponding to the ego trajectory of one of our drivers), exhibiting large acceleration and breaking patterns that can typically lead to stop-and-go waves. © 2024 IEEE.

transitions. An example dataset trajectory containing both low and high speeds is displayed in Figure 9.1.

Training and Evaluation Framework

The simulation employs a single-lane environment where a mixed platoon of *human* vehicles and automated vehicles (AVs) follow a real-world trajectory from the dataset. A render from the simulator can be seen in Figure 8.2. Human vehicles use the Intelligent Driver Model (IDM) (see Sec. 8.2) with string-unstable parameters to ensure realistic driving dynamics and wave propagation in congestion. The simulator achieves high efficiency, allowing faster training compared to micro-simulation, and sidesteps the need for extensive calibration of large micro-simulations. A data-based lane-changing model was also developed for the simulator, allowing for assessing the robustness of the different controls to lane-changes.

Energy Models Compatible with Optimization

Vehicle energy models that accurately reproduce the dynamics of a specific vehicle tend to be complex and possess discontinuities (for instance at gear shifts). This renders their use in energy optimization challenging, because optimizers may get trapped in local minima that are exploiting the gear shifting characteristics of one specific vehicle. However, for real traffic composed of similar but not identical vehicles, such behavior is undesirable. Thus, optimization-friendly energy model frameworks are needed in the design and training of controllers.

Figure 8.4 shows how an energy model with realistic gear shifting will have the tendency to yield unsteady profiles (pink profile) that are specific to one vehicle, rather than producing smooth steady driving (red profile) that is more desirable for traffic composed of different vehicles.

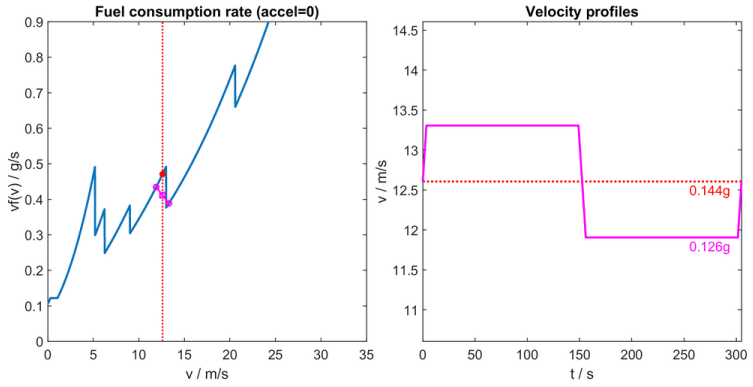


Figure 8.4: Fuel consumption rate as function of speed (for zero acceleration) of an energy model with discontinuities due to gear shifts (left), showing that constant speed driving is not fuel-optimal driving (right). © 2024 IEEE.

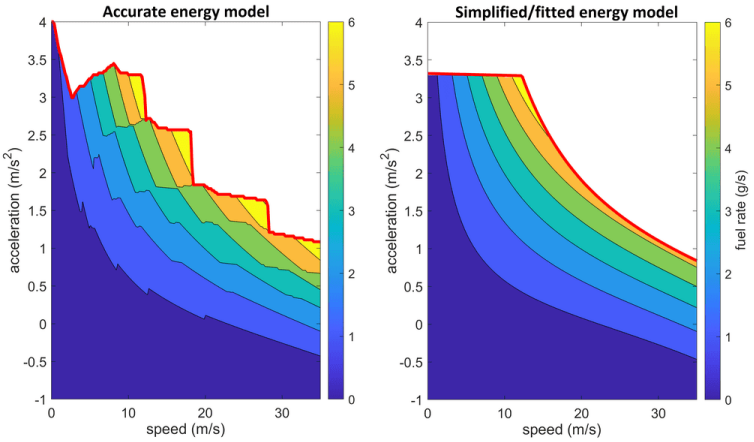


Figure 8.5: Contours of (a) a vehicle-specific energy model with discontinuities at gear shifts, and (b) the fitted/averaged simplified energy model that has desirable convexity properties. © 2024 IEEE.

To address these needs, energy models are developed and used [69] that are of the form $f(v, a, \theta)$ where the fuel consumption rate function f possesses desirable "convexity" properties (v is speed, a acceleration, and θ the road grade). The models average out the non-convex behavior of the vehicle-specific accurate models via carefully constructed fitted polynomial functions. Besides yielding a significant computational speed-up, these simplified models also are more representative of an average energy profile over many makes and models from a given vehicle class.

Figure 8.5 compares the fuel rate contours of an energy model with discontinuities at gear shifts (left) against the contours of the simplified model used here (right), showing the

absence of the gear shift jumps in the latter. The fitted energy function is not convex in the sense of having a positive definite Hessian matrix (vehicle energy models are very far from possessing such a structure), but rather: for a fixed road grade, f is convex in the a-direction for a constant v, and f/v is convex with respect to v for constant a (with a minor exception of the fuel-cut boundary). Mathematically, this notion of "convexity" provides precisely the properties that imply that constant speed solutions are equivalent to fuel-optimal trajectories (under reasonable constraints).

8.3 Simulation / Problem Formulation

In this article, we explore the avenues in which AVs can improve upon fuel economy for the entirety of the vehicles on the road, and not just for themselves. One effective focus to achieve this goal is targeting stop-and-go waves. Stop-and-go waves are a phenomenon in which high density traffic can cause vehicles to stop and restart without any apparent reason. This phenomena is commonly experienced in all kinds of driving scenarios, including highways and local roads. They are easy to form, as shown in a variety of literature and past experiments [132, 38]. We use AVs as a tool in attempting to solve this problem by leveraging their understanding of human-driving and their access to nonlocal traffic information, thus improving driving efficiency for themselves and for surrounding vehicles. The success of this approach in contained scenarios has been documented in past work [112], in which a single AV can be shown to impact traffic conditions significantly, seeing improvements of up to 49% in a ring road setting [168].

Prior to the MVT, we conducted an experiment with 4 RL-controlled AVs deployed on the I-24 [87]. Energy savings of about 11% were obtained in simulation, with AVs having the most impact at lower speeds, where waves are more likely to occur. Besides, AV trajectories and their corresponding leader trajectories were replayed in simulation, demonstrating a low sim-to-real gap and increased energy efficiency.

With the intention of deploying our AVs onto a four-mile stretch of I-24, instead of simulating the traffic scenario on a simulation remake of the I-24 (which would be both highly complex and time-inefficient), algorithms were trained using a data-driven, one-lane simulator introduced in [87]; see also in 8.2.

A simulation environment was established based on human driver data [104] gathered on a segment of the I-24 highway southeast of Nashville, Tennessee. This data, collected from an instrumented vehicle, included details such as vehicle speed, leader speed, instantaneous acceleration, and space gap. The collected data varied in factors such as the time of day, day of the week, direction of travel, and levels of traffic congestion. Controller Area Network (CAN) data and GPS data from the drives are stored in csv files. The data were then refined by extracting relevant data from the CAN file [24], adjusting it to match the GPS time, and downsampling and interpolating it to align with the GPS time. The resulting dataset [104] encapsulates a wide range of traffic conditions from stationary congested traffic to maximum speed free-flow traffic, including many stop-and-go driving patterns. An example speed pro-

file of such a trajectory is shown in 8.2. This allows us to simulate realistic driving dynamics while avoiding the computational cost of a full micro-simulation of the highway. The training and evaluation framework was developed using a single-lane training environment where an AV and human-driven vehicles follow the trajectory data recorded from human drivers, and as shown in [87], the simulation to reality gap is minimal. Human-driven vehicles in the simulation are modeled using IDM [6], using string-unstable hyperparameters so that the vehicles in the platoon are able to propagate the wave patterns that the AV attempts to smooth out. Figure 8.1 shows an example setup of the simulation. Furthermore, we equip the simulator with a data-driven lane-changing model in order to assess the robustness of the controls to cut-ins and cut-outs. Some trajectories were used for training, different ones were kept for evaluation, and real-world deployment served as the test set.

Initial controller ideas sprouted from previous work [87, 29] on a vehicle which supported acceleration-based control. The opportunity to get access to an additional vehicle-type which supported ACC-based control, prompted a pivot from our initial idea of an acceleration-based controller to also develop and support an ACC-based controller. Discussion and results will be provided on the performance and efficacy of each controller. Ultimately, of the two RL controllers presented here, the ACC-based controller was deployed for the MVT and is the controller whose results are presented here. In addition, we present the results of two different alternative algorithms that were developed during CIRCLES but were not ultimately deployed.

In this article and the following sections, we discuss the development and results of two controllers:

1. The acceleration-based controller
2. The ACC-based controller

Common Problem Formulation

Due to the considerable overlap in algorithm design between the acceleration-based controller and the ACC-based controller, the parameters that are common to both controllers will be discussed here.

We use single-agent *Proximal Policy Optimization* [122] (PPO) with an augmented value function as our training algorithm. PPO is a well-known policy gradient algorithm known for sample efficiency and performance. For more background on the technical details of policy gradient algorithms, view Chapter 2.

The energy models that build up the MPG-based reward function described in Section 8.3 and Table 8.3 that are used to train RL controllers are discussed in Section 8.2 and [69].

Both the acceleration-based and ACC-based controllers use the same IDM parameters, as shown in Table 8.2 and adhere to the dynamics prescribed by 8.1.

	Speed Planner A						Speed Planner B					
BOTTLENECK	Fuel Economy		Throughput		Speed		Fuel Economy		Throughput		Speed	
	Val	Pct	Val	Pct	Val	Pct	Val	Pct	Val	Pct	Val	Pct
Baseline	33.61	0.00%	524.75	0.00%	11.64	0.00%	33.61	0.00%	524.75	0.00%	11.64	0.00%
Microaccel A	35	3.97%	593.81	13.16%	12.09	3.87%	35.34	4.90%	591.78	12.77%	12.13	4.21%
Microaccel B	33.41	-0.60%	593.81	13.16%	12.57	7.99%	34.23	1.81%	595.86	13.55%	12.59	8.16%
Microaccel C	35	3.97%	593.81	13.16%	12.09	3.87%	35.34	4.90%	591.78	12.77%	12.13	4.21%
Microaccel D	35	3.97%	593.81	13.16%	12.09	3.87%	35.34	4.90%	591.78	12.77%	12.13	4.21%
Microaccel E	35	3.97%	593.81	13.16%	12.09	3.87%	35.34	4.90%	591.78	12.77%	12.13	4.21%
Microaccel F	33.98	1.09%	589.76	12.39%	12.01	3.18%	34.04	1.26%	593.81	13.16%	12.03	3.35%
RL A	34.81	3.45%	573.56	9.30%	12.18	4.64%	35.1	4.25%	573.56	9.30%	12.16	4.47%
RL B	34.78	3.36%	573.56	9.30%	12.18	4.64%	34.95	3.83%	573.56	9.30%	12.17	4.55%
RL C	33.97	1.06%	571.03	8.82%	12.2	4.81%	33.92	0.91%	591.78	12.77%	12.08	3.78%
High-speed RL	33.91	0.88%	593.81	13.16%	12.01	3.18%	33.91	0.88%	593.81	13.16%	12.01	3.18%
Low-speed RL	36.03	6.72%	602.05	14.73%	12.65	8.68%	36.62	8.22%	600	14.34%	12.58	8.08%
Deployed	34.62		602.09		12.71		34.63		602.05		12.71	
SHOCKWAVE	Fuel Economy		Throughput		Speed		Fuel Economy		Throughput		Speed	
	Val	Pct	Val	Pct	Val	Pct	Val	Pct	Val	Pct	Val	Pct
Baseline	37.92	0.00%	2,038.55	0.00%	40.13	0.00%	37.92	0.00%	2,038.55	0.00%	40.13	0.00%
Microaccel A	42.47	10.71%	2,062.57	1.18%	39.83	-0.75%	41.82	9.33%	2,036.83	-0.08%	39.98	-0.37%
Microaccel B	41.61	8.87%	2,084.85	2.27%	41.28	2.87%	39.73	4.56%	2,043.91	0.26%	43.21	7.68%
Microaccel C	42.47	10.71%	2,062.57	1.18%	39.83	-0.75%	41.82	9.33%	2,036.83	-0.08%	39.98	-0.37%
Microaccel D	42.47	10.71%	2,062.57	1.18%	39.83	-0.75%	41.82	9.33%	2,036.83	-0.08%	39.98	-0.37%
Microaccel E	42.47	10.71%	2,062.57	1.18%	39.83	-0.75%	41.82	9.33%	2,036.83	-0.08%	39.98	-0.37%
Microaccel F	44.58	14.94%	2,045.59	0.35%	40.31	0.45%	43.19	12.20%	2,059.33	1.02%	40.28	0.37%
RL A	43.71	13.25%	2,011.67	-1.32%	39.45	-1.69%	44.25	14.31%	2,015.90	-1.11%	39.44	-1.72%
RL B	42.91	11.63%	2,008.93	-1.45%	39.38	-1.87%	43.28	12.38%	2,016.03	-1.10%	39.38	-1.87%
RL C	37.72	-0.53%	2,040.71	0.11%	40.02	-0.27%	39.19	3.24%	2,043.93	0.26%	40.4	0.67%
High-speed RL	37.39	-1.42%	2,045.98	0.36%	40.19	0.15%	36.9	-2.76%	2,055.17	0.82%	40.28	0.37%
Low-speed RL	46.51	18.47%	2,080.09	2.04%	39.45	-1.69%	49.36	23.18%	2,096.52	2.84%	39.25	-2.19%
Deployed	45.96		2,072.91		39.27		49.22		2,094.57		39.21	
FREEFLOW	Fuel Economy		Throughput		Speed		Fuel Economy		Throughput		Speed	
	Val	Pct	Val	Pct	Val	Pct	Val	Pct	Val	Pct	Val	Pct
Baseline	36.85	0.00%	2,136.49	0.00%	71.9	0.00%	36.85	0.00%	2,136.49	0.00%	71.9	0.00%
Microaccel A	43.11	14.52%	2,247.74	5.21%	63.61	-11.53%	43.1	14.50%	2,246.04	5.13%	63.54	-11.63%
Microaccel B	42.75	13.80%	2,262.47	5.90%	63.54	-11.63%	42.74	13.78%	2,232.41	4.49%	62.75	-12.73%
Microaccel C	43.11	14.52%	2,247.74	5.21%	63.61	-11.53%	43.1	14.50%	2,246.04	5.13%	63.54	-11.63%
Microaccel D	43.11	14.52%	2,247.74	5.21%	63.61	-11.53%	43.1	14.50%	2,246.04	5.13%	63.54	-11.63%
Microaccel E	43.11	14.52%	2,247.74	5.21%	63.61	-11.53%	43.1	14.50%	2,246.04	5.13%	63.54	-11.63%
Microaccel F	43.77	15.81%	2,149.98	0.63%	60.97	-15.20%	43.7	15.68%	2,143.64	0.33%	60.84	-15.38%
RL A	49.39	25.39%	2,193.21	2.65%	55.81	-22.38%	50.67	27.27%	2,189.98	2.50%	53.79	-25.19%
RL B	49.31	25.27%	2,210.67	3.47%	56.41	-21.54%	54.8	32.76%	2,186.25	2.33%	54.8	-23.78%
RL C	36.87	0.05%	2,139.03	0.12%	71.28	-0.86%	36.87	0.05%	2,139.03	0.12%	71.28	-0.86%
High-speed RL	36.84	-0.03%	2,170.76	1.60%	72.09	0.26%	36.78	-0.19%	2,168.24	1.49%	72.21	0.43%
Low-speed RL	49.22	25.13%	2,186.74	2.35%	52.11	-27.52%	52.25	29.47%	2,166.56	1.41%	47.25	-34.28%
Deployed	36.84	-0.03%	2,170.76	1.60%	72.09	0.26%	36.78	-0.19%	2,168.24	1.49%	72.21	0.43%

Table 8.1: Performance results across a variety of metrics. The final speed planner was a combination of Speed Planners 1 and 2. The controllers are tested across the bottleneck and a shockwave scenarios in column 1. The three controllers tested here include (1) the high-speed controller, (2) the low-speed controller, and (3) the final controller described in Section 8.3 is a combination of (1) and (2). Evaluations were computed by running each controller on simulation with two leader trajectories, one in a congested setting and one in a bottleneck setting. Evaluations are run with a 4% penetration rate of AVs. These evaluations were ultimately used to determine which controllers to use during the MVT.

	Intelligent Driver Model						
Parameter	v_0	T	a	b	δ	s_0	ϵ
Value	35	1.24	1.3	2.0	4	2	$\mathcal{N}(0,0)$

Table 8.2: Selected IDM parameters (see Eq. 8.1-8.2). © 2024 IEEE.

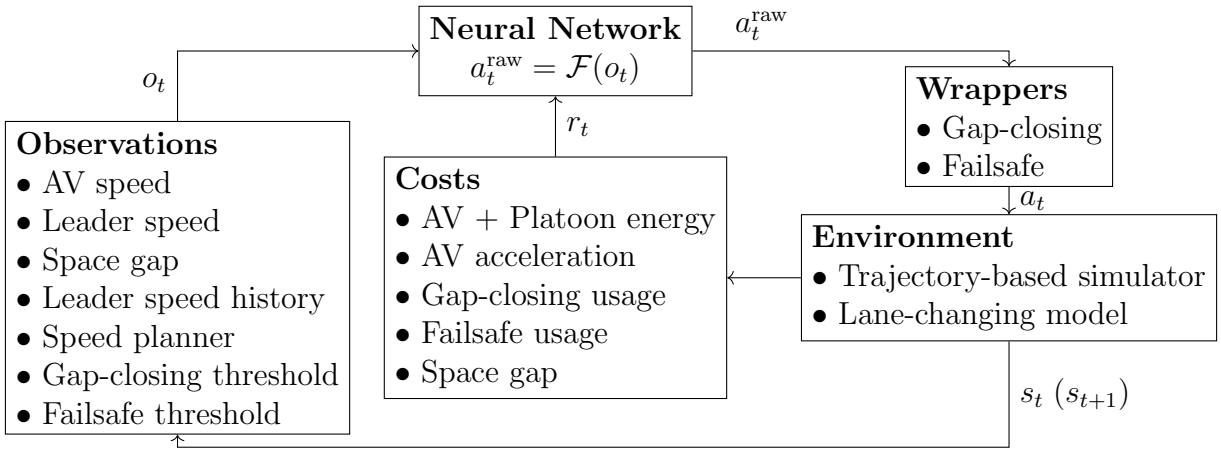


Figure 8.6: Diagram summarizing the design of the acceleration-based controller. The environment is in state s_t, from which we obtain a vector of observations o_t which is fed into the neural network to get a raw acceleration a_t^{raw}. This acceleration is wrapped by a gap-closing and a failsafe term if necessary, the resulting acceleration a_t is applied to the AV and the whole simulation is updated. This leads to a new state s_{t+1}, and the process is repeated. Additionally, the environment computes a reward r_t from the action, which is used to optimize the neural network. © 2024 IEEE.

Acceleration-based RL Problem Formulation

We had the opportunity to gain access to a vehicle-type which could be deployed for the MVT and supported ACC-based control, so we pivoted from our initial idea of an acceleration-based controller to also develop and support an RL ACC-based controller. However, we still discuss problem formulation of the acceleration-based controller and compare simulation performance.

Observation Space

The observation space at time t consists of:

- v_t, AV speed
- v_t^{lead}, AV leader vehicle speed

- h_t, space gap (bumper-to-bumper distance) between AV and leader

The "leader" vehicle is defined to be the vehicle in front of the AV, and can be seen in Figure 8.1. Note that all distances are in m, speeds in m/s, and accelerations in m/s^2. Furthermore, two gap thresholds are also included: $h_t^{\min}$, which is the failsafe threshold below which the vehicle will always brake, and $h_t^{\max}$, the gap-closing threshold above which the vehicle will always accelerate. These thresholds are described in the following action space section. We also include the history of the ego vehicle's speed over the last 0.5 seconds. Finally, the observation space includes traffic information from the Speed Planner. This consists of the current target speed v_t^{sp}, as well as the target speeds 200m, 500m, and 1km downstream of the AV's current position. Note that the AV controller only observes its leading vehicle and that there is no explicit communication between AVs. All observations provided to the RL agent are rescaled to the range $[-1, 1]$.

Action Space

This section contains several numerical values for parameters, whose origin are not explained in the text. They come from vehicle specifications, estimation, modeling and are empirical or design-based. They are shown here for completeness.

The output of the policy is an instantaneous acceleration in the range $a_t \in [-3, 1.5]m/s^2$, chosen to maintain a comfortable driving experience. We assume that the acceleration is instantly achieved when the command is passed to the vehicle. This acceleration output is then wrapped by a gap-closing component and a failsafe component, to ensure and add a soft constraint on safety and reasonable behavior. The gap above which the gap-closing wrapper will activate is $h_t^{\max} = \max(120, 6v_t^{\text{av}})$, resulting in the control being overridden and the AV forced to accelerate if its space gap becomes larger than 120m, or its time gap larger than 6s. 120m is chosen as an arbitrary large space gap value. Note that in the absence of a leader, the space gap is set to some arbitrarily large value and the AV will accelerate. In freeflow, the AV behavior will be to drive at the road speed limit.

The resulting acceleration is further wrapped by a low-gap failsafe, to encourage safe following distances and prevent collisions. Note that this is enacted in simulation to ensure better training behavior; during deployment on the road, many other failsafes are present on the car. We define the time-to-collision (TTC) Δ_t^{TTC} of the ego and leader vehicles at time t as

$$\Delta_t^{\text{TTC}} = \begin{cases} \dfrac{h_t}{v_t^{\text{diff}}} & \text{if } v_t^{\text{diff}} > 0, \\ +\infty & \text{otherwise,} \end{cases} \tag{8.3}$$

where

$$v_t^{\text{diff}} = \left[v_t^{\text{av}} \left(1 + \frac{4}{30} \right) + 1 \right] - v_t^{\text{lead}}. \tag{8.4}$$

In case this modified TTC goes below 6 seconds, the failsafe will trigger, overriding the outputted acceleration and forcing the AV to brake. The failsafe triggering threshold can be

computed $h_t^{\min} = 6v_t^{\text{diff}}$. The reasoning for using $v_t^{\text{av}}\left(1 + \frac{4}{30}\right) + 1$ instead of simply v_t^{av} is that if v_t^{diff} was set equal to $v_t^{\text{av}} - v_t^{\text{lead}}$, and say both the ego and lead vehicles are driving at $30m/s$, then the failsafe triggering threshold would be $h_t^{\min} = 0$ which wouldn't be triggered even if both vehicles maintained a space gap as small as 1m. We cannot allow such a dangerous situation to occur, consequently we artificially increase the AV speed for additional safety precautions. With the new formulation, in our example above the minimal gap would now be $h_t^{\min} = 30$m. This, when driving at 30 m/s, corresponds to a time gap of 1s which is far more reasonable (and is also how we empirically chose the coefficients). Note that the $+1$ term helps to also slightly exaggerate the AV speed when v_t^{av} is very low, ensuring robustness at both low and high speeds.

The final RL acceleration is given by:

$$a_t^{\text{out}} = \begin{cases} -3 & \text{if } \Delta_t^{\text{TTC}} \leq 6 \quad (\text{iff. } \ h_t \leq h_t^{\min}), \\ 1.5 & \text{if } \Delta_t^{\text{TTC}} > 6 \text{ and } h_t \geq h_t^{\max}, \\ a_t & \text{otherwise,} \end{cases} \tag{8.5}$$

which is further clipped to ensure that the speed v_t^{av} remains within the speed limit boundaries $[0, 35]\frac{\text{m}}{\text{s}}$.

Reward Function

The primary objective of the MVT was to improve the overall energy consumption of the stop-and-go traffic. This could be modeled as a reward of 0 for all time steps, except at the last time step where it would be the negative overall fuel economy of all the vehicles (that is, the total miles driven divided by total gallons of gasoline consumed). However, that would lead to a highly sparse reward which is hard to optimize. Instead, we considered proxies that can be optimized at each time step; ultimately we minimize the instantaneous fuel consumption of the AV and of a platoon of human-driven vehicles following it at each time step. Over the course of the simulation, this reward integrates exactly to the total fuel consumed; since the reward is computed over a fixed length of road, it is effectively rewarding the overall fuel economy. For driver comfort, and as an additional proxy for energy consumption, we also minimize the squared instantaneous accelerations. Furthermore, we penalize going outside of the failsafe and gap-closing thresholds—that is, we penalize the controller's output being overridden by the wrappers, to entice the policy not to rely on them. Having these wrappers, or penalizing large gaps in another way (we also penalize time gap), is important to maintain reasonable, socially-acceptable gaps and to avoid the degenerate energy-optimal solution: to simply come to a stop.

This is formalized as the reward function r_t, which is given by:

$$\begin{aligned} r_t = &- c_1 \frac{1}{n} \sum_{i=1}^{n} E_t^i - c_2 (a_t^{\text{out}})^2 - c_3 \mathbb{1}\left[h_t \notin [h_t^{\min}, h_t^{\max}]\right] \\ &- c_4 \frac{h_t}{v_t^{\text{av}}} \mathbb{1}\left[h_t > 10 \wedge v_t^{\text{av}} > 1\right] \end{aligned}$$

where E_t^i is the instantaneous fuel consumption of vehicle i at time t, index $i = 1$ corresponds to the AV, and indexes $i = 2$ to $i = n$ correspond to the following $n - 1$ human-driven vehicles. $\mathbb{1}$ is defined such that $\mathbb{1}[\mathcal{P}] = 1$ if $\mathcal{P}$ is true, 0 otherwise.

Acceleration-based RL Simulation Results

After training, we have a controller that provides acceleration based on AV speed, leader speed, space gap, and other parameters outlined in 8.3. The control combines the trained neural network with the failsafe and gap-closing components detailed in Eq. (8.5).

During the analysis phase, we extensively explore the dynamics of a leading vehicle replaying a preset trajectory from a dataset, trailed by a series of platoons. Each platoon comprises a single AV (either IDM-controlled or RL-controlled) followed by 19 IDM-controlled human vehicles, which translates to a 5% penetration rate of AVs.

Figure 8.7 portrays the speed trends of the trajectory leader and the AVs in the first four platoons. While the remaining vehicles are not explicitly shown for clarity, they follow a consistent trend with the leading AVs. A significant observation around the $t = 300$s mark is the trajectory leader's momentary stop at the trough of the wave. In contrast, successive AVs not only avoid halting but also never decelerate to speeds as low as the preceding vehicle. This pattern repeats in the subsequent wave's crest, where the AVs do not reach speeds as elevated as the leader. Such behavior of the AVs demonstrates a pronounced smoothing effect, which diminishes the speed variance.

Further analysis, as seen in Figure 8.8, provides an aggregated metrics overview throughout the simulation relative to the vehicle ID. Every subsequent AV manifests a decrease in speed standard deviation, corroborating the dampening of waves as intuited from Figure8.7. Furthermore, these AVs elevate the system's average speed while considerably augmenting the cumulative energy efficiency across all vehicles. They achieve this by creating larger gaps than the IDM vehicles, as evidenced in the bottom subfigure. It's also noteworthy that the average gaps established by AVs reduce progressively along the platoons, culminating in waves becoming increasingly attenuated.

In Figure 8.9, a time-space diagram provides a juxtaposition between IDM-controlled AVs and those controlled by RL. The portion depicted in black signifies the trajectory leader's halt at the $t = 300$s mark, as previously seen in Figure 8.7. It becomes evident that RL-controlled AVs significantly dampen this effect, yielding more uniformity in trajectory patterns when compared to their IDM counterparts, as evidenced by the smoother color gradient in the RL-controlled time-space diagram.

Lastly, the space gaps for the foremost AV in the platoon are illustrated in Figure 8.10 for both IDM and RL-controlled AVs. IDM predominantly maintains a gap equivalent to the minimal failsafe gap permissible for RL control. On the contrary, RL-controlled AVs shows a tendency for opening larger gaps, providing them with the flexibility to accommodate slow-downs and thus absorb waves, underlining the sophisticated adaptability of RL in these scenarios.

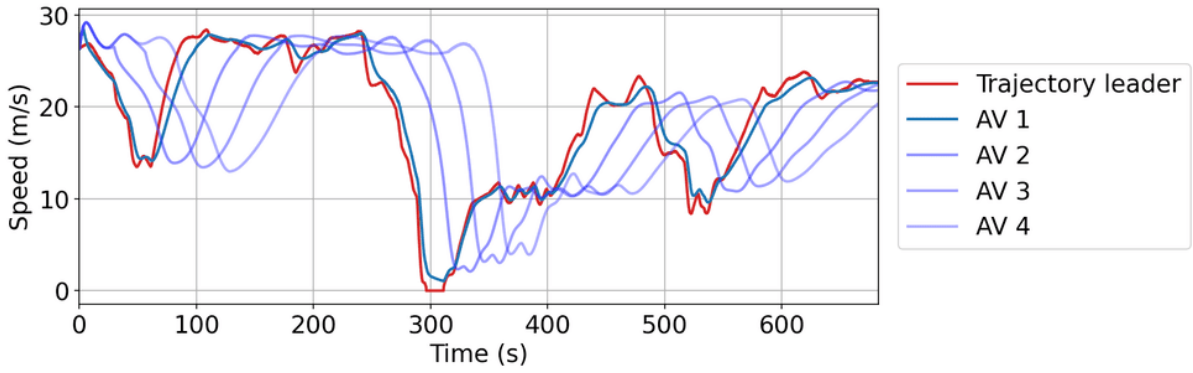


Figure 8.7: Trajectory leader vehicle replaying a trajectory from the dataset, followed by 10 platoons each composed of 1 AV and 19 IDM human vehicles, which corresponds to 200 vehicles (not including the trajectory leader) and a 5% AV penetration rate. The plot displays the speed of the trajectory leader as well as the first 4 AVs in the platoon. The remaining ones are omitted for visibility but follow a similar trend. © 2024 IEEE.

ACC-based RL Problem Formulation

The second of the two major classes of algorithms that were designed is the ACC-based RL controller. While both controllers ultimately control the driving behavior of the AV with the goal of minimizing fuel consumption, they do this in different ways. While the output of the acceleration-based RL controller is simply a scalar indicating what acceleration to actuate, the output of the ACC-based RL controller controls the inputs to the vehicle's native ACC system, and contains two values: (1) the speed setting and (2) the gap setting, both of which are described below in Section 8.3.

The final rendition of the controller that was ultimately deployed during the MVT is a combination of two controllers, with a transition being triggered at 60 mph. They are referred to as the low-speed controller and high-speed controller, with the low-speed controller operating when the vehicle is driving below 60 mph and the high-speed controller operating when the vehicle is driving above 60 mph.

In [79], we describe additional details regarding control interfaces with each vehicle. When interacting through ACC-based control, an approximate model was used to convert desired changes in speed into set point changes for the ACC, which was replicated for RL training purposes. Interaction through ACC-based control reduces the theoretical controllability of the system. However, it has the benefit that it broadens the implementation pathway to include more vehicle types, and depends on stock safety behaviors in order to protect vehicle occupants.

In Table 8.3, we enumerate the progression of the controller until its final stage, including all the intermediate observation spaces, reward functions, and results that prompted design

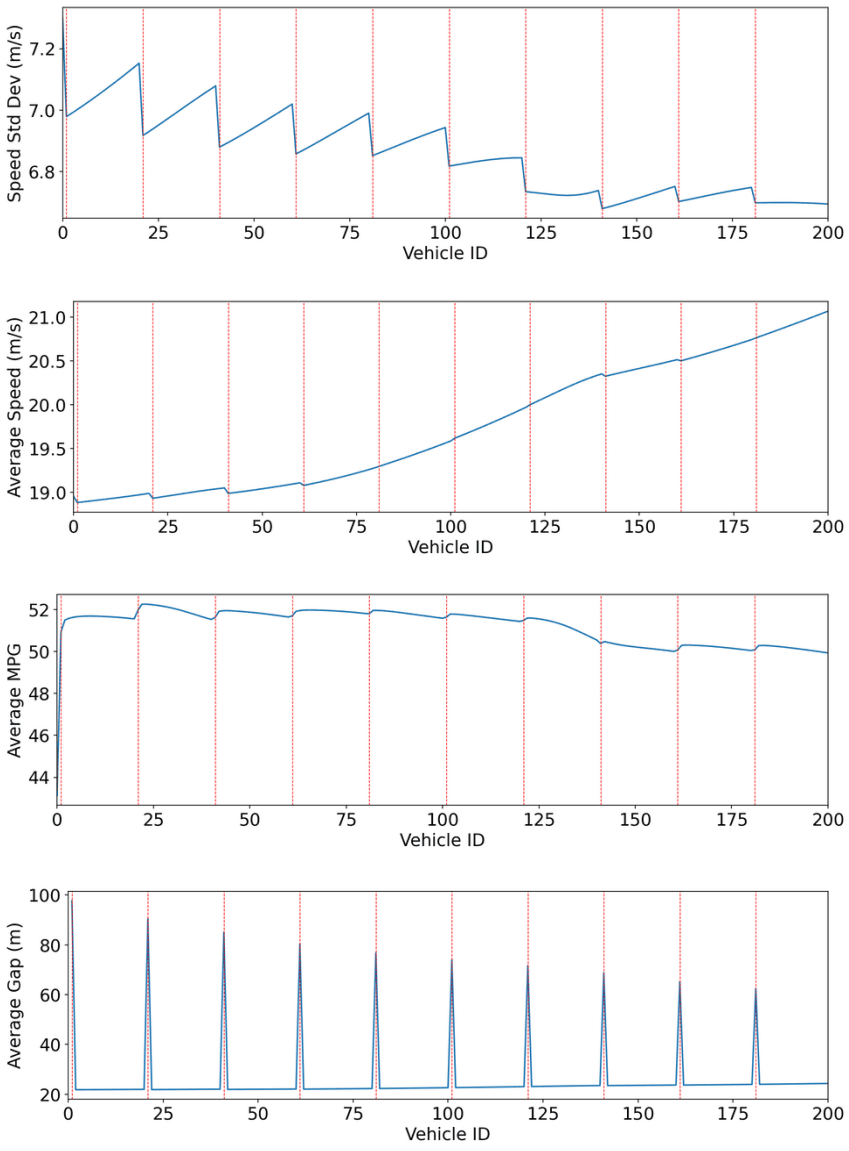


Figure 8.8: Trajectory leader vehicle replaying a trajectory from the dataset (displayed in Figure 8.7), followed by 10 platoons each composed of 1 AV and 19 IDM human vehicles, which corresponds to 200 vehicles (not including the trajectory leader) and a 5% AV penetration rate. We plot metrics aggregated over the whole simulation as a function of vehicle ID, 0 being the trajectory leader, 200 the last vehicle in the platoon, and $1 + 20k$ for $k \in \{0, 1, \ldots, 9\}$ are AVs (indicated by vertical red lines). From top to bottom, the aggregated metrics are speed variance, speed average, miles-per-gallon average and space gap average. Large spikes correspond to the start of a platoon. © 2024 IEEE.

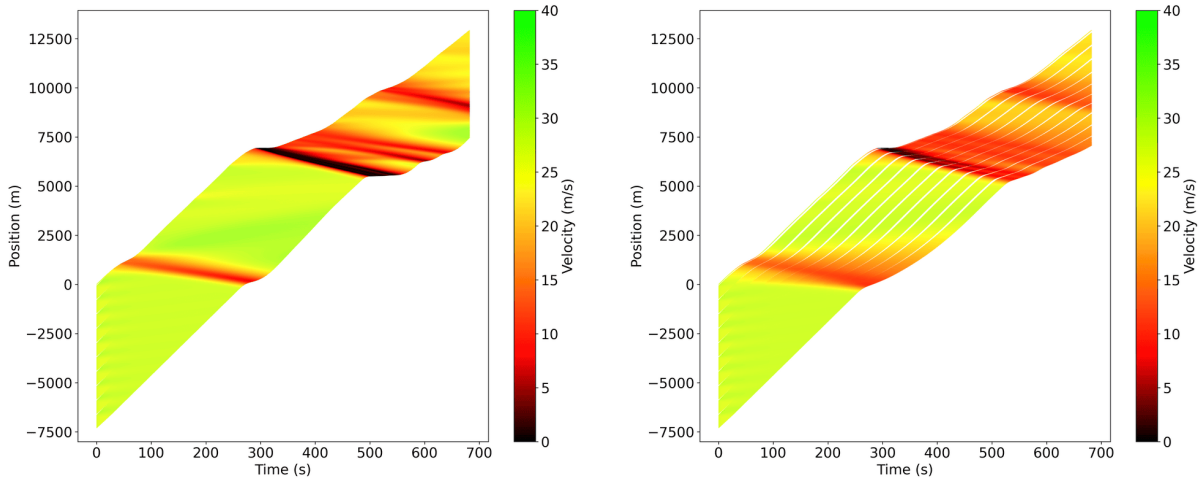


Figure 8.9: Trajectory leader vehicle replaying a trajectory from the dataset (displayed in Figure 8.7), followed by 10 platoons each composed of 1 AV and 19 IDM human vehicles, which corresponds to 200 vehicles (not including the trajectory leader) and a 5% AV penetration rate. We plot a time-space diagram when AVs are IDM-controlled (top) and when they are RL-controlled (bottom). Colors correspond to vehicle speeds. The black color region corresponds to the part where the trajectory leader comes to a stop around the $t = 300$s mark (see Figure 8.7), which the AVs manage to dampen (bottom). On this particular trajectory, the AVs improve the fuel efficiency of all of the 200 vehicles by 12.67%, as evidenced by the smoothed out colors. © 2024 IEEE.

changes, which led to the final controller. For the final problem formulation, refer to sections 8.3 and 8.3. The action space remains the same throughout all these variations. The switch between acceleration-based control and ACC-based control can be noted between versions v4 and v5. This also marks the progression to "undetectable" controllers, when lead vehicle distance was removed from the controller's observation space (although lead vehicle distance was still used by the car for its own execution of the ACC), in order to support as many different vehicle types as possible for the final field test. ACC-based control with no lead vehicle information provided the opportunity to field control on nearly 100 vehicles, instead of acceleration-based control on only a few vehicles.

Action Space

The final action space of the ACC-based controller we deployed consists of:

1. The speed setting, which dictates the maximum speed that the ACC can drive at;
2. The gap setting, which takes one, two, or three bars, with each bar indicating a higher allowable gap. Each bar corresponds roughly to constant time gaps of 1.2, 1.5, and 2.0

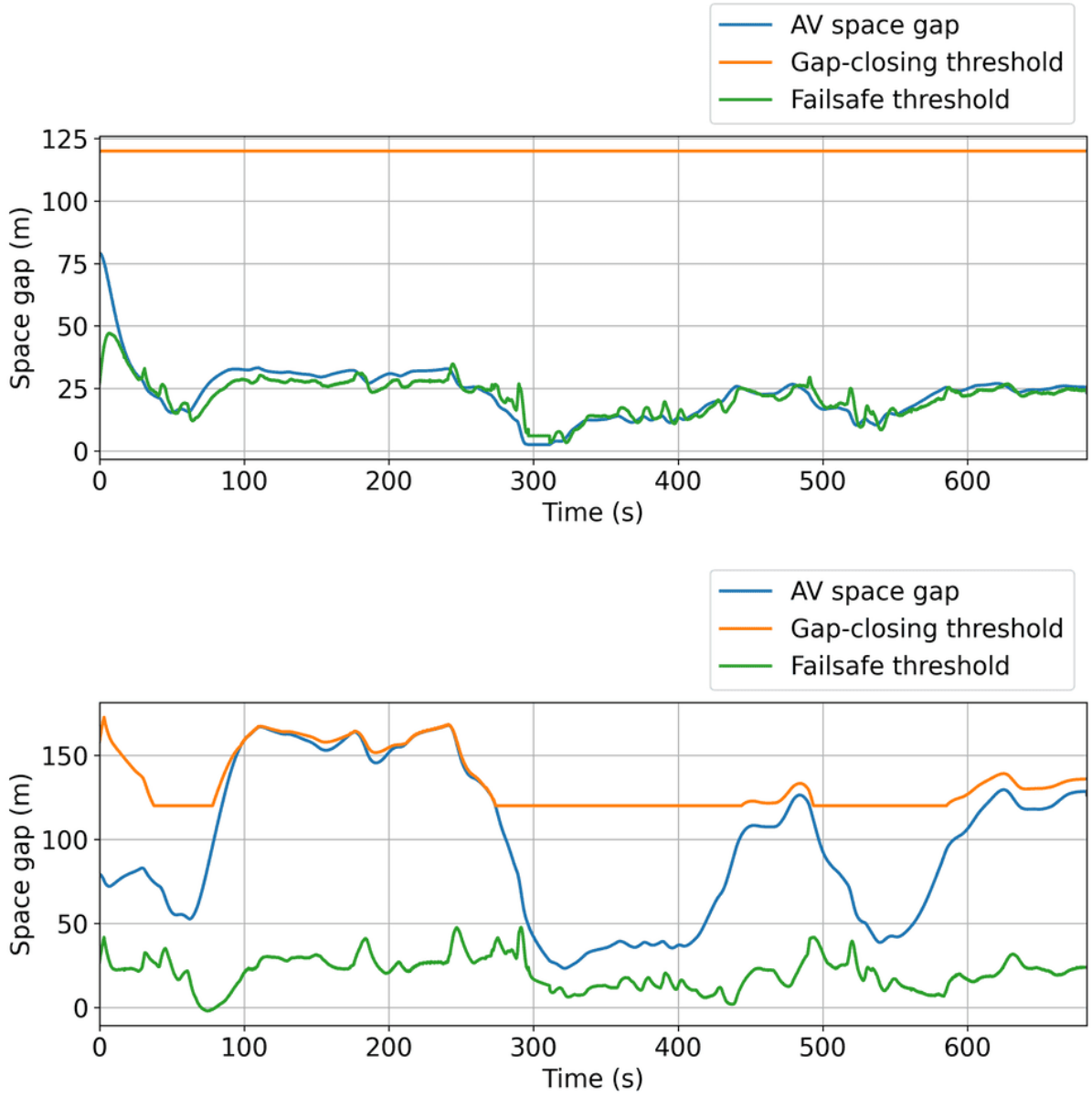


Figure 8.10: Trajectory leader vehicle replaying a trajectory from the dataset (displayed in Figure 8.7), followed by 10 platoons each composed of 1 AV and 19 IDM human vehicles, which corresponds to 200 vehicles (not including the trajectory leader) and a 5% AV penetration rate. We plot the space gap of the first AV in the platoon, in the case where it is IDM-controlled (top) and RL-controlled (bottom). The orange line represents the gap above which our gap-closing wrapper triggers, while the green line represents the gap below which our failsafe wrapper triggers. Note that the wrappers are not applied on IDM but are still shown for comparison.

Controllers v1-v5				
Version	**Action Space**	**Observation Space**	**Reward Function**	**Results**
v1	Acceleration	• v^{av}, AV speed • v^{l}, leader speed • h, space gap	$r_t = -c_0 E_t - c_1 a_t^{av2}$	Simulated cars crash in order to avoid penalties, opening large space gaps to optimize for the energy model
v2	Acceleration	• v^{av}, AV speed • v^{l}, leader speed • h, space gap • Failsafe threshold • Gap-closing threshold	$r_t = 1 - c_0 E_t - c_1 a_t^{av2} - c_2 P_t$, where $P_t = \mathbb{1}_{h_t^{av} \leq h_{min} \vee h_t^{av} \geq h_{max}}$	Continued simulation crashes and large space gaps; most controllers worsen MPG; for controllers with AV energy benefits, system energy consumption does not improve
v3	Acceleration	• v^{av}, AV speed • v^{l}, leader speed • h, space gap • Previous leader speeds • Failsafe threshold • Gap-closing threshold	$r_t = 1 - c_0 \sum_{i=1}^{n} E_t^i - c_1 a_t^{av2} - c_2 I_t$ In controller v3, instantaneous energy is taken as a function of an entire platoon of vehicles (the AV and the 24 human vehicles behind it) instead of just the AV. The horizon is also increased for controller v3.	Eliminated simulated crashes and stopping behavior; MPG gains significantly above baseline (System MPG: **+20%**, AV MPG: **+27%**); agent maintains maximum distance allowed by gap-closing; reward function aligned with MPG & comfort objectives, RL successfully optimizing reward function

Controllers v1-v5				
Version	**Action Space**	**Observation Space**	**Reward Function**	**Results**
v4	Acceleration	v^{av}, AV speed; v^l, leader speed; h, space gap; Previous leader speeds; Failsafe threshold; Gap-closing threshold; Target speed (speed planner)	$r_t = 1 - c_0 \sum_{i=1}^{n} E_t^i - c_1 a_t^{av2} - c_2 I_t - c_3 h_t$ A small gap penalty is introduced to discourage the controller from staying too close to the gap-closing threshold.	Reasonable space gaps, MPG gains substantially above baseline (System MPG: **+23%**, AV MPG: **+27%**)
v5 (Deployed)	• ACC speed setting • ACC gap setting	v^{av}, AV speed; Previous accelerations*; Previous velocities*; Previous requested velocities*; Target speed; Minicar flag; Max headway flag; Speed setting; Gap setting	$r_t = 1 - c_1 a_t^2 - c_2 (v_t^{av} - v_t^{sp})^2 - \frac{c_3}{n} \sum_{i=1}^{n} E_t^i - c_4 \mathbb{1}_{h_t^{av} \leq h_{min} \vee h_t^{av} \geq h_{max}}$,	Reasonable space gaps, MPG gains substantially above baseline (System MPG: **+23%**). *Since the deployed controller is a combination of two RL controllers, these states are slightly different. View 8.3 for these distinctions. For more details on performance, see Table 8.1.

Table 8.3: A description of the progression of controllers, from the first rendition to the final controller that was deployed at the MVT. High-level information on the action space, observation space, reward function, and results are given for each version of the controller.

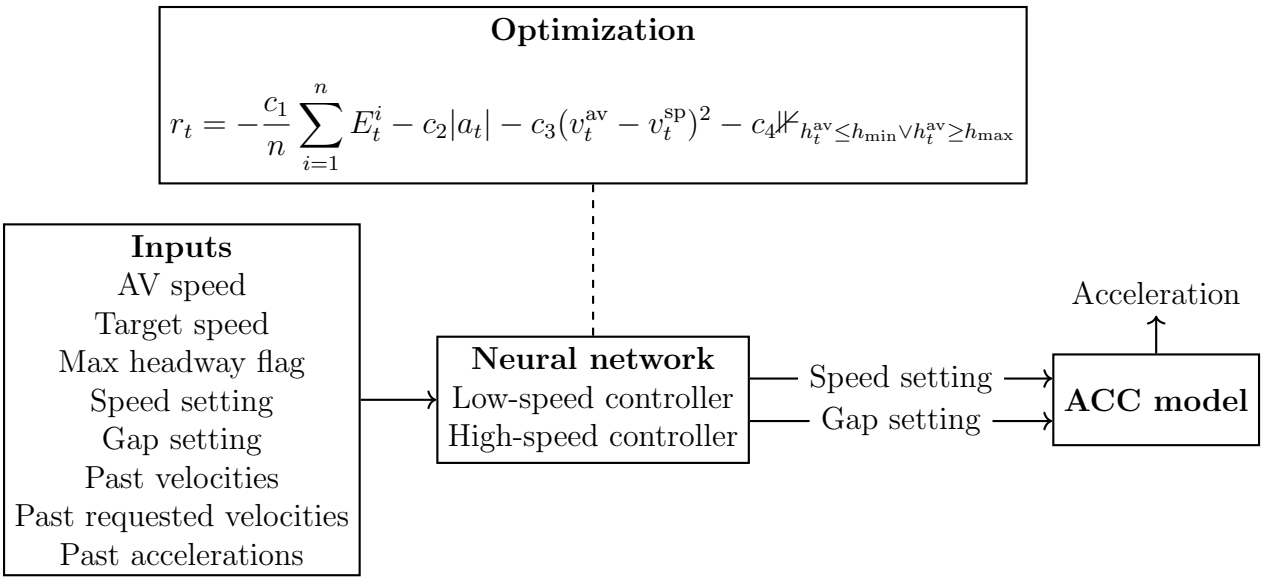


Figure 8.11: Diagram summarizing the design of the ACC-based controller. The observations described are the inputs into the controller. The controller outputs a speed setting and a gap setting, which are then input into the Rogue's native ACC model, which finally outputs an acceleration. The neural network is optimized via the reward function described in this section.

Figure 8.12: Dashboard view while the RL controller is being run. The speed setting is set at 45 MPH and the gap setting is set at three bars. © 2024 IEEE.

seconds.

Depicted in Figure 8.12, the speed setting appears as the green speed number at the top, and the gap setting appears as green horizontal bars between the two vehicles. In implementation, the controller actions are realized as requested settings, sent to software which sends commands through the in-vehicle network [29, 42, 101], and ultimately are realized by

physical systems like the engine and wheel torques.

Observation Space

The observation space of the ACC-based RL controller is as follows:

- v, velocity of the AV;
- v_s, target speed given by the Speed Planner;
- h_{max}, the maximum headeway recommended by the Speed Planner;
- l, a flag that indicates whether there is a leader vehicle within approximately 80 meters;
- s, the current ACC speed setting;
- g, the current ACC gap setting.

The observation spaces of the low-speed and high-speed controller are slightly different. In addition to the base state listed above, the high-speed controller contains the following states:

- $\overrightarrow{a_{prev}}$, the accelerations during the previous 6 time steps.

The low-speed controller contains the following states in addition to the base state:

- v_{s200}, target speed given by the Speed Planner 200m downstream;
- v_{s500}, target speed given by the Speed Planner 500m downstream;
- v_{s1000}, target speed given by the Speed Planner 1000m downstream;
- $\overrightarrow{a_{prev}}$, the accelerations during the previous 5 time steps;
- $\overrightarrow{v_{prev}}$, the velocities during the previous 10 time steps;
- $\overrightarrow{r_{prev}}$, the requested velocities during the previous 10 time steps.

Reward Function

The final reward function for the ACC-based controller was:

$$r_t = 1 - c_1 a_t^2 - c_2 (v_t^{av} - v_t^{sp})^2 - \frac{c_3}{n} \sum_{i=1}^{n} E_t^i - c_4 \mathbb{1}_{h_t^{av} \leq h_{min} \vee h_t^{av} \geq h_{max}},$$

where $c_1 - c_4$ are coefficients, a_t^2 is an acceleration penalty; $(v_t^{av} - v_t^{sp})^2$ is a squared penalty on the difference between the Speed Planner's suggested speed and the actual speed; $\frac{1}{n} \sum_{i=1}^{n} E_t^i$ is instantaneous fuel consumption; and the last indicator term is an intervention penalty that is invoked if the space gap is less than the minimum space gap (and thus invokes the failsafe) or greater than the max space gap (and thus invokes gap closing mode).

ACC-based RL Simulation Results

The deployed ACC-based RL controller interfaces with the Rogue's ACC system by toggling its speed and gap settings. Note that the version of the controller that is deployed during the MVT interfaces with several hardware layers, detailed in Section 8.4, so deployment results are discussed in a separate section. We now proceed to analysis of the trained controller.

A series of candidate controllers, including some non-RL controllers, were tested and evaluated, with their results on display in Table 8.1. Controllers are evaluated in three different traffic scenarios: a) Bottleneck, b) Shockwave, and c) Freeflow. They are also evaluated across two different speed planners, here labeled "Speed Planner A" and "Speed Planner B," the combination of which was ultimately deployed. Fuel economy (described in Section 8.2), throughput, and speed are measured in this evaluation. The other controller, titled "Microaccel," is also featured in this issue [60]. The performance metrics on display inform the choice to create the deployed RL controller, a mixture of the "low-speed controller" and "high-speed controller", also exhibited in this table. The low-speed controller achieves higher performance than the high-speed controller across all metrics in the bottleneck scenario. Compared to the IDM baseline and maximizing across the two speed planners, it achieves 8.22% gains in fuel economy, 14.73% gains in throughput, and 8.68% gains in speed for and in fuel economy and throughput in the shockwave scenario. In the shockwave scenario, the low-speed RL controller achieves, over the two speed planners, 22.96% gains in fuel economy and 2.75% gains in throughput. It decreases speed by 2.14%. In the freeflow scenario, the low-speed controller saw difficulty with maintaining traffic speeds, thus resulting in a -27.52% decrease in speed. The high-speed RL controller slightly improves the speed and throughput by 1.6% and 0.26%, respectively. Thus, the deployed controller (henceforth simply referred to as the RL controller) is composed of the low-speed controller below 60 mph and the high-speed controller above 60 mph.

In general, flow smoothing behavior is evident in the deployed controller. As shown in Figure 8.13, which compares the time-space diagrams of the IDM baseline with the RL controller of a congested scenario with real data taken from the I-24, a 9.1% increase in MPG can be observed. Notably, in evaluation with RL-controlled AVs, the deep red "stop" portions of the stop-and-go-waves are shorter, and there is a smaller deviation in speed. Large fluctuations in speed are contributors to poor fuel economy, so the stabilization of speed explains the performance improvement.

8.4 Migrating Control from Simulation to Vehicles

Validation Work

A chief safety issue concerned the migration of the controllers from software to hardware. While safety and feasibility of a controller can be evaluated in a simulation via unit tests and performance analysis, the measured efficacy of the controller may not be maintained when crossing into the real world. This phenomenon is commonly referred to in RL as

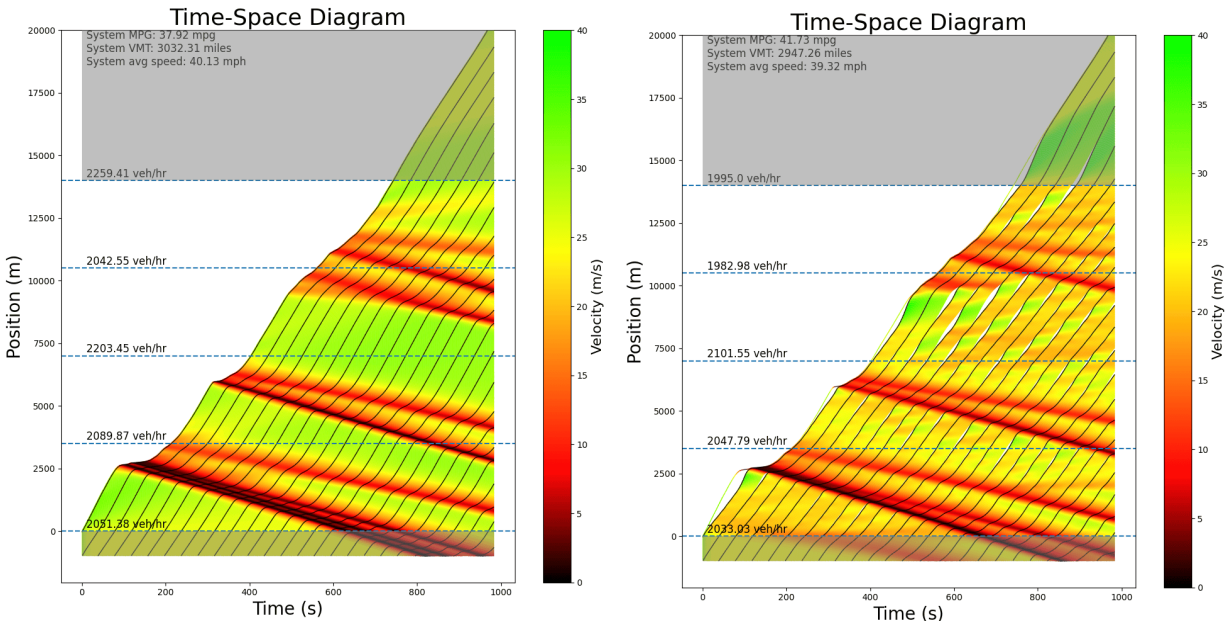


Figure 8.13: Time-space diagrams of a congested traffic scenario with real data from the I-24. The top depicts the IDM baseline with 0% penetration rate of AVs. The bottom depicts a 4% penetration rate of the RL controller. Between the baseline and the RL controller, a 9.1% increase in MPG is measured. The black lines depict AVs which can be seen opening gaps in the bottom diagram. © 2024 IEEE.

the "sim-to-real gap" or "reality gap," describing how the discrepancies between simulation and the real world can cause a policy to behave in unexpected ways [66, 110]. This reality gap challenge is increased by the nuanced complexity of underlying software and hardware platforms which enable the RL controller to be executed in a real vehicle.

The path from simulation to hardware is as follows:

1. The form the controllers takes when trained in simulation is that of a neural net, wrapped as a PyTorch model.

2. The PyTorch model is then exported to ONNX format. ONNX (the Open Neural Network Exchange) is a machine learning model standard that is compatible with a variety of platforms [107].

3. The ONNX model communicates with the Robot Operating System (ROS), the development of which is outside the scope of this paper.

4. ROS-based control commands are passed from the *software bridge* via can_to_ros [101] and libpanda [29], which decodes the Controller Area Network (CAN) bus sensor information.

Figure 8.14: The Nissan Rogues with their hoods popped open to access custom equipment necessary to enable RL control. © 2024 IEEE.

5. The *hardware bridge* consists of an embedded computer (Figure 8.18), a custom printed circuit board (PCB), and a custom CAN cable that exposes access to the in-vehicle network. The custom PCB, managed by the embedded software, transforms ROS-based commands into voltages to change ACC setting.

In the migration from software to hardware, there are four individual steps, thereby increasing the surface area for error. An array of validation tools were used to guarantee and test for safety prior to deployment. These include:

- Simulating hardware controllers in simulation
- Extensive test drives and live debugging

Software-in-the-loop tests consisted of using the ONNX model (that interfaces with ROS), and running a ROS simulation using one of the bag files from a test drive along the routes

Figure 8.15: A shot of two rows of Nissan Rogues, which the RL controllers were run on. The Rogues were stored in a parking lot during the duration of the MVT. © 2024 IEEE.

discussed in Section 8.4 and Figure 8.17. A bag file is a ROS-generated file format for storing data, saving a timestamped set of all ROS messages. For a test drive, this bag file would include metrics such as velocity, ACC state, GPS position, and sufficient information such that the driving trajectory can be replicated in simulation. ROS replays this trajectory, then uses the ONNX model to produce outputs (either accelerations or ACC settings, depending on the controller) to respond to the simulated trajectory. We compared the ONNX output to that of the PyTorch output to ensure no loss of information between this migration.

Following this, the next step of validation was a series of rigorous test drives, one of which is shown in Figure 8.16. Test drives occurred on any subset of either of the yellow or orange routes depicted in Figure 8.17. Through test driving, we discovered and patched numerous bugs in our control scheme that led to some of the following features.

Activation when Westbound

During both the actual MVT and the test drives, the controller is only activated on the westbound side of the highway. This is due to the fact that morning traffic, the focus of the MVT, only occurs for cars that are on the westbound side of the I-24 in the test corridor. A positional latitude-based westbound flag was initially used. This boolean function, returning

Figure 8.16: Performing a dry run in the Rogue on the I-24 with the RL-controller turned on, testing controller efficacy live with a connected laptop, before the MVT. © 2024 IEEE.

true or false, is given by:

$$w = x_{i+1} \geq x_i, \tag{8.6}$$

where i is time step, and x is the latitude portion of the GPS coordinate. The initial rendition of the westbound flag was a position-based function, checking if the latitude was increasing from the previous timestep and returning True if so. Due to the frequent standstill traffic where vehicles would come to a full stop, as well as slight noise in the GPS reading, this flag was prone to false-negatives while at a stop during westbound standstill traffic, with the latitude occasionally reading incorrectly as decreasing.

The final estimator fielded at the close of the MVT was a spline-based method. A spline of the entire westbound portion of I-24 was extracted from OpenStreetMap (OSM). Each vehicle's (X, Y) GPS coordinates were compared with the spline mapping, with the most recent 10 comparisons "voting" to determine travel direction.

Clipping Mechanism

Due to driver feedback, a clipping mechanism was applied to the speed setting portion of the action tuple to ensure smooth, comfortable control. The lower and upper clip bounds are based on the average speed of the vehicle during the last one second (10 time steps):

$$l = \frac{1}{10}\sum_{i=1}^{10} v_i - 15\text{mph} \ , \tag{8.7}$$

$$u = \frac{1}{10}\sum_{i=1}^{10} v_i + 5\text{mph} \ , \tag{8.8}$$

where l and u are the lower and upper bounds of the clip, respectively. The final clamp is executed as:

$$s = \min(\max(\min(\max(a, l), u), 20\text{mph}), 73\text{mph}) \tag{8.9}$$

where s is the final speed setting command, and a is the un-normalized output, or action, from the neural network. That is, the acceleration is capped between u and l, the two bounds, and then capped between 20 mph and 73 mph. The values 20 and 73 mph were chosen to keep controls in a safe region.

Acceleration Estimation

One notable quantity that was not directly read from the CAN bus was acceleration. Since accelerations are an input to our controller, we used the speed history to estimate the accelerations via a sliding window approach. First, noisy accelerations are calculated based on the current velocity and the previous velocity from the last time step (0.1 seconds). Then, the final acceleration a_t^{fixed} is estimated as as average of the last four noisy acceleration estimates.

$$a_t^{\text{noisy}} = \frac{v[t+1] - v[t]}{0.1} \ , \tag{8.10}$$

$$a_t^{\text{fixed}} = \frac{1}{4}\sum_{i=0}^{3} a_{t-i}^{\text{noisy}} \ , \tag{8.11}$$

where t indicates time step.

Experimental Design

The RL controller discussed in this article was deployed on 100 Nissan Rogue cars, which can be viewed in Figures 8.14 and 8.15. This section will describe the design decisions of test week, including information on routes, vehicle fleet, and day by day controller choices. For a high-level overview of the schedule of test week, please refer to Table 8.4. This will discuss exactly which version of the RL controller is being run each day ("Production"), which

MVT Schedule (continued below)			
Date	**Description of Events**	**Changes from Previous Day**	**Issues Encountered**
Monday 11/14	**Production:** 80+ Nissans running stock ACC. **Experimental:** 5 Nissans running RL controller with speed planner, westbound flag hardcoded to True.	n/a	(1) Latency in ACC Actuation. (2) Speed planner I/O, getting default values only. (3) Speed planner not publishing. (4) Westbound flag not robust. (5) Speed setting defaulting to 59 mph at full stop.
Tuesday 11/15	**Production:** None, due to inclement weather. **Experimental:** 8 Nissans running RL controller with speed planner, data fusion turned off.	Updated westbound flag calculation; dynamic lane assignment; new clamping logic; minor bugs.	n/a
Wednesday 11/16	**Production:** 80+ Nissans deployed with RL controller on westbound, stock ACC elsewhere. **Experimental:** 10 Nissans running RL controller on westbound and eastbound, stock ACC on side streets	Controls are allowed if server agrees and if not on a side street.	(1) Some locations identified I-24 East incorrectly as westbound. (2) Server for speed planner was offline $\approx$ 10 minutes.

versions are being ran to test efficacy for the following day ("Experimental"), as well as changes and issues encountered during each day with either the Production or Experimental controllers.

Route Plans

There were two selected routes that drivers operated on during the MVT. Both of these routes share an origin. They are:

MVT Schedule (cont...)			
Date	**Description of Events**	**Changes from Previous Day**	**Issues Encountered**
Thursday 11/17	**Production:** 80+ Nissans deployed with RL controller with server error (detailed right). **Experimental:** 10 Nissans running RL controller with fix to same bug.	Fixed westbound bug from Wednesday.	(1) Server for speed planner responded slowly, or not at all, due to exponential growth in computationally-demanding speed planner query.
Friday 11/18	**Production:** 100 Nissans deployed with RL controller. **Experimental:** None.	Speed planner query periodically executed, and served from cache. This speeds up queries and cached results by over 100x.	n/a

Table 8.4: A day-by-day description of the events of the MVT week, including which algorithms were run and changes from previous days. All the dates are in 2022. © 2024 IEEE.

- Orange Route: Starts at HQ, heads northwest, loops back southeast at Haywood Lane, heads southeast, loops back northwest at Waldron Road. Roughly 7.2 miles one way. Assigned to lanes 2 or 3.
- Yellow Route: Starts at HQ, heads southeast, and loops back northwest at Sam Ridley Parkway West. Roughly 6.6 miles one way. Assigned to lane 4.

Drivers are assigned with a 67 Orange–33 Yellow split. They are asked to drive the route and loop around as many times as they feel comfortable, or until they are asked to return to HQ when traffic had dissipated. Generally, driving would take place between 6:30 am and 9:30 am.

Driver Information

We recruited 170 drivers to drive for the MVT that week. 702 drivers responded to our announcement, 308 were cleared, and 170 attended our in-person training session, leading to 100 on the road at a given time. For further details on how logistics for the MVT operated, from driver recruitment to fleet management to route assignment, we direct you to the corresponding standalone article [9].

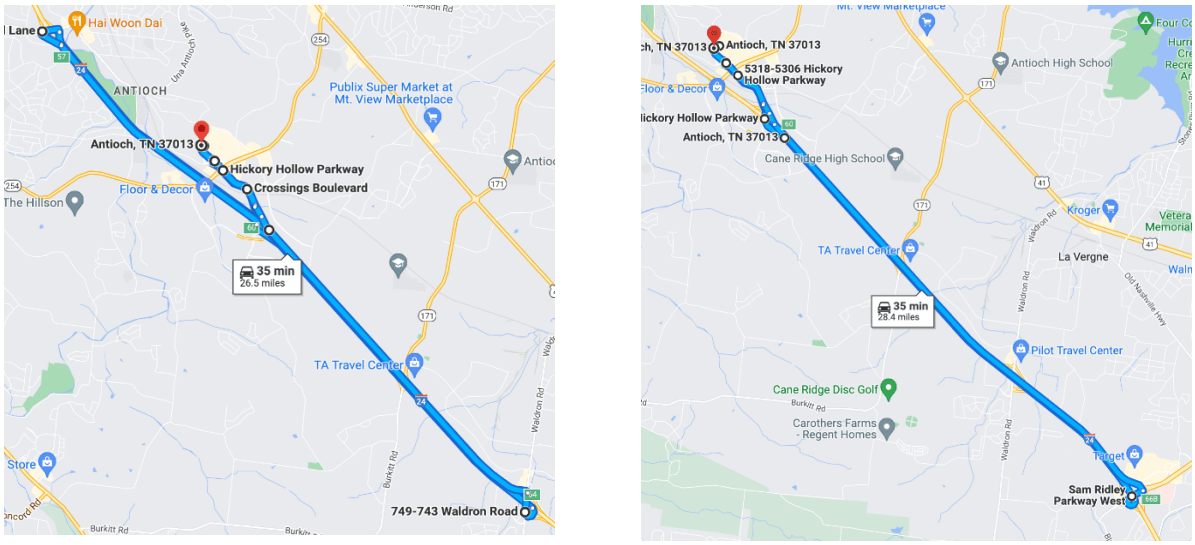


Figure 8.17: Depicted here are the two routes test drivers used during the MVT. Left: The orange route, roughly 7.2 miles one way. Right: The yellow route, roughly 6.6 miles one way, is further southeast on the I-24 than the orange route. For further details on routes and logistics, refer to [9]. © 2024 IEEE.

Result Analysis

In this section, we present findings from our analysis of the MVT data [79] collected from I-24. Data was captured using the I-24 MOTION testbed [50] where the 100 RL-controlled vehicles were deployed. The testbed uses a computer vision pipeline [49, 52, 53], a trajectory post-processing pipeline [158, 159], and a visualizaton package [175] to produce the trajectories of all vehicles on the designated highway portion during the experiment period. One challenge we face is the sheer volume of the data, which makes comprehensive analysis complex. In the subsequent discussion, we demonstrate our preliminary insights, suggesting that our AVs effectively mitigate traffic waves. However, it is important to emphasize that given the vastness of the dataset, further analysis may reveal additional perspectives or correlations and potentially lead to a reevaluation of our initial conclusions. Nonetheless, the trends that we observe appear consistent across the several metrics that we measure.

Data collected during the experiment

Data is collected using the I-24 MOTION testbed [50, 51], which consists of 276 high-resolution cameras mounted on poles along a 4.2 miles stretch of the I-24. Individual vehicle trajectories are extracted from the camera videos using computer vision algorithms (see [50] for a detailed overview of the system, algorithms, and datasets). This is a challenging problem due to the raw volume of data and difficulty of stitching images from different cameras

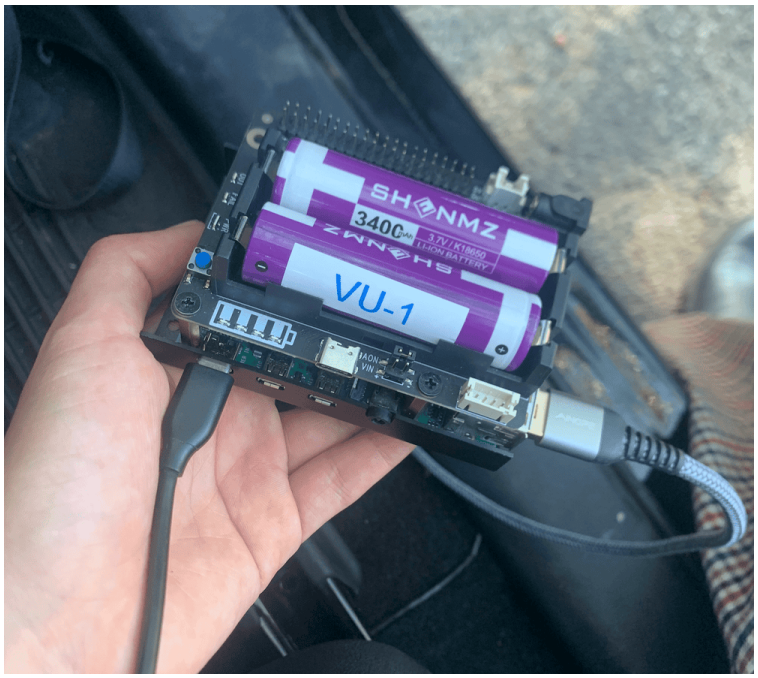


Figure 8.18: One of the many Raspberry Pis that contain the code for the RL controller. These are connected to the vehicle via a series of cables, which can be partially seen in Figure 8.16. © 2024 IEEE.

together as well as the presence of bridges, resulting in several segments of trajectories for a same vehicle. More information about how the data is collected, processed, formatted and accessible online can be found in [50, 52] and [79].

The data collected during the experiment spans from 11/14/2022 to 11/18/2022, from 6am to 10am each day. In total, the resulting dataset is over 200GB, with about 600,000 vehicle trajectories each morning, or 3 million over the whole week, where each trajectory consists of 440 data points on average (with a large variance). Trajectories are collected on all lanes of the highway (ranging from 4 to 5 lanes) in both eastbound and westbound directions. The whole database contains over 1.3 billion rows, about 1% of which (so about 13 million points) correspond to trajectories of our AVs. The average duration of a trajectory is 18.2s (with a standard deviation of 16.3s), and the average distance traveled per trajectory is 311.4m (with a standard deviation of 348.7m).

The vast volume of the data presents analytical challenges. Therefore, for many of our assessments, we choose a representative random sample to compute aggregate metrics. This sample is sized to ensure consistency across different subsets, while also being computationally manageable. Besides, as evidenced by the time-space diagrams in [79], our AVs experienced peak engagement in the period between 7am and 9am, registering the highest penetration rate of engaged AVs. Consequently, we focus our analysis during this time frame.

Finally, the data is pre-processed to include energy information: for each data point, and depending on the type of vehicle (extracted from the video data among one of 6 classes [50]), a corresponding calibrated energy model is leveraged to estimate the instantaneous fuel consumption of the vehicle. This allows us to compute energy metrics. See the background section from Chapter 9 or[69] for more information.

Dampening of traffic waves

It is a complex task to measure the energy impact of our AVs due to the absence of a counterfactual. Indeed, we can measure the energy consumption of vehicles on the highway during the experiment, but cannot know what would have happened in the absence of our controlled vehicles; comparing between different days does not appear to be a viable option, due to significant differences in day-to-day traffic patterns and dynamics. Instead, we consider a proxy by computing metrics as a function of the distance to the closest engaged AV in the traffic ahead (downstream). The idea is to show that vehicles that are close behind an AV experience smaller amplitude waves or less fuel consumption on average than vehicles far away behind any AV, which would suggest that our AVs have a smoothing effect.

We consider the data collected from Wednesday 11/16/2022 to Friday 11/18/2022, between 7am and 9am each day. The amount of data this corresponds to is shown in Figure 8.19. In this analysis, we bin the distances to the nearest engaged downstream AV by intervals of 50m, which allows us to get a smoother measurement and also account for errors of up to $\pm$10m in the computation of these distances.

We observe in Figure 8.20 that the average and standard deviation of the vehicles' speeds steadily increase as a function of distance to AV. The increase in speed is understandable: in order to smooth traffic, the AVs aim to open larger gaps used to absorb waves (as illustrated in Section 8.3), which they do by driving at lower speeds. More importantly, we observe a smaller speed variance behind AVs than far away from them, which could suggest that the AVs are reducing the amplitude of the waves.

Another metric pointing towards the AVs having a dampening effect is the fuel consumption analysis presented in Figure 8.21. Similarly to the speed analysis, we observe that the instantaneous fuel consumption increases both in mean and variance as a function of distance to AV, indicating than vehicles closer behind AVs consume less fuel on average.

Data in $v - a$ space

Figures 8.22 and 8.23 represent the data taken on two separate days, Tuesday and Friday, of the MVT. Tuesday is considered the baseline of the MVT, as no AVs were run on that day; however, it is important to note that inclement weather on Tuesday may have affected the data. Data from 7:00 am to 9:00 am CST are represented as a scatterplot in velocity-acceleration space. Two observations may be made. First, the two distinct clusters that are seen in Figure 8.22 represent the two bounds of the stop-and-go waves, with one at around $10m/s$ and another at around $30m/s$. In Figure 8.23, which represents RL-controller AVs,

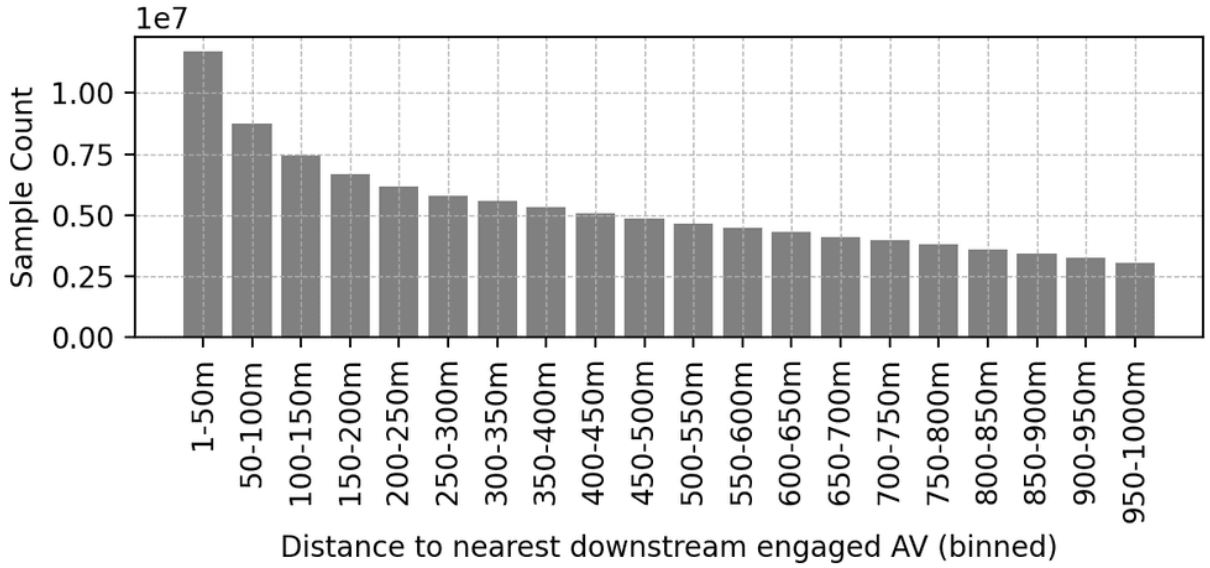


Figure 8.19: Number of data points as a function of distance to the nearest engaged AV in the downstream traffic, for all of the data collected between 7am and 9am on Wednesday 16, Thursday 17 and Friday 18, days on which our RL-controlled AVs were deployed. © 2024 IEEE.

the clusters are far less distinct. This is consistent with the results seen in Figure 8.13, where stop-and-waves are replaced with more consistent velocities. Second, it can also be observed that convex hull (ignoring 1% of outliers) of the RL-controlled day is considerably smaller in size.

Figure 8.24 shows two samples of the data collected on Friday 11/18/2022, one for vehicles that are closer behind AVs, the other for vehicles further way. We can notice two clusters: a high-speed one likely corresponding to free flow, and a lower-speed one that should contain both congestion and waves. We are mostly interested in what happens in the lower speed cluster, which we split at a speed of $23\frac{\text{m}}{\text{s}}$ corresponding to the vertical red line. We would like to understand how much "space" does our lower-speed cluster take. Computing the area of the convex hull would be very sensitive to outliers; instead, we fit the data with a Gaussian kernel, which gives us a variance matrix, and we compute the determinant of that matrix which gives us an estimate of the size of the Gaussian. Doing this procedure, which is shown in Figure 8.25, we obtain a determinant of 10.23 for the data up to 300m behind an AV (top plot), and of 14.40 for the data from 300m to 600m behind an AV (bottom plot). This means that closer behind an AV, there is on average a smaller range of speeds and accelerations, which is another indicator suggesting that the AVs have a smoothing effect.

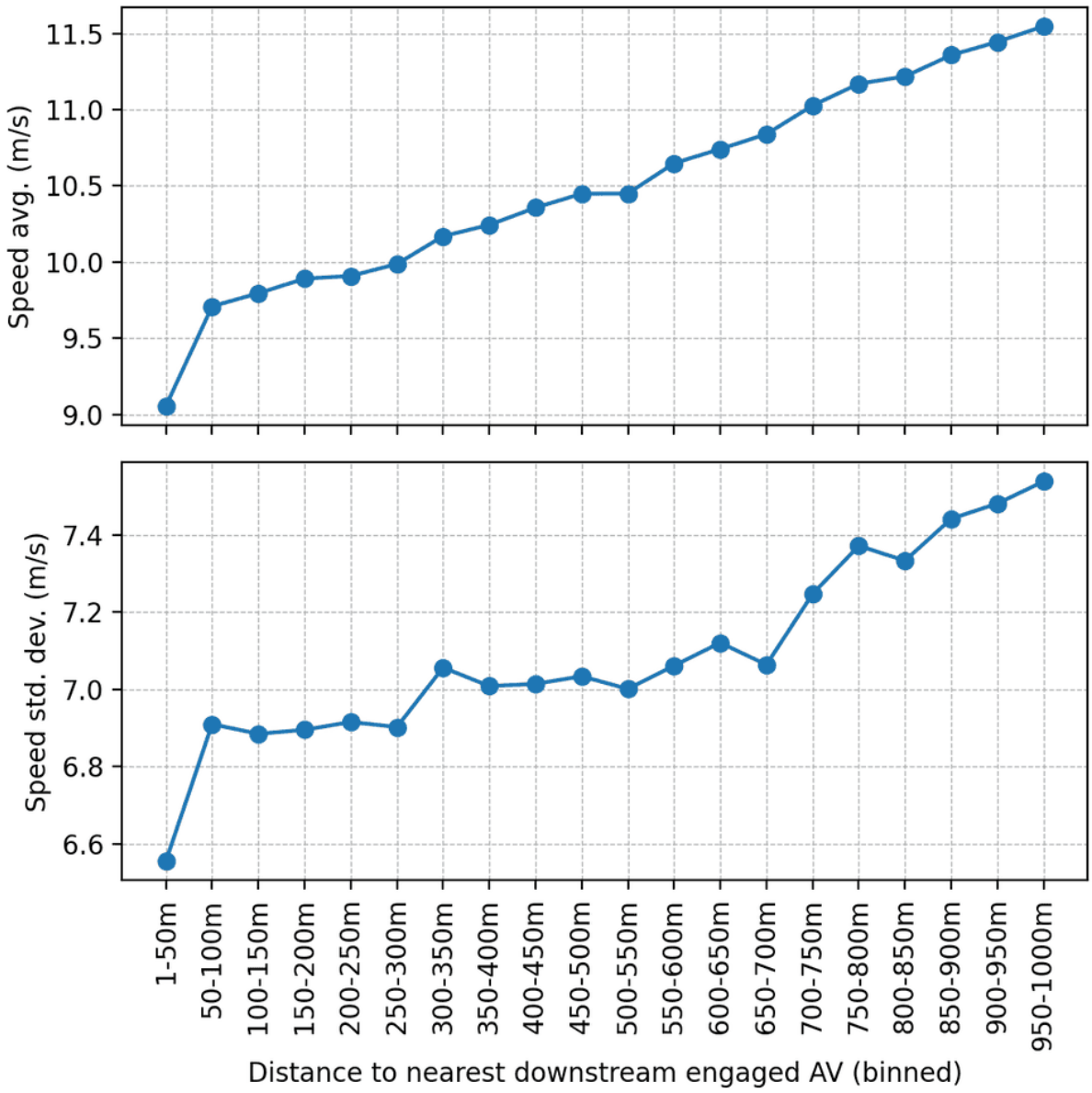


Figure 8.20: Speed average and standard deviation as a function of distance to the nearest engaged AV in the downstream traffic. The averages and standard deviations are computed over all of the data collected between 7am and 9am on Wednesday 16, Thursday 17 and Friday 18, days on which our RL-controlled AVs were deployed. This corresponds to over 2.5 million data points for each distance bin.

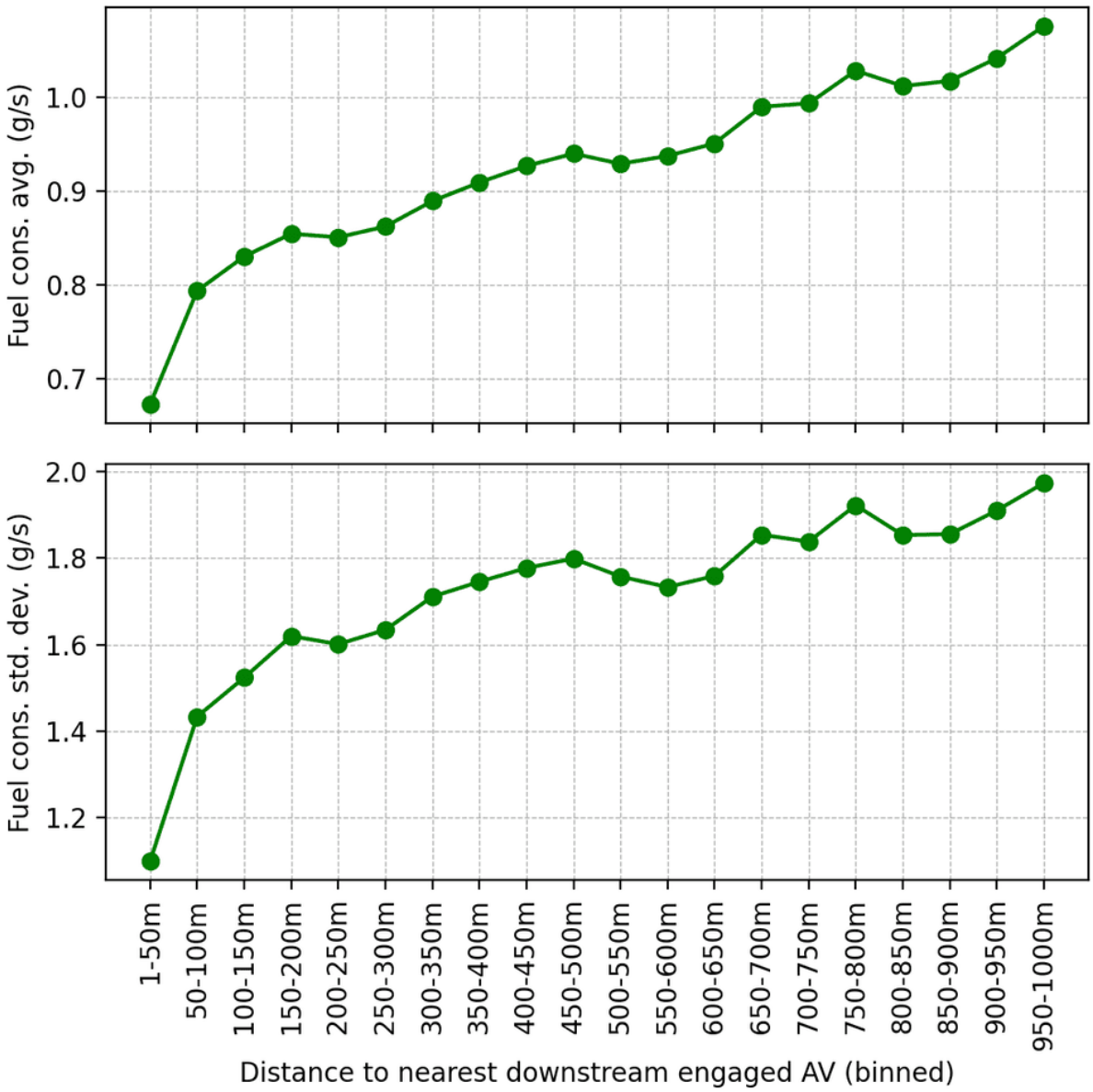


Figure 8.21: Instantaneous fuel consumption average and standard deviation as a function of distance to the nearest engaged AV in the downstream traffic. Instantaneous fuel consumption is calculated as a function of vehicle type, speed, instantaneous acceleration and road grade, using calibrated energy models available in [36]. The averages and standard deviations are computed over all of the data collected between 7am and 9am on Wednesday 16, Thursday 17 and Friday 18, days on which our RL-controlled AVs were deployed. This corresponds to over 2.5 million data points for each distance bin.

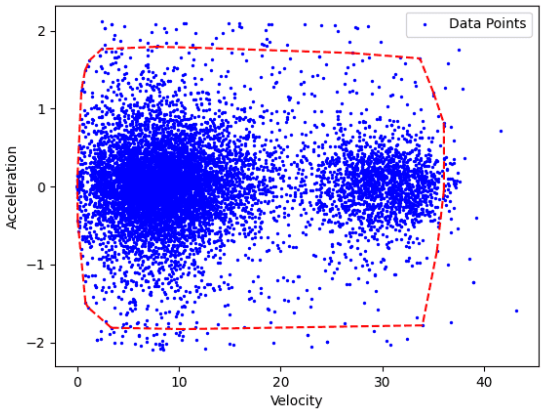


Figure 8.22: A scatterplot of velocity-acceleration pairs. We consider data on Tuesday, November 15th, 2022, from 7:00 am to 9:00 am CST, which we sample at a 1:1000 rate. Comparing to Figure 8.23, we note that the convex hull bounding 99% of the data points is larger in the absence of AVs, and that the data appear in two distinct clusters. © 2024 IEEE.

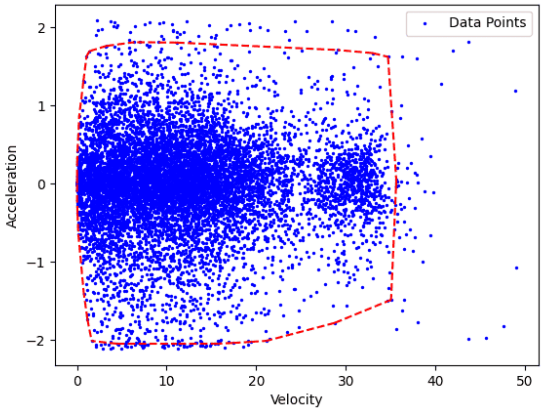


Figure 8.23: We consider data on Friday, November 18th, 2022, from 7:00 am to 9:00 am CST, which we sample at a 1:1000 rate. Comparing to Figure 8.22, we note that the convex hull bounding 99% of the data points is smaller with AVs, and that the data appears to be more spread out, indicating smoothing. © 2024 IEEE.

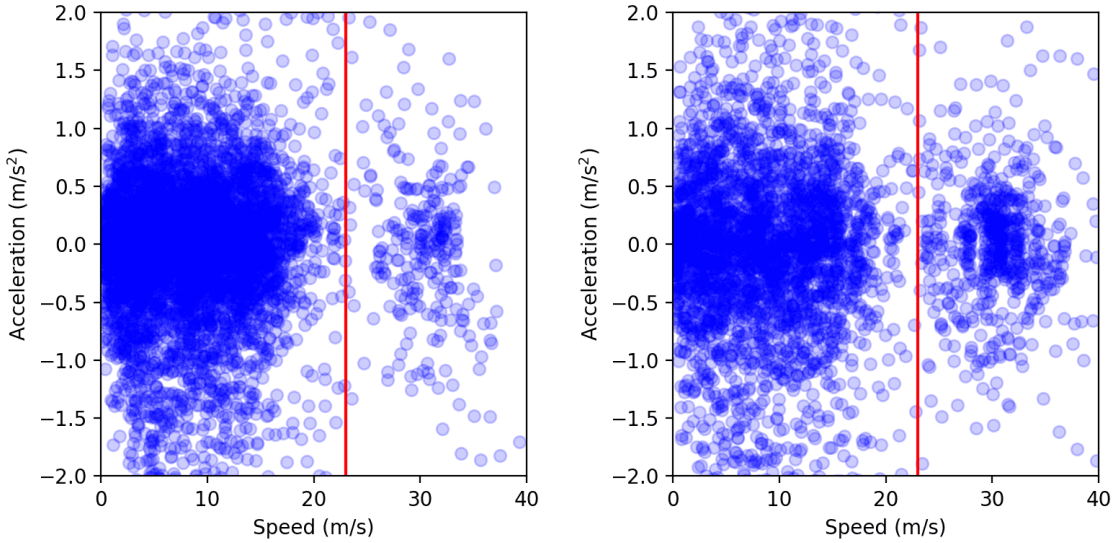


Figure 8.24: Data collected on Friday 18 between 7am and 8am, sampling 1 in 1000 data points, plotted in acceleration vs speed space. We plot points corresponding to vehicles that are up to 300m behind an AV (top), and from 300m to 600m behind an AV (bottom). © 2024 IEEE.

Histograms

The data are also represented in histograms with different speed bins. We compare three days: (a) Monday (Stock ACC running on AVs), shown in Figure 8.27 (b) Tuesday (No AVs), shown in Figure 8.26, and (c) Friday (RL-controlled AVs) 8.28. In general, we notice that the variance of the RL-controlled day is smaller than that of Monday or Tuesday, which indicates that traffic flow smoothing is being done. The difference is especially apparent between Tuesday and Friday, with frequencies of speed bins as high as 8%, indicating high magnitudes of stop-and-go waves.

8.5 Conclusion

This article has provided a comprehensive discussion of the technical intricacies surrounding the design and deployment of the RL controller at the MVT, the largest deployment of AVs to date designed to smooth traffic flow. By detailing the challenges and development of the controller design and training process, we have shed light on how RL may be utilized to influence autonomous systems in the future.

Across both the acceleration-based and ACC-based RL controllers that were presented in this article, we observe that the RL controller has a positive impact on flow-smoothing in a congested scenario. While the acceleration-based controller offers more flexibility in

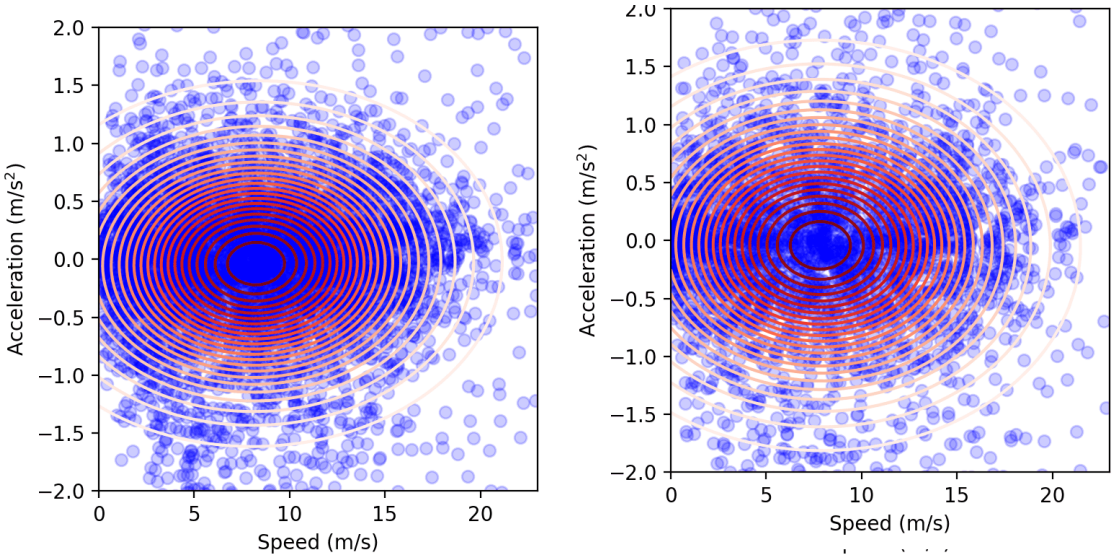


Figure 8.25: We would like to analyze how much "space" does our lower-speed cluster take. A naive approach would be to compute the area of the convex hull, but that is very sensitive to outliers. Instead, we fit the data with a Gaussian kernel, which gives us a variance matrix, and we compute the determinant of that matrix which is a way to measure the size of the Gaussian. Doing this procedure, we obtain a determinant of 10.23 for the data up to 300m behind an AV (top plot), and of 14.40 for the data from 300m to 600m behind an AV (bottom plot). This means that closer behind an AV, there is on average a smaller range in terms of speeds and/or accelerations, suggesting reduced waves compared to further away from the AV. © 2024 IEEE.

actuation, the ACC-based controller provides a smoother driving experience. We were able to produce 9% energy gains in simulation via the ACC-based controller. Through a variety of methods, we are able to share how the RL controller stabilizes speeds between stop-and-go waves by yielding smaller deviations in accelerations, which is an essential component of achieving energy-efficient solutions.

For a more detailed look at the other work that went into making the MVT happen, please look at [79] for a detailed look at how all the various components of the MVT worked together to produce this result; [157] for an explanation of the speed planner that influenced the RL controller; [9] for a deep-dive into another controller that was used for a different platoon of vehicles (Toyotas) during the MVT; and [69] for a look at the energy modeling that informed the RL algorithm.

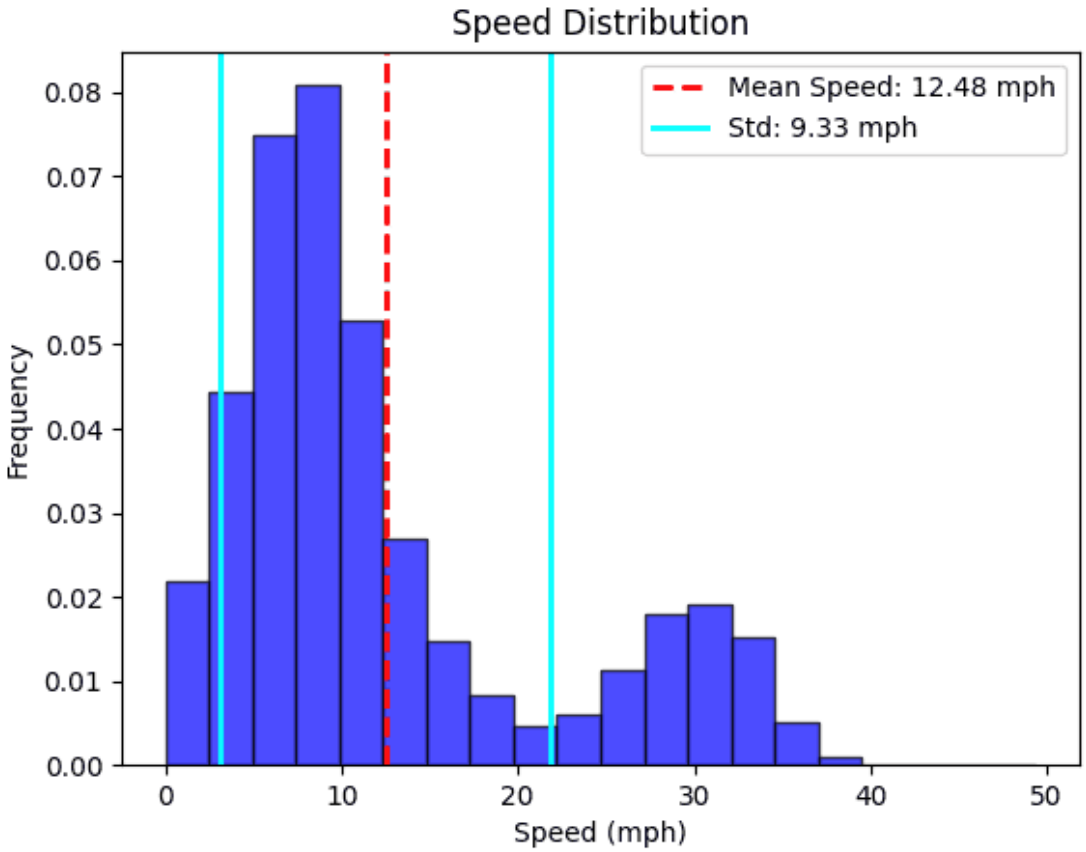


Figure 8.26: The speed distribution of vehicles on day 2 of the MVT. We consider data on Monday, November 15th, 2022, from 7:00 am to 9:00 am CST, which we sample at a 1:1000 rate. Comparing with Figure 8.27, we notice that the standard deviation tends to be higher and the speed bins are less spread out, with the highest frequency at 8%.

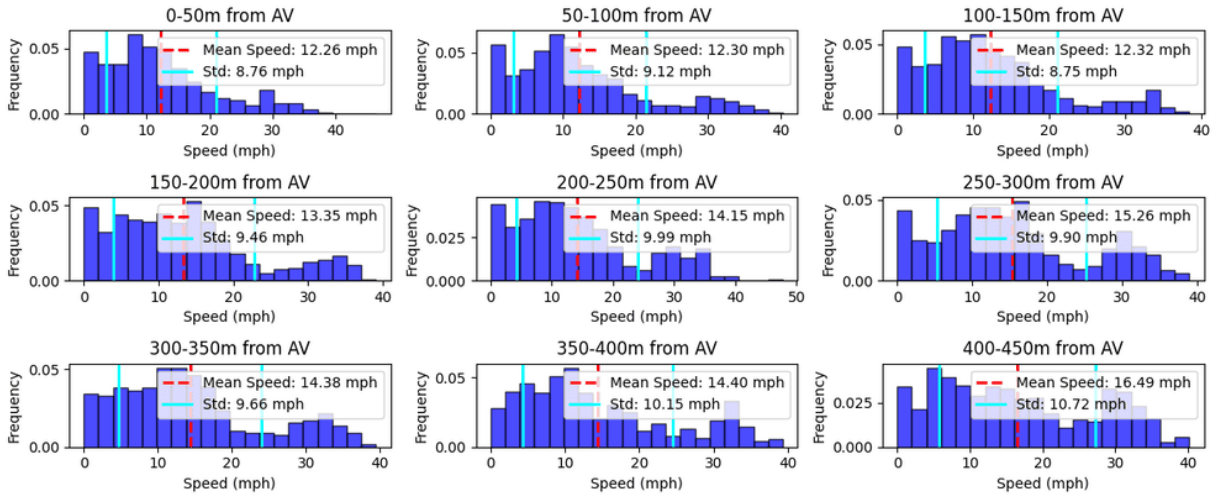


Figure 8.27: The speed distribution of vehicles on day 1 of the MVT, when stock ACC was run on the AVs. We consider data on Monday, November 14th, 2022, from 7:00 am to 9:00 am CST, which we sample at a 1:1000 rate. Comparing with Figure 8.27, we notice that the standard deviation tends to be higher and the speed bins are less spread out. © 2024 IEEE.

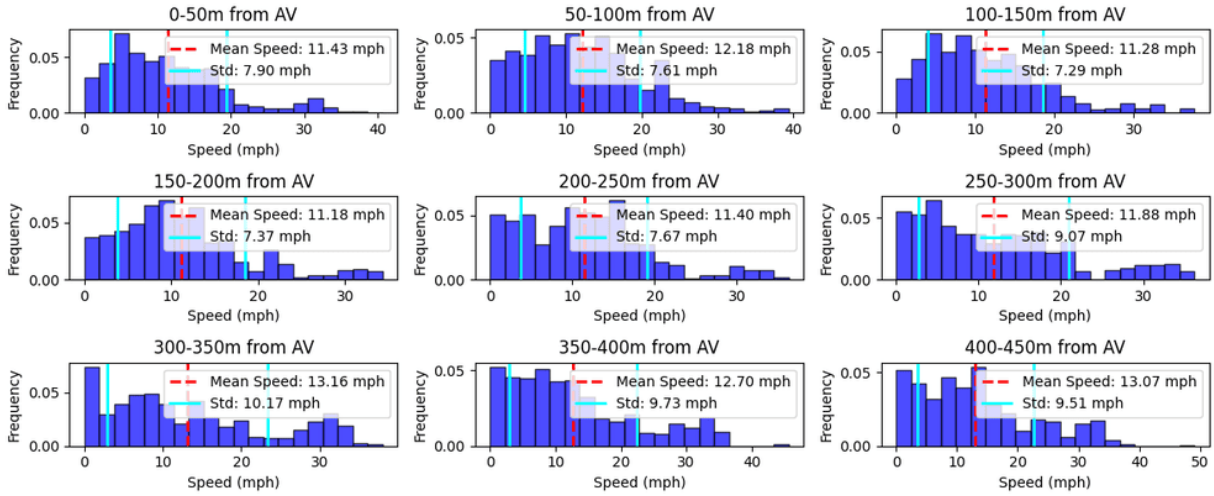


Figure 8.28: The speed distribution of vehicles on day 5 of the MVT. We consider data on Friday, November 18th, 2022, from 7:00 am to 9:00 am CST, which we sample at a 1:1000 rate. Comparing with Figure 8.26, we notice that the standard deviation tends to be higher and the speed bins are more spread out, indicating that the AVs are smoothing the clusters that form at low and high speeds. © 2024 IEEE.

8.6 ACKNOWLEDGMENT

This work was supported in part by the C3.ai Digital Transformation Institute under Grant Number 053483. This material is based upon work supported by the National Science Foundation under Grants CNS-1837244 (K. Jang), CNS-1837244/1837481/1837652 (A. Bayen/B. Piccoli/D. Work), CNS-2135579 (D. Work, A. Bayen, J. Lee, J. Sprinkle). Nathan Lichtlé is supported in part by the International Emerging Actions project SHYSTRA (CNRS). This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Vehicle Technologies Office 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.

Chapter 9

Traffic Control via Connected and Automated Vehicles: An Open-Road Field Experiment with 100 CAVs

9.1 Introduction

For an long time, the idea of harnessing societal benefits from automated vehicle (AV) technology has intrigued researchers, lawmakers, and mainstream culture. Its influence is perceptible in the initial phases of modern research and development in the United States dedicated to AVs. The progression of AV technology, extending over a span of more than three decades, is sometimes categorized into three successive generations:

Generation One. The 1990s witnessed the establishment of the National Automated Highway System Consortium (NAHSC) by the Federal Highway Administration (FHWA), dedicated to unveiling the potential societal advantages of an automated vehicle and highway system [131]. Collaborative entities within the consortium comprised institutions such as the University of California, Berkeley, California Partners for Advanced Transit and Highways (PATH), and General Motors (GM), among others. The enactment of Demo '97 on Interstate 15 in San Diego [138], California, marked the introduction and realization of groundbreaking concepts, including Adaptive Cruise Control (ACC), Vehicle-to-Vehicle (V2V) Communication, and Cooperative ACC (CACC) [126, 137, 63]. These demonstrations spotlighted high-speed platooning facilitated by V2V and CACC, coupled with the integration of infrastructure sensors, laying the groundwork for contemporary automated highway system (AHS) architectures. Emphasis was placed on the significance of both vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications [48]. Challenges during this period included scalability issues linked to infrastructure, the establishment of reliable communication systems, and the assurance of accurate vehicle positioning. The eventual decline occurred in the late 1990s due to constraints in research budgets at USDOT.

Generation Two. In the 2000s, the DARPA Grand Challenge was introduced by the

Defense Advanced Research Projects Agency (DARPA). The inaugural Grand Challenge in 2004 presented a $1 million prize to any team whose autonomous vehicle could traverse a 150-mile course in the Mojave Desert; unfortunately, no teams achieved success. The subsequent Grand Challenge in 2005 doubled the prize money, resulting in success for five teams who completed the course [27]. The third Grand Challenge in 2007 shifted its focus to an urban setting. The DARPA Grand Challenge events not only sparked a resurgence of interest in autonomous vehicles but also provided direct backing for the commercialization of successful technologies [139] over the ensuing decade.

Generation Three. In the 2010s, prominent automotive manufacturers (such as GM, BMW, Audi, and Tesla) and technology companies (such as Google/Waymo, Uber, Lyft) commenced the development of autonomous vehicle technology with the goal of commercialization. Substantial investments in this endeavor have spawned an industrial ecosystem centered on hardware, software, and services, all aimed at propelling the advancement of fully autonomous driving. Currently, various companies are conducting AV tests on public roads in the US. Notably, Waymo operates an autonomous ride-share service in Chandler, AZ, without a safety driver, and Cruise managed autonomous shuttles in San Francisco. Concurrently, Level 1 and Level 2 automated vehicle technologies, including ACC, are readily accessible in the commercial market. These commercial systems primarily prioritize safety and driver comfort [171], diverging from a primary emphasis on enhancing transportation system efficiency [58].

Our efforts culminate in a work tracing back to the early 1990s, wherein the fruition of the Generation One vision for automating highways in the interest of societal welfare began to manifest in the latter part of the 2010s, both within our research team and across various international research cohorts. In 2016, a consortium led by co-authors Delle Monache, Piccoli, Seibold, Sprinkle, and Work replicated the Sugiyama experiment [173] on string instability within a ring road, notably termed the "Arizona ring experiment." This initiative substantiated the mitigation of phantom jams through the incorporation of partially automated vehicle (AV) technologies and custom algorithms [112]. Shortly thereafter, co-author Bayen's laboratory at UC Berkeley pioneered the development of `Flow`, a software framework facilitating the integration of traffic microsimulation software with deep reinforcement learning (deep-RL) libraries [31, 165, 150, 66, 37, 176]. The UC Berkeley team was able to independently train an AV controller using `Flow`, successfully replicating the outcomes of the Arizona ring experiment [30], and extrapolated these results to diverse contexts, including freeway and urban traffic [151]. Of particular note is the distinction in terminology between "autonomous vehicle" and "automated vehicle," a deliberate specification emphasizing our historical and present focus on automated longitudinal control as Lagrangian traffic controllers.

In 2019, the Congestion Impacts Reduction via CAV-in-the-loop Lagrangian Energy Smoothing (CIRCLES) Consortium (see `https://circles-consortium.github.io/`) [3, 4, 64] was formed by co-authors Bayen, Piccoli, Seibold, Sprinkle, and Work; followed by co-authors Lee and Delle Monache. The objective of CIRCLES is to extend the aforementioned goals into real traffic experiments. The CIRCLES Consortium designed and developed a

modular, hierarchical control framework, consisting of a centralized Speed Planner and decentralized Vehicle Controllers, and implemented it on 100 vehicles in a large-scale field operational test, dubbed the MegaVanderTest (MVT).

This article presents our control system design and subsequent analysis of field test results. Diverse candidates for each module of the framework are developed utilizing cross-disciplinary knowledge and tools, including ordinary differential equation/partial differential equation (ODE/PDE)-based flow control [157], deep-RL [65], stabilization theory [13], functional analysis [174], optimal control on microscopic and macroscopic systems [25, 32], approximation theory [142], mean-field limits [54], non-entropic solutions to hyperbolic systems [15], model predictive control (MPC) via linearly constrained quadratic programming (LCQP) [169], kernel smoothing [46], traffic flow theory [47], variable speed limit [90], and many other areas. The control system was then evaluated and tested in the open road as part of the MVT, the largest deployment of AVs designed to smooth traffic flow. In this test, we deployed 100 AVs on Interstate 24 (I-24) near Nashville, TN in November 2022. The experiment coincided with the debut of I-24 MOTION (`https://i24motion.org`) [50, 51], a four-mile section of I-24 near Nashville, TN to capture ultra-high resolution trajectory data of all vehicles. Disclaimer: The work in this chapter is derived from publications contributed to by many authors; this author's personal contribution is focused on the controller development, selection, as well as migration onto vehicles. Other work is included for comprehensiveness and readability for readers.

9.2 Background

Vehicle Energy Models

For more detail on the following subsection, refer to [69]. The quantification of the energy demand of the vehicles on the road, given their trajectories, requires vehicle-specific energy models that take as an input the velocity profile $v(t)$ and the road grade profile $\theta(t)$ and output the resulting energy/fuel consumption rate $f(t)$. This project requires the quantification of the energy demand of traffic flow at large, composed of many vehicles; and also the use of reinforcement learning and optimization techniques that minimize (under certain constraints) the energy demand of traffic. Thus, the energy models used should accurately represent different vehicle types on the road and should average out any local non-convexity behavior due to gear switching, to avoid trapping the optimizer in local minima. For that purpose, we use energy models derived from a systematic model-reduction procedure to generate simple fitted models. The procedure starts from the fidelity software Autonomie [76], for a number of vehicles, each of which represents a typical average vehicle of a given class.

Vehicle Portfolio

To capture the diversity and prevalence of different vehicle types on US roads we select a representative group of vehicle classes on which we apply the model-reduction process to

derive their corresponding simplified energy models. Those vehicle classes are divided into two categories: (1) light-duty vehicles: compact size sedan, midsize sedan, midsize SUV, and Pickup, and (2) heavy-duty vehicles: Class4PND (Pickup and Delivery) and Class8Tractor. Each vehicle model represents a class of vehicles that have comparable weight (with load assumed half full) and fuel consumption characteristics [76].

Autonomie and Virtual Chassis Dynamometer (VCD)

We use the simulation software Autonomie Rev 16SP7 [76] with a library of energy models for several types of vehicles, including a detailed plant and controller model for its components in MATLAB and Simulink, where the blocks and files can be customized. To build our models, we use (i) physics-based vehicle parameters extracted directly from Autonomie, (ii) tuned parameters extracted in an automated fashion by running the Autonomie model on test cycles, and (iii) performance maps computed gear-by-gear on a complete velocity–load phase space of driving by running Autonomie's customized vehicle models on a VCD (those maps are vehicle speed to engine speed, vehicle speed and wheel force to engine torque, and engine speed and torque to fuel rate).

Semi-Principled Energy Models

We build an energy model that is semi-principled in that it has a physics-based part using Autonomie's extracted physics-based vehicle parameters, but it also relies on the maps obtained from the VCD. Gear scheduling in this model is based on choosing the feasible gear that yields the minimal fuel consumption, and the torque converter bypass clutch is assumed open in the first gear only. In contrast to Autonomie that considers hysteresis effects, this model yields the fuel consumption rate (and other outputs) as a direct function of instantaneous velocity v, acceleration a, and road grade θ.

Simplified Energy Models

A further model-reduction step is conducted by fitting the semi-principled models into simplified models. Those models have a simple polynomial structure that can easily be integrated into optimization and control problems, yet they are highly accurate. The fuel consumption rate function is

$$f(v,a,\theta) = \max\left\{\beta,\ C(v) + P(v)a + Q(v)a_+^2 + Z(v)\theta\right\}\ , \tag{9.1}$$

where β is the minimum fuel rate set to be zero or a positive constant depending on the fuel cut criteria, $a_+ = \max(-\frac{P(v)}{2Q(v)}, a)$, $C(v) = c_0 + c_1 v + c_2 v^2 + c_3 v^3$, $P(v) = p_0 + p_1 v + p_2 v^2$, $Q(v) = q_0 + q_1 v$, and $Z(v) = z_0 + z_1 v + z_2 v^2$.

In the above functions, c_0 ensures that fuel is being consumed at idle, the c_1, c_2, and c_3 terms can be interpreted as fuel consumed due to friction and air-drag, the $P(v)$ term yields fuel demand due to non-zero accelerations, the $Q(v)$ term captures the super-linear trend of

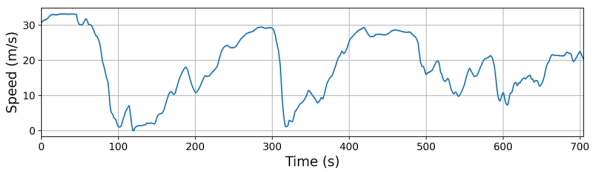


Figure 9.1: Speed vs. time for one of the dataset trajectories, exhibiting large acceleration and breaking patterns that can typically lead to stop-and-go waves. © 2024 IEEE.

fuel rate with respect to a, and the $Z(v)$ term captures the fuel consumed due to road grade, with the z_1 term playing the role of the weight force exerted at θ. Both types of models are validated, for all different vehicle types, against Autonomie models as the ground truth on standard EPA drive cycles [43] for flat roads and constant road grades drive cycles, and the results showed that the models are highly accurate (within 4% for zero road grades).

Trajectory Simulator

For more details on this work, please refer to [87]. Establishing a robust, representative, and reliable simulation environment is key for the successful implementation of our wave-smoothing controllers in autonomous vehicles, as it ensures their efficient operation across a wide range of traffic conditions. Our simulator is based on real human driver data collected on the highway [105], and has been extensively used for designing, training and evaluating the different types of controls implemented in this work, assessing their safety, energy-reducing performances, robustness and smoothness. This sidebar describes the procedure by which highway trajectory data was gathered and a simulator created from it.

Acquisition of Data

The trajectory dataset [105] for our study were recorded on a 14.5-kilometer segment of I-24, located southeast of Nashville, Tennessee. An example recorded speed trajectory profile can be seen in Figure 9.1. An instrumented vehicle is used to gather data which logs vehicle controller area network (CAN) data through `libpanda` [29] and GPS information from an in-built receiver. The CAN data collection comprises measurements like the speed of the vehicle under consideration (ego vehicle), the relative speed of the lead vehicle (the vehicle in front), the instantaneous acceleration, and the space gap (distance from bumper to bumper).

Refining Raw Data

Each drive's raw data is stored in two separate files: a CAN data file and a GPS file. The relevant data are extracted from the CAN file and adjusted to match the GPS time, which is recorded at 10 Hz. The high-frequency CAN data are downsampled and interpolated linearly to align with GPS time, whereas low-frequency CAN data are subjected to linear

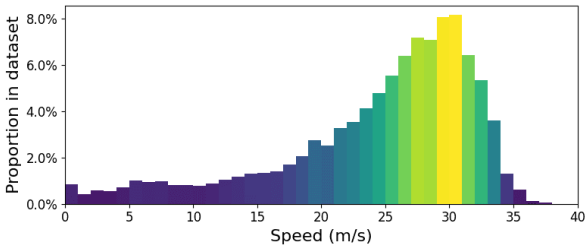


Figure 9.2: Distribution of speeds in the I-24 dataset. © 2024 IEEE.

interpolation to match the 10 Hz GPS time. GPS position data is used to calculate the distance traveled and direction. As westbound data usually demonstrate more consistent congestion, they are primarily used for training, comprising 60 trajectories, which translates to 8.8 hours and 772.3 kilometers of driving.

Analyzing the Dataset

The trajectory dataset encapsulates a wide variety of traffic conditions ranging from nearly stationary congested traffic to maximum speed free-flow traffic, including diverse acceleration and deceleration patterns associated with stop-and-go traffic. The example trajectory displayed in Figure 9.1 demonstrates the ego vehicle's quick transitions between low and high speeds.

While the primary interest lies in mitigating high-frequency waves that are common in congestion, Figure 9.2 indicates the tendency of speeds in the training dataset towards the higher end. Despite the possibility of simplifying the learning problem by filtering out high-speed data, congestion zones are often immediately followed by high-speed areas. To ensure our controller's competent behavior at high speeds and during transitions from high to low-speed zones, the training dataset retains both low and high-speed data.

Developing the Training and Evaluation Framework

To exploit the gathered data, a single-lane training environment is designed where an AV follows the trajectory data recorded from human drivers. The human driver is simulated at the front of a vehicle platoon, followed by the AV, and then a number of vehicles operating according to the *Intelligent Driver Model* (see Chapter 2). This setup guarantees the growth of waves in congestion, due to the string-unstable nature of the chosen IDM parameters. While a comprehensive micro-simulation of I-24 might allow training with more complex vehicle interactions, the proposed simulator focuses on realistic driving dynamics representative of the highway's wave types and drivers' reactions to wave formation. The simulator's efficiency is also commendable as it achieves 2000 steps per second, whereas a

micro-simulation of the complete 14-kilometer section would be computationally expensive due to the thousands of vehicles in congestion.

9.3 Design of the Controller Architecture

We introduce the **MegaController**, a control framework, depicted in Figure 9.3, for the mixed autonomy traffic flow problem. Mixed autonomy refers to the setting in which some vehicles are automated, while other vehicles are manually controlled (level 0 automation). The primary design paradigm of the proposed control framework is to achieve two goals: hierarchy for task allocation and modularity for control flexibility.

We designed a hierarchical structure to efficiently coordinate the control objectives in the optimization of macroscopic traffic flow optimization [16] and microscopic vehicle control [5, 129, 23]. Additionally, we addressed the computational task allocation challenge between the server and vehicle sides. There is an inherent interplay between the two components, as the Speed Planner communicates downstream events to the Vehicle Controller, and vehicles are equipped to transmit observations to the server for data aggregation.

Use of modular design

We employ a modular structure for two crucial purposes: (1) to facilitate diverse controller design approaches and (2) to accommodate a heterogeneous vehicle fleet with varying sensing and actuation capabilities.

As long as each controller utilizes the available interfaces, exchanging controllers becomes technically straightforward within our modular design. This decision supports a wide range of expertise within our team, allowing exploration of different designs with distinct technical focuses. While integration decisions based strictly on type matching may present subtleties related to stability, convergence, and other properties, our modular approach effectively addresses a key challenge. The dynamics of the roadway and our control understanding may only fully emerge during testing. Since testing may only be possible at a scale when we can evaluate its impact, we will be very close to the full test deadline when we make final decisions. Embracing modularity enables us to use data from tests conducted less than 24 hours earlier to inform controller decisions for subsequent tests, even close to the final deadline.

The MVT fleet comprised vehicles from three different years, makes, and models: 2023 Nissan Rogue, 2022 Cadillac XT5, and 2020-2021 Toyota Rav4. Due to trade secrets and design variations, achieving a uniform system interface for sensing and control is not feasible. For instance, although each car features adaptive cruise control using forward-facing radar for safety, the system couldn't access data from these sensors for all vehicle types. While designing a controller for a specific vehicle type may enhance controllability, it diminishes the potential societal impact when deployed at scale.

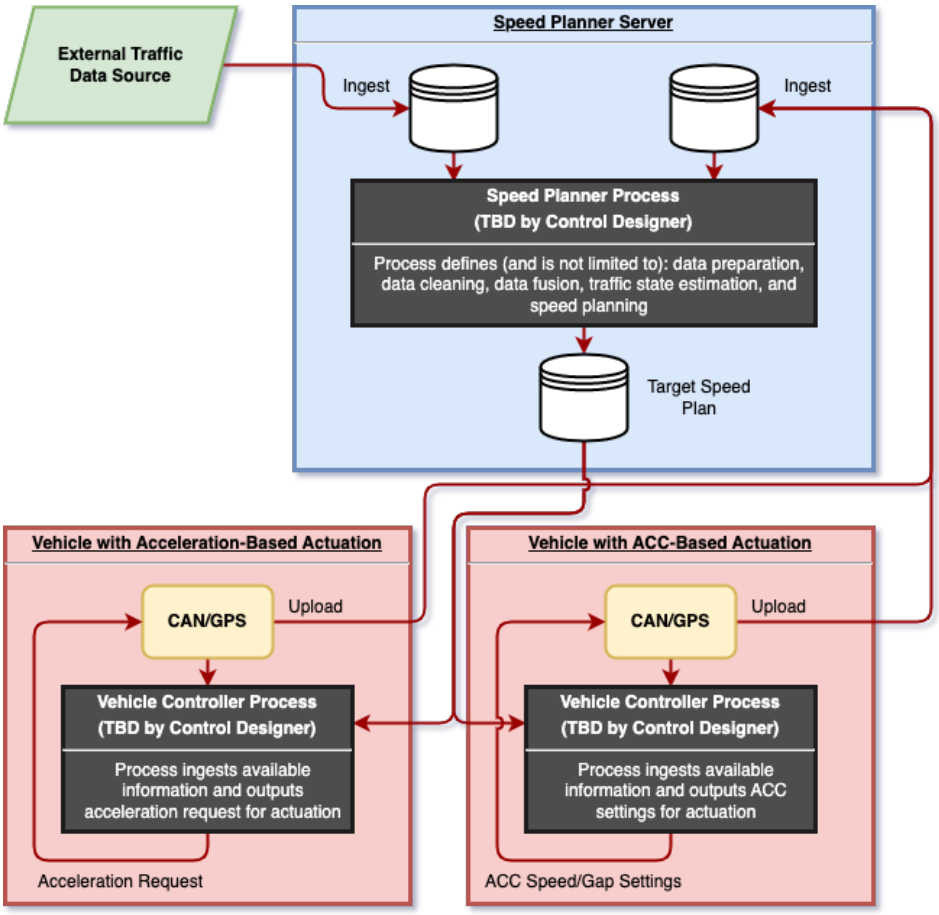


Figure 9.3: The MegaController's architectural framework is characterized by its hierarchical and modular structure, providing increased adaptability in design choices and the management of diverse sensing and actuation capabilities within the heterogeneous fleet. The centralized Speed Planner unit is depicted in blue, while the decentralized Vehicle Controllers, tailored to each vehicle's distinct control architecture, are represented by red boxes. These components collaborate to accomplish overarching objectives related to flow smoothing.

Hence, the system's strength lies in its ability to operate across vehicles with diverse sensing and control modalities. While the interfaces to the Speed Planner remain the same, different vehicles will have different implementations that have completely unique characteristics compared to other vehicles, similar to textbook design patterns in software engineering for producer/consumer architectures. This flexibility allows adaptation to a heterogeneous fleet, opening possibilities for various vehicle configurations and constraints.

In the MegaController design, there are two distinct components:

- A server-side **Speed Planner**, which is a centralized planner unit that provides high-level macroscopic speed suggestions based on periodic state updates from the distributed vehicles and external macroscopic data source, like INRIX employed in this study. The algorithms deployed on the server-side are designed to handle computationally-heavy data aggregation and macro-state tasks.

- A vehicle-side **Vehicle Controller**, which is a networked decentralized controller [14] that commands local actuation of the vehicle. The algorithms deployed here take into consideration the target speed suggested by the server-side planner unit, the latency of that information, and any observations from the vehicle's onboard sensors.

Borrowing conceptually from object-oriented programming, the interfaces are agreed upon *a priori*, allowing an abstraction of individual components for ease of design and testing.

Speed Planner

The goal of the Speed Planner is to enhance the overall efficiency of traffic flow by determining macroscopic guiding speeds for autonomous vehicle fleets based on real-time traffic data sourced from various heterogeneous channels. While our deployment aimed at optimizing flow efficiency, our framework accommodates arbitrary objective functions.

The data sources at our disposal include recent vehicle ping messages from our autonomous vehicle fleet, generated using our custom hardware and software stack (detailed later in Figure 9.8), as well as vehicular velocity data within distinct sections of highway through INRIX, a provider of real-time data for traffic, parking, and other transportation data. INRIX data hav been previously examined in the literature for its relevance in traffic estimation and validated for accuracy through *post hoc* methods [149].Utilizing these heterogeneous data sources posed specific challenges and considerations, notably the unknown latency of external data sources and differences in spatial and temporal resolution when fused with our vehicle message data. While the specific aggregation algorithm of INRIX data is not publicly disclosed, literature such as [70] explores its latency, indicating delays ranging from 3-5 minutes, and occasionally up to 10 minutes. A summary of data specifications is provided in Table 9.1.

Addressing the challenges at hand, the Speed Planner serves two primary functions: (1) Traffic State Estimation (TSE) enhancement and (2) designing the Speed Plan, which represents an ideal target speed profile. The TSE enhancement module is designed to mitigate the

Data Source:	INRIX	AV Ping
Period (seconds)	60	1
Latency (seconds)	∼180	negligible
Segment Length (miles)	∼0.5 (varied)	N/A
Lateral aggregation	Lane averaged	Lane specific

Table 9.1: A summary table outlining the data sources used for traffic state estimation and subsequent processes. INRIX aggregates data from a fleet of vehicles on the road. The AV Ping data originate from our own fleet, where each car posts speeds to the server through an API that contains speed (from CAN) and positioning (from GPS), as well as timing and vehicle identity information.

impact of inherent latency in third-party data sources, aiming to refine the spatial and temporal resolution of the input traffic data. The target design module generates Speed Plans with the goal of reducing vehicle energy consumption and increasing the overall throughput of the traffic flow. Various methodologies, such as kernel smoothing [46], ODE/PDE-based flow control [157], and vehicle trajectory optimization [5], are employed to develop the target design module.

Vehicle Controller

The Vehicle Controller enables each car to use traffic information provided by the Speed Planner in context with local state, such as the distance to the vehicle in front, relative speed, etc. The Vehicle Controller aims to improve the flow of traffic while ensuring the safety of both the AV and other vehicles around it.

Input data for the Vehicle Controller are obtained from the in-vehicle network, and may include other inputs such as the target speeds from the Speed Planner. A subset of data from the CAN bus are made available to the vehicle controller, and vary depending on the vehicle make/model/year. Realizing control algorithms on vehicles required new computer hardware, electronics, and software development [29, 42, 87, 22, 101, 102] because of the heterogeneity in vehicle data and control interfaces.

Depending on the vehicle platform, we have explored options including speed-based control [28, 34], acceleration-based control [87, 88], and ACC-based control [65]. Speed-based control was initially explored, but then deprecated as acceleration based control best matched desired goals from algorithm designers. The acceleration-based and ACC-based approaches fit into the overall architectural design. Torque-based control (that is, applying the requisite torque directly to the gears) would be the most prescriptive control with tightest actuation, but it also requires a level of CAN bus access that prevents quick return of the vehicle to stock state. The majority of the vehicles used in the experiment were stock vehicles that were later returned to the manufacturer, so all changes to enable experimental control needed

to be minimal and non-invasive changes to the vehicle. Therefore, for this deployment, our architecture included two types of control capabilities on our heterogeneous fleet, which are summarized in Table 9.2.

Vehicle(s):		Toyota RAV4	Nissan Rogue, Cadillac XT5
Actuation Type:		Acceleration	ACC
Sensing	Ego Position	✓	✓
	Ego Speed	✓	✓
	Ego ACC Speed Setting	✓	✓
	Ego ACC Gap Setting	✓	✓
	Leader Space Gap	✓	✗
	Leader Minicar	✓	✓
	Leader Relative Speed	✓	✗

Table 9.2: A summary table of the sensing and actuation available to our system, for the heterogeneous vehicle fleet. Space gap refers to the distance between the ego vehicle's front bumper and the leader vehicle's rear bumper. The minicar refers to a boolean indication whether the ego vehicle's sensors detect the presence of a leader vehicle (an approximate 80-100 meter maximum distance). Relative speed is the leader vehicle's speed minus the ego vehicle's speed.

ACC-based Controller

We demonstrated that it was possible to provide effective yet safe control by is to updating vehicle's *Adaptive Cruise Control* (ACC) set points electronically. This mimics what the driver can do through buttons on the steering wheel, and has the benefit of keeping the safety features of the car's ACC systems in the loop. This approach is contrasted with the acceleration-based approach, in which vehicles are controlled via actuations to their acceleration, in a few ways, notably:

- The controllability of the system is affected and the bandwidth of the controller is likely reduced, since the stock ACC has its own gains and modes.
- The multi-model design features of the acceleration-based approach (namely the lane-change recovery controller and safety wrapper) are not needed here, as those features are built into the stock ACC algorithm.

To understand the changes in controllability, we explored the vehicle's ACC dynamics with a goal to make design decisions on whether rate limits or other constraints on the input signals would be required for stability or performance reasons. Every vehicle make and

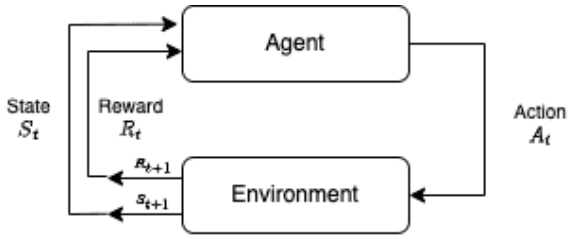


Figure 9.4: The Markov Decision Process that RL is based on. An agent exists in an environment and repeatedly chooses action based upon a state and receives rewards, which then informs the agent on the value of the state and action pair. © 2024 IEEE.

model equipped with ACC has its own unique mapping of state space (for example, ego speed, space gap, leader speed, etc.) and ACC inputs (speed setting and gap setting) to actuated acceleration. Architecturally, we take as given that this model is available for the design of the base ACC controller.

As a result, the only component of the ACC-based controller that requires specific designing is the base controller. Just as with the acceleration-based controller, the base controller is performance-based, striving for some performance goal. Similar design philosophies are applied as before. Uniquely different, as noted above, is that this base controller does not need to explicitly consider safety and vehicle-specific dynamics—the controller design can implicitly account for the vehicle-specific dynamics by utilizing the ACC model in a feedback loop.

9.4 Design of Controller Components

Here, we describe in detail how each of the modular components was designed. The discussion is in a parallel structure to the previous section, with particular focus on controllers that were actually implemented for deployment. In the lifespan of the project, we explored many approaches for designing the controller, which contributed key concepts that helped the ultimate implementation, but will not be covered in this thesis.

Speed Planner

Based on the hierarchical framework, Figure 9.5 illustrates the deployment of the speed planner tested in the MVT. As outlined in Table 9.1, INRIX and probe vehicle data are transmitted to the central database at varying frequencies: each vehicle posts its data approximately every 1 second, while server-side processes insert new INRIX data approximately every 60 seconds. Notably, the INRIX data furnishes a singular speed value across all lanes, while the AV pings supply lane-level speed information.

The process of publishing a Speed Plan unfolds in the following steps:

1. Each new INRIX update is merged with historical INRIX data and input into the prediction module.
2. Vehicle observations from the preceding 60 seconds are retrieved and combined with the INRIX prediction to derive a lane-level traffic state estimate.
3. The lane-level TSE undergoes smoothing using the forward-kernel average.
4. Bottlenecks are identified via the smoothed lane-level TSE.
5. In the presence of a standing bottleneck in the lane, a deceleration region is defined as a buffer segment. For all other regions, the smoothed lane-level TSE serves as the lane-level Speed Plan.
6. Publish the Speed Plan for all lanes.

The subsequent sections discuss the intricacies of the TSE Enhancement and Target Design in detail, providing a comprehensive understanding of the procedure described above. Wang et al. [157] provides a more in-depth methodology description of the Speed Planner.

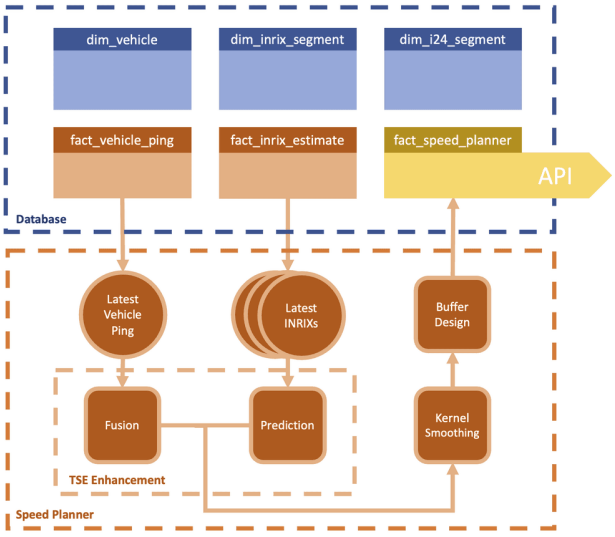


Figure 9.5: **Data Pipeline and Major Function Modules of Speed Planner** At the beginning of each update, the Speed Planner retrieves a mixture of macroscopic TSE and vehicle observations from the corresponding factual tables (`fact_inrix_estimate`, `fact_vehicle_ping`) in the database to compute the target speed profile. The raw TSE is used as the input of the prediction module, whose output is fused with vehicle observations. The resulting fusion is then smoothed and used in the buffer design module, with the output being stored in the database (`fact_speed_planner`) and published as the target speed profile.

TSE Enhancement

The TSE enhancement module is composed of two key elements: the INRIX prediction module and the data fusion module. The prediction module is designed to address latency issues associated with real-time INRIX data, particularly when applied to vehicular control, in contrast to its conventional use for general traffic insights. The fusion module complements this by integrating real-time data from our system's probe vehicles, enabling a more detailed, lane-level TSE with improved time-space precision. A comprehensive mathematical explanation of the TSE enhancement module for MVT implementation is provided in [157].

To achieve a lane-level TSE with superior spatial detail, we further partition the INRIX data into smaller units. After generating the INRIX prediction, it is combined with real-time data from the system's controlled vehicles. As each AV communicates with the server at a rate of 1Hz, we receive 60 ping records from each vehicle for every Speed Plan generation. These records assist in determining the average speed of each vehicle over the previous update period, subsequently updating the TSE for the corresponding sub-segment. Assuming consistent lane adherence by our drivers (largely confirmed through systematic data review), lane-specific TSE estimates can be generated by combining vehicle data with broader INRIX speed data. When it comes to data fusion, priority is given to data from our vehicles over INRIX data, considering the inherent characteristics of both sources. Our vehicles collect and transmit perception data through an observable and controllable system, featuring quantifiable error and latency as detailed in [114]. In contrast, the INRIX real-time API employs averaging techniques for forecasting over a given period. This approach may introduce inaccuracies at specific points within that timeframe, potentially impacting our control execution. The INRIX system thus operates as a somewhat opaque system, with its error and latency aspects largely inferred from provider descriptions. In this paper, we adopt the following notation to represent the discrete TSE:

$$\{(x_j, \bar{v_j}), j \in \mathcal{J}\}, \tag{9.2}$$

where j represents the index of road segments, x_j is the postmile coordination of the central of road segment j and $\bar{v_j}$ is the average speed of the corresponding road segment j. Wang et al. [157] details the procedure of data fusion applied in MVT.

Target Design

This section introduces the primary modules of the Target Design, specifically the kernel-based smoothing and the learning-based buffer design. The kernel smoothing involves processing the enhanced TSE at each time step using a selected kernel to improve fuel consumption induced by shockwaves in high-density traffic flow. The buffer design employs Reinforcement Learning (RL) to establish a buffer area upstream of the standing bottleneck, aiming to enhance throughput at the bottleneck. The RL-suggested target speed is integrated into a mathematical traffic model, represented by a strongly coupled system of Partial and Ordinary Differential Equations (PDE-ODE). The result of this mathematical model is the

identification of traffic density $(t, x) \mapsto \rho(t, x)$. Subsequently, the kernel smoothing utilizes this information to learn the velocity for the next time step.

In the kernel smoothing module, we leverage enhanced TSE data to synchronize the driving speeds of automated vehicles. In particular, vehicles are assigned target speed profiles based on shared and common traffic state information among all AVs.

At any fixed time step t, the desired speed profile $\mathbf{v} : \mathcal{R}_+ \times \mathcal{R} \to \mathcal{R}_+$ is extracted from the enhanced TSE via kernel methods. First, we preprocess the sparse TSE data by interpolating the discrete data pairs $(x_i, \bar{v}_i)$ to a continuous speed profile $(t, x) \in \mathcal{R}_+ \times \mathcal{R} \mapsto v(t, x)$, as an approximation of the average speed of a higher granularity traffic at a position x and at time t. Then, for any fixed time $t = t_\circ$ we obtain the desired speed by applying a kernel function $K(\cdot)$ at a position $x = x_\alpha$:

$$\mathbf{v}(t_\circ, x_\alpha) = \frac{\int_{x=x_\alpha}^{x_\alpha+w} K(x_\alpha, x) v(t_\circ, x) dx}{\int_{x=x_\alpha}^{x_\alpha+w} K(x_\alpha, x) dx}, \tag{9.3}$$

where w is the width of the estimation window. While many different kernel functions such as the Gaussian kernel, Triangular kernel, Quartic kernel, Uniform kernel etc., can be chosen; for the purposes of this paper, we consider a uniform kernel. The desired speed profile at a position x_α is defined as:

$$\mathbf{v}(t_\circ, x_\alpha) = \frac{\int_{x=x_\alpha}^{x_\alpha+w} v(t_\circ, x) dx}{w}. \tag{9.4}$$

When human drivers perceive a gap between their vehicle and the one in front of it, the leader vehicle, they tend to accelerate to close the distance. Our proposed desired speed profile aims to proactively slow down, though not excessively, to establish a gap from the leader vehicle. This approach takes into account the information provided by the TSE, which indicates the presence of congestion in the nearby downstream area. The proposed desired speed profile is adaptive to different traffic states and is relatively robustness, requiring only one parameter, w, to tune. For the buffer design, we consider the interval $\mathcal{I} \subset \mathcal{R}$ as the region of interest, with an additional subregion $\mathcal{I}_c \subset \mathcal{I}$ representing a congested area. The objective is to determine the controlled vehicle target speed at a time step $t_\circ$, denoted by $\mathbf{u}(t_\circ, x)$, such that the density $\rho(t, x)$ for $x \in \mathcal{I}_c$ and $t \geq t_\circ$ is distributed uniformly through the region $\mathcal{I}$. Determining the controlled vehicle's target speed is performed via the following procedure: (i) Designing a target speed $\mathbf{u}(t_\circ, x)$, given the input $\mathbf{v}(t_\circ, x)$ from the kernel smoothing step, (ii) Identifying the density $(t, x) \in \mathcal{R}_+ \times \mathcal{I} \mapsto \rho(t, x)$, given the target speed of the controlled vehicle, employing a strongly coupled PDE-ODE model of traffic flow, (iii) Evaluation step wherein the speed profile is updated by the smoothing kernel using the obtained density.

Detailed procedures for the above steps are provided in the paper [157].

ACC-based Vehicle Controller

Recall in the architectural design of the controller, the introduction of the Vehicle Controller layer features one particular method for actuating the vehicle: ACC-based control. Here, we

will discuss component implementations prepared for this method of control.

The ACC-based controller is the version of the controller that was ultimately deployed on 97 of the 100 vehicles during the final MVT. The crucial distinction between the ACC-based controller and the acceleration-based controller introduced in the prior section is the controller's output. The ACC-based controller, rather than providing an acceleration to actuate, provides outputs setpoints for the AV's native ACC system, which controls the AV's longitudinal movements based on those setpoints. We use the stock ACC system's safety assurances and lane-change handling. Thus while this provides a less direct form of control, it is more robust in ensuring the safety and smoothness of the ride. For context on the simulator that was developed for training this algorithm, please refer to 9.2.

The ACC-based controller is an RL controller that is trained using Proximal Policy Optimization [122]. Elements of training the Markov Decision Process (MDP) problem [91] are described below:

Observation Space

- v, velocity of the AV.
- v_s, target speed given by the Speed Planner.
- l, a "minicar" flag that indicates whether there is a leader vehicle detected, nominally within 80 meters (dubbed as such due to the miniature car icon that appears in the dashboard when the leader is detected).
- s, the current ACC speed setting
- g, the current ACC gap setting

Action Space

- The requested ACC speed setting, which our onboard computer will realize in a series of button presses (see Figure 9.6). The speed setting dictates the maximum speed at which the ACC can drive.
- The requested ACC gap setting, which the onboard computer will realize with priority over the speed setting. The gap setting takes on three bars between one and three, with each bar indicating a higher allowable gap. Each bar roughly corresponds to constant time gaps of 1.2, 1.5, and 2.0 seconds.

In addition, a clipping mechanism was used to ensure controller safety and social acceptability, particularly in the absence of leader state information. The *post facto* lower and upper bounds placed on the speed setting output are based on the average speed of the ego

Figure 9.6: A photograph of a Nissan Rogue's steering wheel buttons that control the vehicle's ACC. The ACC system is turned on by manually pressing the blue icon on the far right. Through our vehicle interfacing efforts, we are able to electronically press the + and − buttons to toggle the ACC speed setting up 1 mph and down 1 mph, respectively, or hold them to increment by 5 mph. The three ACC gap settings are rotated through by pressing the button on the bottom right. © 2024 IEEE.

vehicle during the last one second (ten timesteps, with 0 mph and null observations omitted):

$$\begin{aligned} v_{\text{lower}} &= \frac{1}{10}\sum_{i=1}^{10} v_i - 15\text{mph} \ , \\ v_{\text{upper}} &= \frac{1}{10}\sum_{i=1}^{10} v_i + 5\text{mph} \ , \end{aligned} \tag{9.5}$$

where v_{lower} and v_{upper} are the lower and upper bounds of the clip, respectively. The final clip is executed as:

$$\begin{aligned} v_{\text{clip}} &= \min(\max(v_{\text{action}}, v_{\text{lower}}), v_{\text{upper}}) \\ v_{\text{final}} &= \min(\max(v_{\text{clip}}, 20\text{mph}), 73\text{mph}) \ , \end{aligned} \tag{9.6}$$

where v_{final} is the final speed setting command and v_{action} is the un-normalized speed output, or action, from the neural net. The 73 mph maximum speed accounts for modest speeds in excess of the posted speed limit of 70 mph.

Reward Function The reward function is:

$$\begin{aligned} r_t =& 1 - c_1 a_t^2 - c_2 (v_t^{av} - v_t^{sp})^2 + ... \\ & - \frac{c_3}{n} \sum_{i=1}^{n} E_t^i - c_4 \mathbb{1}_{h_t^{av} \leq h_{min} \vee h_t^{av} \geq h_{max}} \ , \end{aligned} \tag{9.7}$$

where $c_1 - c_4$ are coefficients, a_t^2 is an acceleration penalty; $(v_t^{av} - v_t^{sp})^2$ is a squared penalty on the difference between the Speed Planner's suggested speed and the actual speed; $\frac{1}{n} \sum_{i=1}^{n} E_t^i$ is instantaneous fuel consumption according to the energy models described in 9.2; and the last indicator term is an intervention penalty that is invoked if the space gap is less than the minimum space gap or greater than the maximum space gap.

Further details on the development and design decisions for the state and reward representations and the intermediate and final policies are given in depth in [65].

9.5 Candidate Controller Selection

In the months leading up to the MVT open road test, our team developed a multitude of candidate controllers, as seen in previous sections and sidebars. In order to pare down the list of candidate controllers to be considered for real-world deployment in November 2022, various (Speed Planner, ACC-based Vehicle Controller)-combinations were assessed over a range of simulation scenarios. The simulation and testing framework follows from [78] and features an updated simulation methodology described in 9.2.

Simulation scenarios included shockwaves, bottlenecks, and freeflow. Shockwave and freeflow scenarios follow precisely from the simulator described in 9.2 and only differ in the leader trajectory (one being stop-and-go, and the other being all high speed). The bottleneck scenario features a dynamically imposed speed limit in a spatial region of the domain. The speed limit is inversely related to the vehicle density of the region, and the scaling parameter is tuned to historically observed speeds in the I-24 region. The bottleneck region is meant to model a weaving area where a close proximity of on-ramps and off-ramps results in increased lane-change frequency.

The key performance indicators (KPIs) used to assess the controllers include fuel economy, throughput, and network speed. Fuel economy is the overall miles per gallon of all vehicles, and it is computed by applying the energy model (see 9.2) and computing

$$\text{fuel economy} = \frac{\sum_{t=0}^{T} \sum_{i=1}^{n} E_t^i}{\sum_{i=1}^{n} x^i} \ , \tag{9.8}$$

where the numerator is the total fuel consumed by all vehicles over all simulation time, and the denominator is the total distance traveled by all vehicles. The throughput is measured by counting the number of vehicles crossing various positions along the highway, normalized by time. For the shockwave and freeflow simulations, this is taken to be the straight average of the time-average of five equi-spaced measurement locations. For the bottleneck,

this is taken to be the steady-state (or final value) at the measurement location just downstream of the bottleneck region. The network speed KPI is defined as the total distance traveled by all vehicles divided by the total driving time of all vehicles.

At the time of our self-imposed "controller freeze" (that is, the time at which substantial changes to controllers are no longer allowed), there were two Speed Planner variants and 12 Vehicle Controller variants. KPI performance for each of these controllers is shown in Figure 9.7. The Speed Planner variants are kernel smooth (just the first three steps described in Speed Planner) and kernel smooth with RL buffer (all steps described in Speed Planner). The Vehicle Controller variants include:

- **Simple**: a hand-designed logic-based controller that largely adheres to the Speed Planner suggestion.
- **MicroAccel**: the main controller described in Acceleration-Based Controller with modifications to output ACC set points. This controller itself has six variants with different parameter choices.
- **RL**: the RL controller described in ACC-Based Controller. This controller has five variants with different training meta-parameter choices.

In the shockwave scenario, an ideal controller should maximize fuel economy improvements and remain neutral on throughput and network speed. In the bottleneck scenario, an ideal controller should maximize throughput improvements and optionally improve or stay neutral on the other two KPIs. In the freeflow scenario, an ideal controller will not worsen on any of the KPIs. Given these criteria, we determined that the HybridRL Vehicle Controller paired with the Kernel Smooth with RL Buffer Speed Planner presented the ideal combination for the ACC-Based vehicles.

Server-side implementation

As illustrated in Figure 9.5, the server-side implementation encompasses the creation of the database, the API, and the Speed Planner algorithm scripts. The database schema is structured based on the star schema data model, comprising dimension tables (for storing static metadata) and fact tables (for storing quantitative data). The tables include:

- `dim_vehicle` for storing the vehicle metadata, route and lane assignment;
- `dim_inrix_segment` for storing INRIX road segment data for I-24;
- `dim_i24_segment` for storing finer-grained road segment data, which supports TSE and Speed Plan profiles after data fusion;
- `fact_vehicle_ping` for storing the realtime AV ping information;

SHOCKWAVE SCENARIO		Speed Planner					
		Kernel Smooth			Kernel Smooth with RL Buffer		
		Fuel Economy % Increase	Throughput % Increase	Speed % Increase	Fuel Economy % Increase	Throughput % Increase	Speed % Increase
Vehicle Controller	Simple	17.33%	1.18%	-0.32%	24.24%	-0.04%	-5.86%
	MicroAccel-a	9.33%	-0.08%	-0.37%	10.71%	1.18%	-0.75%
	MicroAccel-b	4.56%	0.26%	7.68%	8.87%	2.27%	2.87%
	MicroAccel-c	9.33%	-0.08%	-0.37%	10.71%	1.18%	-0.75%
	MicroAccel-d	9.33%	-0.08%	-0.37%	10.71%	1.18%	-0.75%
	MicroAccel-e	9.33%	-0.08%	-0.37%	10.71%	1.18%	-0.75%
	MicroAccel-f	12.20%	1.02%	0.37%	14.94%	0.35%	0.45%
	RL-a	14.31%	-1.11%	-1.72%	13.25%	-1.32%	-1.69%
	RL-b	12.38%	-1.10%	-1.87%	11.63%	-1.45%	-1.87%
	RL-c	-2.76%	0.82%	0.37%	-1.42%	0.36%	0.15%
	RL-d	3.24%	0.26%	0.67%	-0.53%	0.11%	-0.27%
	RL-e	23.18%	2.84%	-2.19%	18.47%	2.04%	-1.69%
	HybridRL	22.96%	2.75%	-2.29%	17.49%	1.69%	-2.14%

BOTTLENECK SCENARIO		Speed Planner					
		Kernel Smooth			Kernel Smooth with RL Buffer		
		Fuel Economy % Increase	Throughput % Increase	Speed % Increase	Fuel Economy % Increase	Throughput % Increase	Speed % Increase
Vehicle Controller	Simple	6.85%	13.57%	8.76%	8.44%	14.34%	12.63%
	MicroAccel-a	4.90%	12.77%	4.21%	3.97%	13.16%	3.87%
	MicroAccel-b	1.81%	13.55%	8.16%	-0.60%	13.16%	7.99%
	MicroAccel-c	4.90%	12.77%	4.21%	3.97%	13.16%	3.87%
	MicroAccel-d	4.90%	12.77%	4.21%	3.97%	13.16%	3.87%
	MicroAccel-e	4.90%	12.77%	4.21%	3.97%	13.16%	3.87%
	MicroAccel-f	1.26%	13.16%	3.35%	1.09%	12.39%	3.18%
	RL-a	4.25%	9.30%	4.47%	3.45%	9.30%	4.64%
	RL-b	3.83%	9.30%	4.55%	3.36%	9.30%	4.64%
	RL-c	0.88%	13.16%	3.18%	0.88%	13.16%	3.18%
	RL-d	0.91%	12.77%	3.78%	1.06%	8.82%	4.81%
	RL-e	8.22%	14.34%	8.08%	6.72%	14.73%	8.68%
	HybridRL	2.95%	14.73%	9.19%	2.92%	14.74%	9.19%

FREEFLOW SCENARIO		Speed Planner					
		Kernel Smooth			Kernel Smooth with RL Buffer		
		Fuel Economy % Increase	Throughput % Increase	Speed % Increase	Fuel Economy % Increase	Throughput % Increase	Speed % Increase
Vehicle Controller	Simple	1.13%	-0.56%	-2.14%	-2.79%	-1.24%	-1.81%
	MicroAccel-a	14.50%	5.13%	-11.63%	14.52%	5.21%	-11.53%
	MicroAccel-b	13.78%	4.49%	-12.73%	13.80%	5.90%	-11.63%
	MicroAccel-c	14.50%	5.13%	-11.63%	14.52%	5.21%	-11.53%
	MicroAccel-d	14.50%	5.13%	-11.63%	14.52%	5.21%	-11.53%
	MicroAccel-e	14.50%	5.13%	-11.63%	14.52%	5.21%	-11.53%
	MicroAccel-f	15.68%	0.33%	-15.38%	15.81%	0.63%	-15.20%
	RL-a	27.27%	2.50%	-25.19%	25.39%	2.65%	-22.38%
	RL-b	32.76%	2.33%	-23.78%	25.27%	3.47%	-21.54%
	RL-c	-0.19%	1.49%	0.43%	-0.03%	1.60%	0.26%
	RL-d	0.05%	0.12%	-0.86%	0.05%	0.12%	-0.86%
	RL-e	29.47%	1.41%	-34.28%	25.13%	2.35%	-27.52%
	HybridRL	-0.19%	1.49%	0.43%	-0.03%	1.60%	0.26%

Figure 9.7: Summary of controller assessments in simulation. Candidate controller components to the MegaController were simulated under three scenarios (top: shockwave; middle: bottleneck; bottom: freeflow) and evaluated against KPIs (fuel economy, throughput, network speed) relative to a baseline human drivers. Speed Planner candidates are shown across the top and Vehicle Controllers are shown in rows. KPIs are shown in columns. Results are color-coded from improved (green) to neutral (white) to worsened (red).

- `fact_vehicle_observation` for storing extracted vehicle observations from the AV ping table;
- `fact_inrix_estimate` for storing realtime INRIX data;
- `fact_speed_planner` for storing Speed Plan profiles.

The `fact_vehicle_ping` and `fact_inrix_estimate` tables are continually being updated in real-time at a rate of 1Hz. After processing these input data via the Python-based Speed Planner algorithm, the Speed Plan is stored in the `fact_speed_planner` table. This information is made accessible on the internet through a PHP-based HTTP API.

Vehicle-side implementation

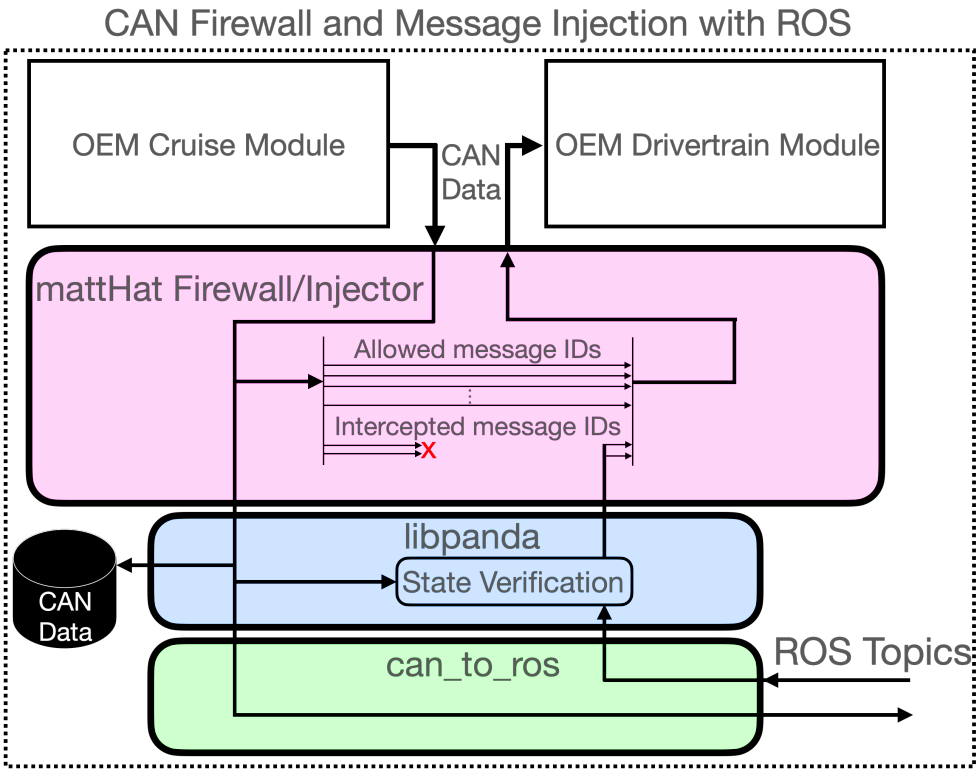


Figure 9.8: A unique CAN interface, named mattHat, serves as a hardware-level firewall for messages exchanged between OEM modules. It has the capability to replace blocked messages with third-party messages from software utilizing libpanda. Libpanda facilitates direct CAN recording as well as ensures verification of CAN data to be sent by checking OEM's CAN states. Reading and sending CAN messages are enabled through ROS topics, made accessible by adapters in can_to_ros. © 2024 IEEE.

The vehicle-side implementation comprises various software libraries and a combination of both custom and commercially available hardware components.

Over the past three decades, there has been a gradual adoption of the open Controller Area Network (CAN) protocol in vehicles for in-vehicle networking communication among diverse electrical modules. The incorporation of the CAN bus allows automotive manufacturers to minimize wiring complexity while designing intricate systems covering aspects such as engine diagnostics, infotainment, security, emissions, and more recently, Adaptive Cruise Control (ACC) and Lane Keep Assist (LKA). In the context of cars, the ACC system typically incorporates a sensor module to gauge the dynamics of the leading vehicle and a distinct controller module that communicates with the vehicle's transmission and engine. As these modules are physically situated in different parts of the vehicle, they communicate via CAN buses, where listening CAN analyzers can capture the information. Figure 9.8 shows the approach of tapping into a CAN bus for message reading.

While automotive manufacturers offer details regarding the structure of the in-vehicle network through wiring diagrams, specific information transmitted on the CAN bus is typically treated as confidential, likely due to trade secrets and safety considerations. Nevertheless, some companies, such as comma.ai [1] and Intrepid Control Systems [2], are venturing into the realm of customized vehicle autonomy. They achieve this by marketing modules that intercept CAN messages between vehicle modules, providing their own inputs based on custom controllers. This is only possible after investing time in decoding messages and reverse engineering the protocols necessary for injecting custom messages. Figure 9.8 illustrates the distinct electrical architecture required for CAN message interception and injection between OEM modules, as opposed to basic CAN reading.

In our research, we adopted a similar principle; however, the presence of additional protocols on the 2023 Nissan Rogue, which constitutes the majority of our control fleet, posed challenges for injecting commands. Despite this limitation, CAN data could still be read for real-time controller inputs. Due to supply chain constraints and the inability to inject CAN messages, a custom circuit board named mattHAT (Hardware Attached on Top) was developed, as depicted in Figure 9.9. The mattHAT served as a CAN interface and provided methods for sending Adaptive Cruise Control (ACC) button press commands using custom circuitry. The board was designed to plug into a Raspberry Pi 4, and custom wire harness cables were created for installation across the 97 Nissan Rogue vehicles, targeting specific CAN busses related to the vehicle's ACC state.

To operate the mattHAT's CAN interface and button spoofer, as well as record raw CAN and GPS data, a C++ based library called `libpanda`[29] was developed. `libpanda` also supports a USB-based GPS module for recording the vehicle's position and synchronizing the system's time. Additionally, a tool named Strym was created to decode and rapidly analyze data recorded by `libpanda` for CAN signal decoding and classification[24, 97, 98]. To assist control designers in utilizing `libpanda`, the Robot Operating System (ROS) was installed on the Raspberry Pi. A set of ROS nodes were implemented in the software project `can_to_ros`[42, 101], which abstracts various sensors and actuators into a set of ROS topics. Another tool, `bagpy`, was developed to efficiently process and visualize recorded ROS data

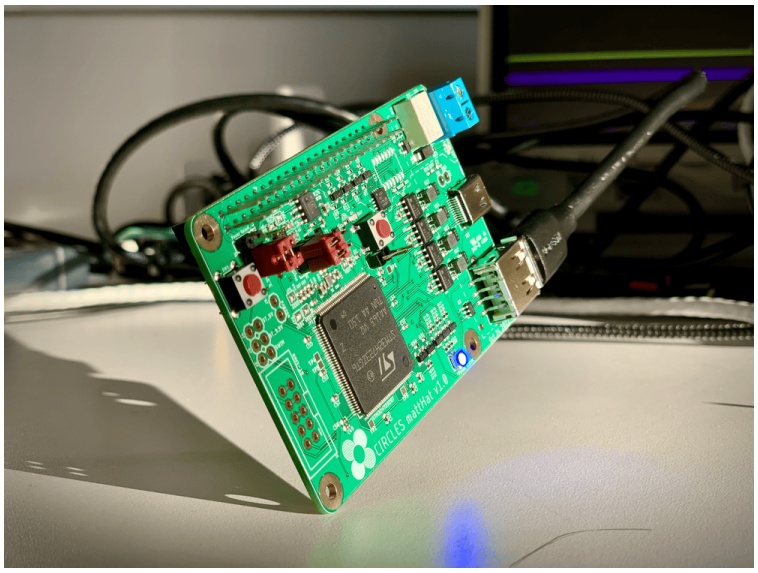


Figure 9.9: The mattHat: a custom designed circuit board rapidly manufactured for both CAN interfacing and ACC button command spoofing. © 2024 IEEE.

in the format of bagfiles[21]. With the ROS infrastructure in place, control designers could leverage modeling software like MATLAB's Simulink to design controllers and generate code, simplifying the integration of controllers.

`libpanda` includes several auxiliary services designed to facilitate scalable vehicle management. An automated method for performing Over The Air (OTA) updates was implemented, eliminating the need for manual handling of vehicles. An additional server integration, named `piStatus`, was developed to enable the system to regularly report its status, providing an easy means to assess the system's health. Figure 9.10 illustrates personnel at headquarters monitoring `piStatus`, ensuring vehicles pass status checks before deployment. Continuous monitoring of `piStatus` throughout each experiment helped identify any hardware issues promptly, enabling vehicle maintenance before the next deployment. A shutdown script, working in conjunction with an off-the-shelf battery-backup Pi HAT, automatically uploads data at the conclusion of an experimental drive when the vehicle is turned off. Additionally, as depicted in Figure 9.11, a WiFi antenna installed in the center of the vehicle parking lot provides each vehicle's embedded computer with internet access for data upload and OTA updates. The implementation of these services was crucial for effectively managing and maintaining a project of this scale.

I-24 MOTION

The CIRCLES team deployed vehicle controllers on I-24 southeast of Nashville, TN, leveraging the newly established I-24 Mobility Technology Interstate Observation Network (MO-

Figure 9.10: The `piStatus` web interface is monitored by one of the hardware team leads to approve vehicles for experiment deployment. © 2024 IEEE.

Figure 9.11: A WiFi antenna positioned at the center of the vehicle parking lot facilitates Over The Air (OTA) data uploads and software updates for each Raspberry Pi within every vehicle. © 2024 IEEE.

TION) testbed [50, 51]. Covering a four-mile section, I-24 MOTION utilizes 276 cameras on fixed roadside poles (110–135ft tall) to generate ultra-high resolution trajectory data of all vehicles. The testbed, designed for traffic science and experimentation on automated vehicles and traffic management, proved ideal for the live CIRCLES experiment.

Raw video processing by I-24 MOTION occurs in two stages. Firstly, a computer vision pipeline [49, 53, 52] conducts initial vehicle detection and type classification, including

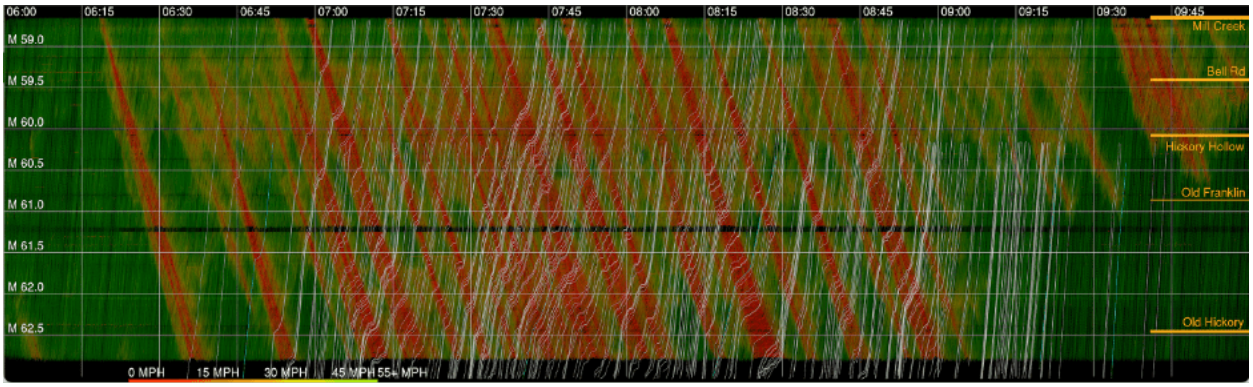


Figure 9.12: The time (horizontal axis) space (vertical axis) diagram, generated by I-24 MOTION [50] and its associated visualization library [175] during the MVT, illustrates vehicle trajectories heading westbound (up). Trajectories are color-coded based on vehicle speed (green: freeflow to red: congested). The trajectories of experiment vehicles are superimposed in white. © 2024 IEEE.

3-D bounding boxes for sedan, midsize, pickup, van, semi, truck, and motorcycle. Tracking detected vehicles across adjacent camera views, the processing is distributed across ten servers. Vehicle trajectories consist of at least ten fragments, with additional fragmentation possible due to occlusion. The computer vision pipeline converts image space coordinates into a roadway coordinate system using an hourly calibrated homography transformation. Secondly, post-processing algorithms [159, 158] stitch fragmented trajectories using an online minimum cost network flow graph problem. Reconciliation procedures ensure feasible and smooth higher-order dynamics, formulated as a quadratic program. A data visualization library [175] aids in interpreting generated datasets.

Due to the data's critical nature, I-24 MOTION retains a secure backup. Raw imagery is inaccessible outside I-24 MOTION administrators, ensuring privacy. This backup allows re-processing of video and/or raw vehicle detections, addressing known errors and limitations through re-processing rounds, including hourly recalibrated homography transformations [52].

Experimental Design

The live traffic experiment took place on I-24 in November 2022. With 100 vehicles to deploy, over 150 drivers were hired from local colleges, security guards, delivery drivers, team members' relatives, and elsewhere. The drivers were trained to adhere to specific assigned routes and lanes and to activate our custom ACC system whenever driving in their assigned lanes. Safety features were implemented such that the controller would default to stock ACC behavior if engaged off of the highway.

The specific routes driven by the AVs were chosen to maximize the penetration rate where

the I-24 MOTION system [50] would capture the effect on the surrounding commuters (bulk traffic). Initially, a single-loop route that would have the AVs circulating from 7:30am to 9:30am was considered. It was determined that this would likely cause increased congestion on the Exit 57 off-ramp, leading to extended queuing and potential spillback onto I-24. For this reason, we divided our AVs into two groups with the two different routes partially overlapped on I-24.

The AVs are released at 6:00am in order for all 100 to be deployed by the target time 7:30am. The drivers are instructed to repeat their driving routes for 2 hours, or sooner if they wanted a break. On a return trip, they exit the highway at Exit 60 (Hickory Hollow), to return to the Field Headquarters parking lot and returned their car keys at a desk before leaving the lot. As AVs return to the lot for a break, other drivers (already on break) head down to a queue to come out to the lot when needed. With this smoothly running rotation system, we were able to keep the 100 AVs on the routes during peak hours. For more details about the routes, and logistical choices made for the experiment, driver training, the daily schedule, or the penetration rate estimates, see the article [9].

Data Release

In this publication, we are publicly releasing a dataset from the MVT, which includes data collected by the AVs and the I-24 MOTION system [51]. Additional datasets generated by I-24 MOTION can be found in [49, 52, 53].

The dataset comprises GPS and CAN data from the 100 AVs across all testing days. Post-processing adjusts the longitudinal position for systematic GPS unit errors, determined by median delta computation. All other data is provided unaltered, except for interpolation onto a fixed 10Hz time grid. Lane identification is not given; assuming the driver's assigned lane when the controller is engaged is a reasonable assumption.

I-24 MOTION trajectory data is processed [158, 159] and supplemented with extra details. Speed and acceleration are inferred from the processed longitudinal position, ensuring smoothness. Road grade information is added using a parametric road grade map model. Fuel information, including various fuel rates and reference trajectories, is appended using fuel rate models with a one-to-one mapping of vehicle classes to fuel model classes. Each trajectory segment has a reference trajectory constructed to minimize the functional $\max_t a(t) - \min_t a(t)$ subject to the additional constraint $v(t) \geq 0$. This optimization minimizes peak accelerations and peak braking. Relative distances and vehicle IDs for nearest upstream/downstream, engaged-or-not AVs are provided based on implicit lane and longitudinal matching with the AV dataset.

Further information on the structure of the data will be provided in the data documentation released with the data.

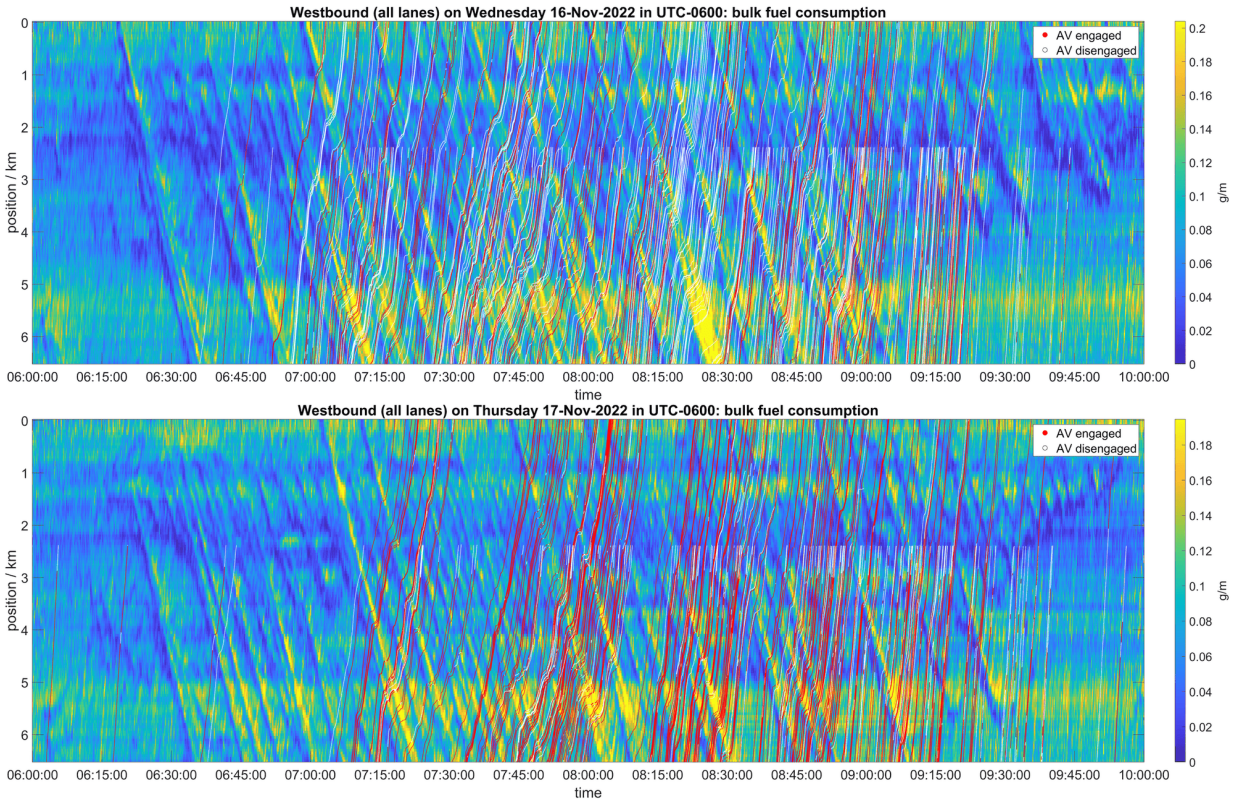


Figure 9.13: A heatmap illustrating bulk fuel consumption across time (horizontal axis) and space (vertical axis) is generated from I-24 MOTION [50]. The data aggregates all vehicles (across lanes) moving in the Westbound direction. Overlaid on the heatmap are AV trajectories, distinguished by color (white indicates controller disengaged, red indicates controller engaged). Heatmaps represent data from November 16, 2022 (top) and November 17, 2022 (bottom). © 2024 IEEE.

9.6 Results

The experimental, observational, data collection, and data processing framework described above generated a large amount of data, capturing every single vehicle on a highway during the deployment of 100 controlled vehicles. Analogous to previous seminal traffic data sets, such as NGSIM [147], the new data are expected to inspire and enable many subsequent findings. To highlight this potential, we here present some key first findings and insights, based on an analysis of the data with a macroscopic perspective.

We construct macroscopically meaningful fields in time–space, most prominently a field that shows the energy (in)efficiency of traffic at large on the I-24 highway segment. This is achieved by applying Edie's method [41] on boxes of size $h_t \times h_x$, where $h_t = 10$s and $h_x = 200$m, to the I-24 MOTION trajectories to construct the following fields:

- vehicle density $\rho(t, x)$, as the total vehicle time spent in each box, divided by the size of the box, $h_t \cdot h_x$;
- flow rate $q(t, x)$, as the total distance traveled in each box, divided by $h_t \cdot h_x$; and
- fuel rate density $f(t, x)$, as the total fuel consumed in each box, divided by $h_t \cdot h_x$.

From these fields, other meaningful fields are obtained, such as the bulk velocity field $u(t, x) = q(t, x)/\rho(t, x)$, the bulk fuel rate $\phi(t, x) = f(t, x)/\rho(t, x)$, and the bulk fuel consumption $\psi(t, x) = f(t, x)/q(t, x)$. The latter quantity $\psi(t, x)$, measureable for instance in grams per meter, represents the fuel demand per distance traveled of all vehicles in the $h_t \times h_x$ vicinity of the position (t, x). Figure 9.13 shows $\psi(t, x)$ for two experimental days: Wednesday November 16 and Thursday November 17, 2022. Each plot is overlaid with the trajectories of all control vehicles. This represents the first time that a complete time–space diagram of the energy inefficiency of traffic, based on accurate trajectories of *all* vehicles on the roadway has been provided.

In the same spirit as the purely microscopic Figure 9.12, one can, for both days shown in Figure 9.13, clearly see the traffic waves traveling backwards along the highway, as well as the increased fuel consumption incurred in these waves. The figure also shows the increased fuel demand in the uphill segment between $x = 5$km and $x = 6$km, and the reduced fuel demand in the downhill segment thereafter. The AV trajectories are colored red when the automated controller was activated, and white when the vehicle was under human control. The two shown test days were quite different in terms of the engagement rates of the controllers: on 11/16, the controllers were engaged 38% of all times, while on 11/17, the engagement rate went up to 78%. This difference was caused by a combination of increased driver comfort with the automation and a more reliable communication of traffic information to the vehicles on 11/17.

Given the low penetration rate of the control vehicles on the highway, it was not expected that they would completely smooth out all traffic waves—and the plots in Figure 9.13 confirm that expectation. However, the AVs may still have had some positive contribution on the energy efficiency of the flow at large. Whether that was in fact the case, we first note that the macroscopic plots like in Figure 9.13 allows for a targeted inspection of different regions of interest in time-space. For instance, on 11/16, there is a distinct region of high fuel inefficiency, around time 08:25:00 and location 5.0km–6.5km; and notably, this high-fuel region coincides with all AV controller being inactive. In contrast, on 11/17 such clusters of inactive AVs did not occur—and the fuel consumption map does not exhibit similarly large high-fuel regions. Another notable anecdote occurs on 11/16 in the wave that goes through $x = 2.3$km at time 07:00:00. First two active AVs notably dampen the wave; then the wave keeps on growing while four inactive AVs run through it (at $x = 3$km); followed by several active AVs (around $x = 4$km) notably dampening the wave again.

9.7 Conclusion

This work describes the control architectures and implementations of a 100 automated vehicle deployment to improve traffic efficiency on a freeway using a small fraction of automated vehicles. It is the largest field experiment to use CAVs to regulate the overall traffic flow, and the deployment strategy enabled algorithms from diverse fields spanning model based control to reinforcement learning. The control strategy presented in this work was a hierarchical control approach in which the upper level speed planner provided target velocities to a lower level control law responsible for performance and safety. These algorithms were deployed in the largest field experiment of its kind, on a heterogeneous vehicle fleet using low cost vehicular instrumentation. The data from the vehicles were combined with datasets generated from I-24 MOTION [50, 51], providing a large data resource for further study on the interaction of control vehicles on bulk traffic flow.

9.8 Acknowledgement

This material is based upon work supported by the National Science Foundation under Grants CNS-1837244 (A. Bayen), CNS-1837652 (D. Work), CNS-1837481 (B. Piccoli), CNS-1446715 (B. Piccoli), CNS-1446690 (B. Seibold), CNS-1446435 (J. Sprinkle), CNS-1446702 (D. Work), CNS-2135579 (D. Work, A. Bayen, J. Sprinkle, J. Lee), and by the French CNRS under the grant IEA SHYSTRA and PEPS JCJC (A. Hayat). This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Vehicle Technologies Office 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. The authors are grateful for the additional support provided by C3AI, Amazon AWS, Siemens, Toyota, GM, Nissan, Caltrans, CCTA, KACST, and the Tennessee Department of Transportation.

Chapter 10

Conclusion and Future Directions

Deep reinforcement learning, with its ability to capture structure in complex, nonlinear data, has shown to be effective not only in traffic systems as was explored in this thesis, but also in a number of other applications, from robotics, gaming, to large language models (LLMs). In this work, we showed how deep RL controllers are capable of exacting efficient behaviors in order to improve efficiency in a wide variety of metrics, from fuel or energy usage to velocity. We examine the processes that are necessary to do this, including developing or sourcing a reliable and efficient simulator for generating data to train on, and extensive RL design. We see the ways in which close and rapid iteration on RL design are important for producing stable and reliable controllers. Anywhere from the design of the reward function to modifications of value functions can have a major impact on the performance of the controller.

Once a sufficient policy has been trained in simulation, we also examined the challenges involved in moving this policy to a real-world scenario. With minor to severe discrepancies arising from factors like model mismatch to domain mismatch, we explored methods for mitigating the decline in performance that is often associated from the policy transfer. An RL algorithm that is more robust to changes is also more robust to policy transfer from simulation to the real world. We then combined the two first parts of this thesis, deep RL and robustness, to fulfill the goal of this work, which was to examine the effects of RL on a real world highway. Deploying 100 RL-controlled AVs on the I-24 highway in Tennessee, we dissected the significant RL training and engineering work that resulted in, to our knowledge, the largest road test of AVs designed to smooth traffic.

Moving forward, a plethora of open research questions remains. While this work always considered real world deployment in mind and thus constrained the bounds of the problem according to the constraints of the real world (e.g. access to expensive sensors or lack of infrastructure), it is interesting to consider the potential that RL may have without such constraints. Ahead, we present a few areas for future exploration.

10.1 Unbounded Constraints

For the majority of this work, RL training was always done with real world deployment in mind. Going into training with these constraints is appropriate and will result in pragmatic solution that is compatible with needs and requirements of a deployment system. Some of these constraints include factors such as:

- **Sensor equipment**: Do we have access to RADAR? LIDAR? Cameras? What is the latency of data flow?
- **Hardware**: What kinds of hardware do we have access to? What kinds of states and sensing equipment does the hardware come with? If it doesn't, how much work or how expensive is it to create access to the desired level?
- **Technological Horizons:** Does this technology even exist? Do we or would we ever have access to controlling the phases of a traffic light in a particular corridor? Do we have the means to control not only an automated fleet, but a connected and automated fleet?

With these questions and points in mind, it is interesting to consider exploring the extents of RL without these constraints. As such, a more detailed state or reward space that could be employed, therefore capturing more useful sensory detail from which the RL agent could potentially learn far more effectively.

10.2 Updated Driver Models

For all the experiments in this work, we used the Intelligent Driver Model (IDM) to model the behavior of human drivers. Although it is a state of the art driver model, it models on the assumption that all vehicles are human-driven. One potentially interesting path to consider is developing a model of human driver behavior around their reactions to a self-driving or automated car. Since self-driving companies eventually aim to deploy completely unsupervised vehicles without a safety driver, developing some model of these behaviors seems prudent. Along with the AV data that is collected during rollouts, this additional modeling could allow for more intentional fine-tuning around unexpected behavior. While the majority of self-driving companies operate on an end-to-end stack, an AV-aware human driver model could allow for a number of training optimizations. For instance, for strange and rare behaviors that occur during deployment, it would be easier to augment data via new simulation trajectories that would provide more data with which to analyze problems.

10.3 Generative AI

With foundational models and generative AI becoming a staple of the ML world, a potentially exciting direction to explore is how these models can contribute to the self-driving

problem. With major advances in autoregressive technology, a focus on transformer-based neural network models will likely have a large impact on self-driving technology. With the ability to more seamlessly analyze historical information and gather implicit data on the environment on which its deployed, such an approach could provide a more flexible and robust solution than ones that we have seen. Further, considering the large size of these models, explorations into model distillation would be salient in order to effectively port these large, complex models into smaller, simpler models that can be deployed locally in AVs. The goal would be to maintain as much of the performance of the larger model, but with reduced latency and computational costs.

Bibliography

[1] URL: https://comma.ai/.

[2] URL: https://intrepidcs.com.

[3] A. M. Bayen et al. "CIRCLES: Congestion Impacts Reduction via CAV-in-the-loop Lagrangian Energy Smoothing". In: Presented at the 2020 Vehicle Technologies Office Annual Merit Review, Washington, DC (virtual), 2020.

[4] A. M. Bayen et al. "CIRCLES: Congestion Impacts Reduction via CAV-in-the-loop Lagrangian Energy Smoothing". In: Presented at the 2021 Vehicle Technologies Office Annual Merit Review, Washington, DC (virtual), 2021.

[5] Arwa Alanqary et al. "Optimal Control of Autonomous Vehicles for Flow Smoothing in Mixed Autonomy Traffic". In: *(to appear) 2023 IEEE 62nd Conference on Decision and Control (CDC)*. IEEE. 2023.

[6] Saleh Albeaik et al. "Limitations and improvements of the intelligent driver model (IDM)". In: *SIAM Journal on Applied Dynamical Systems* 21.3 (2022), pp. 1862–1892.

[7] *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*. Jan. 2019. URL: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

[8] Shun-Ichi Amari. "Natural gradient works efficiently in learning". In: *Neural computation* 10.2 (1998), pp. 251–276.

[9] Mostafa Ameli et al. "Designing, simulating, and performing the 100-AV field test for the CIRCLES consortium: Methodology and Implementation of the Largest mobile traffic control experiment to date". In: *IEEE Control Systems Magazine* (2024).

[10] Kai Arulkumaran et al. "Deep reinforcement learning: A brief survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[11] Michael Athans. "A unified approach to the vehicle-merging problem". In: *Transportation Research* 3.1 (1969), pp. 123–133. ISSN: 00411647. DOI: 10.1016/0041-1647(69)90109-9.

[12] *Autonomous Vehicles — content.naic.org*. https://content.naic.org/cipr-topics/autonomous-vehicles. [Accessed 04-Mar-2023].

[13] Andrea Bacciotti and Lionel Rosier. *Liapunov functions and stability in control theory*. Springer Science & Business Media, 2005.

[14] John Baillieul and Panos J. Antsaklis. "Control and Communication Challenges in Networked Real-Time Systems". In: *Proceedings of the IEEE* 95.1 (2007), pp. 9–28. DOI: 10.1109/JPROC.2006.887290.

[15] Paolo Baiti, Philippe G LeFloch, and Benedetto Piccoli. "Uniqueness of classical and nonclassical solutions for nonlinear hyperbolic systems". In: *Journal of Differential Equations* 172.1 (2001), pp. 59–82.

[16] Alexandre Bayen et al. *Control problems for conservation laws with traffic applications: modeling, analysis, and numerical methods*. Springer Nature, 2022.

[17] W Beaty. "Traffic "Experiments" and a Cure for Waves & Jams". In: *Available World Wide Wed: http://amasci. com/amateur/traffic/trafexp. html (Accessed 15 October 2006)* (1998).

[18] Logan E Beaver et al. "Demonstration of a Time-Efficient Mobility System Using a Scaled Smart City". In: *Vehicle System Dynamics* 58.5 (2020), pp. 787–804.

[19] Francois Belletti et al. "Expert level control of ramp metering based on multi-task deep reinforcement learning". In: *IEEE Transactions on Intelligent Transportation Systems* (2017).

[20] Richard Bellman. "The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515.

[21] Rahul Bhadani and Jonathan Sprinkle. "Prototyping Vehicle Control Applications Using the CAT Vehicle Simulator". In: *arXiv preprint arXiv:2301.04574* (2023).

[22] Rahul Bhadani et al. "Approaches for Synthesis and Deployment of Controller Models on Automated Vehicles for Car-following in Mixed Autonomy". In: *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*. 2023, pp. 158–163.

[23] *Dissipation of Emergent Traffic Waves in Stop-and-Go Traffic Using a Supervisory Controller*. Vol. 57. Fontainbleau, Miami Beach, USA: IEEE, 2018, pp. 3628–3633. DOI: 10.1109/CDC.2018.8619700. URL: https://ieeexplore.ieee.org/document/8619700.

[24] Rahul Bhadani et al. "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)*. IEEE. 2022, pp. 14–23. URL: https://ieeexplore.ieee.org/abstract/document/9805366.

[25] Alberto Bressan and Benedetto Piccoli. *Introduction to the mathematical theory of control*. Vol. 1. American institute of mathematical sciences Springfield, 2007.

[26] Greg Brockman et al. "OpenAI gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[27] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The 2005 DARPA grand challenge: the great robot race*. Vol. 36. springer, 2007.

[28] Matt Bunting et al. "Data from the Development Evolution of a Vehicle for Custom Control". In: *2022 2nd Workshop on Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities Workshop (DI-CPS)*. IEEE. 2022, pp. 40–46.

[29] Matthew Bunting, Rahul Bhadani, and Jonathan 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. URL: https://dl.acm.org/doi/abs/10.1145/3459609.3460529.

[30] C. Wu, A. M .Bayen, and A. Mehta. "Stabilizing Traffic with Autonomous Vehicles". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 6012–6018. DOI: 10.1109/ICRA.2018.8460567.

[31] C. Wu et al. "Flow: Architecture and benchmarking for reinforcement learning in traffic control". In: *arXiv preprint arXiv:1710.05465* (2017), p. 10.

[32] Eduardo Casas and Mariano Mateos. "Optimal control of partial differential equations". In: *Computational Mathematics, Numerical Analysis and Applications: Lecture Notes of the XVII'Jacques-Louis Lions' Spanish-French School* (2017), pp. 3–59.

[33] Anthony R Cassandra. "A survey of POMDP applications". In: *Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes*. Vol. 1724. 1998.

[34] Fang-Chieh Chou et al. "Reachability analysis for followerstopper: Safety analysis and experimental results". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 8607–8613.

[35] Paul Christiano et al. "Transfer from simulation to real world through learning deep inverse dynamics model". In: *arXiv preprint arXiv:1610.03518* (2016).

[36] *CIRCLES Energy Models web page*. https://github.com/CIRCLES-consortium/CIRCLES-ENERGY-MODELS. 2023.

[37] Jiaxun Cui et al. "Scalable Multiagent Driving Policies For Reducing Traffic Congestion". In: *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Virtual, May 2021.

[38] Shumo Cui et al. "Stabilizing traffic flow via a single autonomous vehicle: Possibilities and limitations". In: *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE. 2017, pp. 1336–1341.

[39] US DOT. "National transportation statistics". In: *Bureau of Transportation Statistics, Washington, DC* (2016).

[40] Yan Duan et al. "Benchmarking deep reinforcement learning for continuous control". In: *International Conference on Machine Learning*. 2016, pp. 1329–1338.

[41] Leslie C Edie. "Car-following and steady-state theory for noncongested traffic". In: *Operations research* 9.1 (1961), pp. 66–76.

[42] Safwan Elmadani et al. "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. URL: https://dl.acm.org/doi/abs/10.1145/3459609.3460531.

[43] United States Environmental Protection Agency (US EPA). URL: https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules.

[44] Darrell Etherington. *Over 1,400 self-driving vehicles are now in testing by 80+ companies across the US*. June 2019. URL: https://techcrunch.com/2019/06/11/over-1400-self-driving-vehicles-are-now-in-testing-by-80-companies-across-the-u-s/.

[45] Zhe Fu et al. *Cooperative Driving for Speed Harmonization in Mixed-Traffic Environments*. 2023. arXiv: 2302.07453 [eess.SY].

[46] Zhe Fu et al. "Cooperative Driving for Speed Harmonization in Mixed-Traffic Environments". In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2023, pp. 1–8.

[47] Mauro Garavello, Ke Han, Benedetto Piccoli, et al. *Models for vehicular traffic on networks*. Vol. 9. American Institute of Mathematical Sciences (AIMS), Springfield, MO, 2016.

[48] Alain Girault. "A hybrid controller for autonomous vehicles driving on automated highways". In: *Transportation Research Part C: Emerging Technologies* 12.6 (2004), pp. 421–452. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2004.07.008. URL: https://www.sciencedirect.com/science/article/pii/S0968090X04000208.

[49] Derek Gloudemans and Daniel B Work. "Vehicle Tracking with Crop-based Detection". In: *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2021, pp. 312–319.

[50] Derek Gloudemans et al. "I-24 MOTION: An instrument for freeway traffic science". In: *Transportation Research Part C: Emerging Technologies* 155 (2023), p. 104311.

[51] Derek Gloudemans et al. "Interstate-24 motion: Closing the loop on smart mobility". In: *2020 IEEE Workshop on Design Automation for CPS and IoT (DESTION)*. IEEE. 2020, pp. 49–55.

[52] Derek Gloudemans et al. "So you think you can track?" In: *arXiv preprint arXiv:2309.07268* (2023).

[53] Derek Gloudemans et al. "The Interstate-24 3D Dataset: a new benchmark for 3D multi-camera vehicle tracking". In: *arXiv preprint arXiv:2308.14833* (2023).

[54] François Golse. "On the dynamics of large particle systems in the mean field limit". In: *Macroscopic and large scale phenomena: coarse graining, mean field limits and ergodicity* (2016), pp. 1–144.

[55] Ivo Grondman et al. "A survey of actor-critic reinforcement learning: Standard and natural policy gradients". In: *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)* 42.6 (2012), pp. 1291–1307.

[56] Shixiang Gu et al. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 3389–3396.

[57] Jacopo Guanetti, Yeojun Kim, and Francesco Borrelli. "Control of connected and automated vehicles: State of the art and future challenges". In: *Annual Reviews in Control* (2018).

[58] George Gunter et al. "Are commercially implemented adaptive cruise control systems string stable?" In: *IEEE Transactions on Intelligent Transportation Systems* 22.11 (2020), pp. 6992–7003.

[59] Amaury Hayat et al. "A Holistic Approach to the Energy-Efficient Smoothing of Traffic via Autonomous Vehicles". In: *Intelligent Control and Smart Energy Management: Renewable Resources and Transportation*. Ed. by Maude Josée Blondin et al. Cham: Springer International Publishing, 2022, pp. 285–316.

[60] Amaury Hayat et al. "Traffic smoothing using explicit local controllers: Dissipating Stop-and-Go Waves with a Single Automated Vehicle in Dense Traffic: Experimental Evidence". In: *IEEE Control Systems Magazine* (2024).

[61] *Highlights of the Automotive Trends Report — US EPA — epa.gov*. `https://www.epa.gov/automotive-trends/highlights-automotive-trends-report`. [Accessed 04-Mar-2023].

[62] Ashley Hill et al. *Stable Baselines*. `https://github.com/hill-a/stable-baselines`. 2018.

[63] Petros A Ioannou and Cheng-Chih Chien. "Autonomous intelligent cruise control". In: *IEEE Transactions on Vehicular technology* 42.4 (1993), pp. 657–672.

[64] J. M. Sprinkle et al. "CIRCLES: Congestion Impacts Reduction via CAV-in-the-loop Lagrangian Energy Smoothing". In: Presented at the 2022 Vehicle Technologies Office Annual Merit Review, Washington, DC, 2022.

[65] Kathy Jang et al. "Reinforcement Learning Based Oscillation Dampening: Scaling up Single-Agent RL algorithms to a 100 AV highway field operational test". In: *IEEE Control Systems Magazine* (2024). © 2024 IEEE. Reprinted, with permission, from Control Systems Magazine.

[66] Kathy Jang et al. "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.

[67] Kathy Jang et al. "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles". In: *2019 International Conference on Cyber-Physical Systems*. Montreal, CA, 2019. ISBN: 9781450362856.

[68] Arne Kesting, Martin Treiber, and Dirk Helbing. "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 368.1928 (2010), pp. 4585–4605.

[69] Nour Khoudari et al. "A Systematic Model Reduction Pipeline from Detailed Vehicle Energy Dynamics to Simple Models with Desirable Physics-like Properties". In: *IEEE Control Systems Magazine* (2024).

[70] Seoungbum Kim and Benjamin Coifman. "Comparing INRIX speed data against concurrent loop detector stations over several months". In: *Transportation Research Part C: Emerging Technologies* 49 (2014), pp. 59–72. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2014.10.002. URL: https://www.sciencedirect.com/science/article/pii/S0968090X14002940.

[71] Daniel Krajzewicz et al. "Recent Development and Applications of SUMO - Simulation of Urban MObility". In: *International Journal On Advances in Systems and Measurements* 5.3&4 (Dec. 2012), pp. 128–138.

[72] Abdul Rahman Kreidieh, Yashar Farid, and Kentaro Oguchi. "Non-local Evasive Overtaking of Downstream Incidents in Distributed Behavior Planning of Connected Vehicles". In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2022, pp. 1453–1459.

[73] Abdul Rahman Kreidieh, Cathy Wu, and Alexandre M Bayen. "Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning". In: *2018 21st international conference on intelligent transportation systems (itsc)*. IEEE. 2018, pp. 1475–1480.

[74] Abdul Rahman Kreidieh et al. "Inter-level cooperation in hierarchical reinforcement learning". In: *arXiv preprint arXiv:1912.02368* (2019).

[75] Scott Kuindersma et al. "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot". In: *Autonomous Robots* 40.3 (2016), pp. 429–455.

[76] Argonne National Laboratory. *Autonomie*. https://www.anl.gov/taps/autonomie-vehicle-system-simulation-tool. 2023.

[77] Jonathan W. Lee et al. "Integrated Framework of Vehicle Dynamics, Instabilities, Energy Models, and Sparse Flow Smoothing Controllers". In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. Nashville, TN, USA, 2021, pp. 41–47. ISBN: 9781450384452. DOI: 10.1145/3459609.3460530.

[78] Jonathan W. Lee et al. "Integrated Framework of Vehicle Dynamics, Instabilities, Energy Models, and Sparse Flow Smoothing Controllers". In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. DI-CPS'21. Nashville, TN, USA: Association for Computing Machinery, 2021, pp. 41–47. ISBN: 9781450384452. DOI: 10.1145/3459609.3460530. URL: https://doi.org/10.1145/3459609.3460530.

[79] Jonathan W. Lee et al. "Traffic Smoothing via Connected & Automated Vehicles: A Modular, Hierarchical Control Design Deployed in a 100-CAV Flow Smoothing Experiment". In: *IEEE Control Systems Magazine* (2024). © 2024 IEEE. Reprinted, with permission, from Control Systems Magazine.

[80] Joyoung Lee and Byungkyu Park. "Development and Evaluation of a Cooperative Vehicle Intersection Control Algorithm Under the Connected Vehicles Environment". In: *IEEE Transactions on Intelligent Transportation Systems* 13.1 (2012), pp. 81–90. DOI: 10.1109/TITS.2011.2178836.

[81] *Level 2 autonomous driving Q4 2020 and full year 2020*. URL: https://www.canalys.com/newsroom/canalys-autonomous-driving-starts-to-hit-mainstream-as-35-million-new-cars-had-level-2-features-in-q4-2020.

[82] Sergey Levine and Pieter Abbeel. "Learning neural network policies with guided policy search under unknown dynamics". In: *Advances in Neural Information Processing Systems*. 2014, pp. 1071–1079.

[83] Li Li, Yisheng Lv, and Fei-Yue Wang. "Traffic signal timing via deep reinforcement learning". In: *IEEE/CAA Journal of Automatica Sinica* 3.3 (2016), pp. 247–254.

[84] Nan I Li, Chaozhe R He, and Gábor Orosz. "Sequential parametric optimization for connected cruise control with application to fuel economy optimization". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 227–232.

[85] Xiaopeng Li et al. "Stop-and-go traffic analysis: Theoretical properties, environmental impacts and oscillation mitigation". In: *Transportation Research Part B: Methodological* 70 (2014), pp. 319–339.

[86] Eric Liang et al. "Ray RLLib: A Composable and Scalable Reinforcement Learning Library". In: *arXiv preprint arXiv:1712.09381* (2017).

[87] Nathan Lichtlé et al. "Deploying traffic smoothing cruise controllers learned from trajectory data". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2884–2890. URL: https://ieeexplore.ieee.org/abstract/document/9811912.

[88] Nathan Lichtlé et al. "Traffic Smoothing Controllers for Autonomous Vehicles Using Deep Reinforcement Learning and Real-World Trajectory Data". In: *IEEE Transactions on Intelligent Transportation Systems* todo.todo (2023), todo.

[89] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[90] Xiao-Yun Lu and Steven E Shladover. "Review of variable speed limits and advisories: Theory, algorithms, and practice". In: *Transportation research record* 2423.1 (2014), pp. 15–23.

[91] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[92] A A Malikopoulos, Christos G Cassandras, and Y Zhang. "A Decentralized Energy-Optimal Control Framework for Connected Automated Vehicles at Signal-Free Intersections". In: *arXiv:1602.03786 - (provisionally accepted)* (2017).

[93] A. A. Malikopoulos et al. "Optimal Control for Speed Harmonization of Automated Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* (2018).

[94] Amir Reza Mamdoohi et al. "Comparative analysis of safety performance indicators based on inductive loop detector data". In: *PROMET-Traffic&Transportation* 26.2 (2014), pp. 139–149.

[95] Horia Mania, Aurelia Guy, and Benjamin Recht. "Simple random search provides a competitive approach to reinforcement learning". In: *arXiv preprint arXiv:1803.07055* (2018).

[96] Matthias Mueller et al. "Driving Policy Transfer via Modularity and Abstraction". In: *Conference on Robot Learning*. IEEE, 2018, pp. 1–15.

[97] Paul Ngo and Jonathan Sprinkle. "Lightweight LSTM for CAN Signal Decoding". In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. 2021, pp. 27–31.

[98] Paul Ngo, Jonathan Sprinkle, and Rahul Bhadani. "CANClassify: Automated Decoding and Labeling of CAN Bus Signals". In: *Berkeley Master's Thesis* 1.1 (2022), pp. 1–8.

[99] Jie Ni, Jingwen Han, and Fei Dong. "Multivehicle cooperative lane change control strategy for intelligent connected vehicle". In: *Journal of Advanced Transportation* 2020 (2020).

[100] Matthew Nice et al. "Enabling Mixed Autonomy Traffic Control". In: *arXiv preprint arXiv:2310.18776* (2023).

[101] Matthew Nice et al. "Middleware for a Heterogeneous CAV Fleet". In: *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*. 2023, pp. 86–91. URL: https://dl.acm.org/doi/abs/10.1145/3459609.3460531.

[102] Matthew Nice et al. "Parameter Estimation for Decoding Sensor Signals". In: *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*. 2023, pp. 253–255.

[103] Matthew Nice et al. *The I-24 Trajectory Dataset*. Version 1.1.1. Removing (.) files and removing bug in MileMarker data processing. Zenodo, Sept. 2021. DOI: 10.5281/zenodo.6456348. URL: https://doi.org/10.5281/zenodo.6456348.

[104] Matthew Nice et al. *The I-24 Trajectory Dataset*. Version 1.1.1. Zenodo, Sept. 2021. DOI: 10.5281/zenodo.6456348. URL: https://doi.org/10.5281/zenodo.6456348.

[105] Matthew Nice et al. *The I-24 Trajectory Dataset*. 2021.

[106] Ioannis A Ntousakis, Ioannis K Nikolos, and Markos Papageorgiou. "Optimal vehicle trajectory planning in the context of cooperative merging on highways". In: *Transportation Research Part C: Emerging Technologies* 71 (2016), pp. 464–488.

[107] *ONNX: Open Neural Network Exchange*. https://onnx.ai/. Accessed on: 2023. 2017.

[108] Gábor Orosz. "Connected cruise control: modelling, delay effects, and nonlinear behaviour". In: *Vehicle System Dynamics* 54.8 (2016), pp. 1147–1176.

[109] Xue Bin Peng et al. "Sim-to-real transfer of robotic control with dynamics randomization". In: *arXiv preprint arXiv:1710.06537* (2017).

[110] Lerrel Pinto et al. "Robust adversarial reinforcement learning". In: *arXiv preprint arXiv:1703.02702* (2017).

[111] Xiaobo Qu et al. "Jointly dampening traffic oscillations and improving energy consumption with electric, connected and automated vehicles: a reinforcement learning based approach". In: *Applied Energy* 257 (2020), p. 114030.

[112] R. E. Stern et al. "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments". In: *Transportation Research Part C* 89 (Apr. 2018). DOI: 10.1016/j.trc.2018.02.005.

[113] Hesham Rakha and Raj Kishore Kamalanathsharma. "Eco-driving at signalized intersections using V2I communication". In: *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*. IEEE. 2011, pp. 341–346.

[114] Alex Richardson et al. "Analysis of a Runtime Data Sharing Architecture over LTE for a Heterogeneous CAV Fleet". In: *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*. 2023, pp. 164–169.

[115] Jackeline Rios-Torres and Andreas A Malikopoulos. "Automated and Cooperative Vehicle Merging at Highway On-Ramps". In: *IEEE Transactions on Intelligent Transportation Systems* 18.4 (2017), pp. 780–789.

[116] Jackeline Rios-Torres and Andreas A Malikopoulos. "Impact of Partial Penetrations of Connected and Automated Vehicles on Fuel Consumption and Traffic Flow". In: *IEEE Transactions on Intelligent Vehicles* 3.4 (2018), pp. 453–462.

[117] Jackeline Rios-Torres and Andreas A. Malikopoulos. "A Survey on Coordination of Connected and Automated Vehicles at Intersections and Merging at Highway On-Ramps". In: *IEEE Transactions on Intelligent Transportation Systems* 18.5 (2017), pp. 1066–1077.

[118] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[119] Andrei A Rusu et al. "Sim-to-real robot learning from pixels with progressive nets". In: *arXiv preprint arXiv:1610.04286* (2016).

[120] Tim Salimans et al. "Evolution strategies as a scalable alternative to reinforcement learning". In: *arXiv preprint arXiv:1703.03864* (2017).

[121] B. Schrank et al. *2015 Urban Mobility Scorecard*. Tech. rep. Texas A& M Transportation Institute, 2015.

[122] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[123] John Schulman et al. "Trust Region Policy Optimization". In: *ICML*. 2015, pp. 1889–1897.

[124] John Schulman et al. "Trust region policy optimization". In: *International Conference on Machine Learning*. 2015, pp. 1889–1897.

[125] Steven E Shladover. "Connected and Automated Vehicle Systems: Introduction and Overview". In: *Journal of Intelligent Transportation Systems* just-accepted (2017), pp. 00–00.

[126] Steven E Shladover et al. "Cooperative adaptive cruise control: Definitions and operating concepts". In: *Transportation Research Record* 2489.1 (2015), pp. 145–152.

[127] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

[128] Adam Stager et al. "A Scaled Smart City for Experimental Validation of Connected and Automated Vehicles". In: *IFAC-PapersOnLine* 51.9 (2018), pp. 130–135. ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.07.022.

[129] R. Stern et al. "Stabilizing Traffic with a Single Autonomous Vehicle". In: *7th International Conference on Cyber-Physical Systems (ICCPS)*. ACM/IEEE. 2016.

[130] Raphael E Stern et al. "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments". In: *arXiv preprint arXiv:1705.01693* (2017).

[131] William B Stevens. "The automated highway system program: A progress report". In: *IFAC Proceedings Volumes* 29.1 (1996), pp. 8180–8188.

[132] Yuki Sugiyama et al. "Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam". In: *New journal of physics* 10.3 (2008), p. 033001.

[133] Kang Sun, Xiangmo Zhao, and Xia Wu. "A cooperative lane change model for connected and autonomous vehicles on two lanes highway by considering the traffic efficiency on both lanes". In: *Transportation research interdisciplinary perspectives* 9 (2021), p. 100310.

[134] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press Cambridge, 2018.

[135] Richard S Sutton et al. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems* 12 (1999).

[136] D Swaroop and J Karl Hedrick. "String stability of interconnected systems". In: *IEEE transactions on automatic control* 41.3 (1996), pp. 349–357.

[137] DVAHG Swaroop et al. "A comparision of spacing and headway control laws for automatically controlled vehicles1". In: *Vehicle system dynamics* 23.1 (1994), pp. 597–625.

[138] Han-Shue Tan, Rajesh Rajamani, and Wei-Bin Zhang. "Demonstration of an automated highway platoon system". In: *Proceedings of the 1998 American control conference. ACC (IEEE Cat. No. 98CH36207)*. Vol. 3. IEEE. 1998, pp. 1823–1827.

[139] Sebastian Thrun et al. "Stanley: The robot that won the DARPA Grand Challenge". In: *Journal of field Robotics* 23.9 (2006), pp. 661–692.

[140] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE. 2017, pp. 23–30.

[141] Behrad Toghi et al. "Cooperative autonomous vehicles that sympathize with human drivers". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 4517–4524.

[142] Lloyd N Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[143] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. "Congested traffic states in empirical observations and microscopic simulations". In: *Physical review E* 62.2 (2000), p. 1805.

[144] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. "Microscopic simulation of congested traffic". In: *Traffic and Granular Flow'99: Social, Traffic, and Granular Dynamics*. Springer. 2000, pp. 365–376.

[145] Martin Treiber and Arne Kesting. "The intelligent driver model with stochasticity-new insights into traffic flow oscillations". In: *Transportation research procedia* 23 (2017), pp. 174–187.

[146] *U.S. energy facts explained - consumption and production - U.S. Energy Information Administration (EIA) — eia.gov*. https://www.eia.gov/energyexplained/us-energy-facts/. [Accessed 04-Mar-2023].

[147] US Department of Transportation Federal Highway Administration. *Next Generation Simulation (NGSIM)*. Website. http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.

[148] *Use of energy for transportation - U.S. Energy Information Administration (EIA) — eia.gov*. https://www.eia.gov/energyexplained/use-of-energy/transportation.php. [Accessed 04-Mar-2023].

[149] Skylar Knickerbocker Vesal Ahsani Mostafa Amin-Naseri and Anuj Sharma. "Quantitative analysis of probe data characteristics: Coverage, speed bias and congestion detection precision". In: *Journal of Intelligent Transportation Systems* 23.2 (2019), pp. 103–119. DOI: 10.1080/15472450.2018.1502667. eprint: https://doi.org/10.1080/15472450.2018.1502667. URL: https://doi.org/10.1080/15472450.2018.1502667.

[150] Eugene Vinitsky et al. "Benchmarks for reinforcement learning in mixed-autonomy traffic". In: *Conference on robot learning*. PMLR. 2018, pp. 399–409. URL: https://proceedings.mlr.press/v87/vinitsky18a.html.

[151] Eugene Vinitsky et al. "Benchmarks for reinforcement learning in mixed-autonomy traffic". In: *Proceedings of The 2nd Conference on Robot Learning*. Ed. by Aude Billard et al. Vol. 87. Proceedings of Machine Learning Research. PMLR, 29–31 Oct 2018, pp. 399–409. URL: https://proceedings.mlr.press/v87/vinitsky18a.html.

[152] Eugene Vinitsky et al. "Lagrangian control through deep-rl: Applications to bottleneck decongestion". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 759–765.

[153] Eugene Vinitsky et al. "Optimizing mixed autonomy traffic flow with decentralized autonomous vehicles and multi-agent rl". In: *arXiv preprint arXiv:2011.00120* (2020).

[154] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), pp. 350–354.

[155] Zia Wadud, Don MacKenzie, and Paul Leiby. "Help or hindrance? The travel, energy and carbon impacts of highly automated vehicles". In: *Transportation Research Part A: Policy and Practice* 86 (2016), pp. 1–18.

[156] Chen Wang and Nikiforos Stamatiadis. "Surrogate safety measure for simulation-based conflict study". In: *Transportation research record* 2386.1 (2013), pp. 72–80.

[157] Han Wang et al. "Centralized AV Speed Planner: Hierarchical Framework for Lagrangian Variable Speed Limit in Mixed Autonomy Traffic". In: *IEEE Control Systems Magazine* (2024).

[158] Yanbing Wang et al. "Automatic vehicle trajectory data reconstruction at scale". In: *arXiv preprint arXiv:2212.07907* (2022).

[159] Yanbing Wang et al. "Online Min Cost Circulation for Multi-Object-Tracking on Fragments". In: *2023 IEEE International Intelligent Transportation Systems Conference (ITSC)*. To appear. IEEE. 2023.

[160] Yu Wang, Xiaopeng Li, and Handong Yao. "Review of trajectory optimisation for connected automated vehicles". In: *IET Intelligent Transport Systems* (2018).

[161] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8 (1992), pp. 279–292.

[162] Hua Wei et al. "Intellilight: A reinforcement learning approach for intelligent traffic light control". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 2496–2505.

[163] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8 (1992), pp. 229–256.

[164] R Eddie Wilson and Jonathan A Ward. "Car-following models: fifty years of linear stability analysis–a mathematical perspective". In: *Transportation Planning and Technology* 34.1 (2011), pp. 3–18.

[165] Cathy Wu et al. "Emergent behaviors in mixed-autonomy traffic". In: *Conference on Robot Learning (CoRL)*. 2017. URL: http://proceedings.mlr.press/v78/wu17a.html.

[166] Cathy Wu et al. "Flow: A modular learning framework for mixed autonomy traffic". In: *IEEE Transactions on Robotics* 38.2 (July 2021), pp. 1270–1286. ISSN: 1941-0468. DOI: 10.1109/TRO.2021.3087314.

[167] Cathy Wu et al. "Flow: Architecture and benchmarking for reinforcement learning in traffic control". In: *arXiv preprint arXiv:1710.05465* 10 (2017). URL: https://www.researchgate.net/profile/Abdul-Rahman-Kreidieh/publication/320441979_Flow_Architecture_and_Benchmarking_for_Reinforcement_Learning_in_Traffic_Control/links/5cf6c7f4a6fdcc847506322a/Flow-Architecture-and-Benchmarking-for-Reinforcement-Learning-in-Traffic-Control.pdf.

[168] Cathy Wu et al. "Flow: Architecture and benchmarking for reinforcement learning in traffic control". In: *arXiv preprint arXiv:1710.05465* 10 (2017).

[169] Fangyu Wu and Alexandre M. Bayen. "A Hierarchical MPC Approach to Car-Following via Linearly Constrained Quadratic Programming". In: *IEEE Control Systems Letters* 7 (2023), pp. 532–537. DOI: 10.1109/LCSYS.2022.3201162.

[170] Fangyu Wu et al. "Tracking vehicle trajectories and fuel rates in phantom traffic jams: Methodology and data". In: *Transportation Research Part C: Emerging Technologies* 99 (2019), pp. 82–109.

[171] Lingyun Xiao and Feng Gao. "A comprehensive review of the development of adaptive cruise control systems". In: *Vehicle system dynamics* 48.10 (2010), pp. 1167–1192.

[172] Zhuo Xu, Chen Tang, and Masayoshi Tomizuka. "Zero-shot Deep Reinforcement Learning Driving Policy Transfer for Autonomous Vehicles based on Robust Control". In: *International Conference on Intelligent Transportation Systems*. IEEE, 2018, pp. 2865–2871.

[173] Y. Sugiyama et al. "Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam". In: *New Journal of Physics* 10.3 (Mar. 2008), p. 033001. DOI: 10.1088/1367-2630/10/3/033001. URL: https://dx.doi.org/10.1088/1367-2630/10/3/033001.

[174] Kösaku Yosida. *Functional analysis*. Springer Science & Business Media, 2012.

[175] Gergely Zachár. "Visualization of Large-Scale Trajectory Datasets". In: CPS-IoT Week '23. San Antonio, TX, USA: Association for Computing Machinery, 2023, pp. 152–157. ISBN: 9798400700491. DOI: 10.1145/3576914.3587710. URL: https://doi.org/10.1145/3576914.3587710.

[176] Yulin Zhang et al. *Learning a Robust Multiagent Driving Policy for Traffic Congestion Reduction*. 2023. arXiv: 2112.03759 [cs.AI].

[177] L. Zhao and A. A. Malikopoulos. "Enhanced Mobility with Connectivity and Automation: A Review of Shared Autonomous Vehicle Systems". In: *IEEE Intelligent Transportation Systems Magazine* (2020 (forthcoming)).

[178] Liuhui Zhao and Andreas A Malikopoulos. "Decentralized optimal control of connected and automated vehicles in a corridor". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 1252–1257.

[179] Zuduo Zheng et al. "Freeway traffic oscillations: microscopic analysis of formations and propagations using wavelet transform". In: *Procedia-Social and Behavioral Sciences* 17 (2011), pp. 702–716.