A RESEARCH TESTBED FOR EXPERIMENTAL CONNECTIVITY AND AUTOMATION IN CARS

By

Matthew Walter Nice

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Civil Engineering

May 10, 2024

Nashville, Tennessee

Approved

Mark D. Abkowitz, Ph.D.

Hiba Baroud, Ph.D.

Jonathan M. Sprinkle, Ph.D.

Daniel B. Work (Chair), Ph.D.

To those with the audacity to pursue what's true and worthy.

ACKNOWLEDGMENTS

> What is the point of worrying oneself too much about what one could or could not have done to control the course one's life took? Surely it is enough that the likes of you and I at least try to make our small contribution count for something true and worthy. And if some of us are prepared to sacrifice much in life in order to pursue such aspirations, surely that in itself, whatever the outcome, cause for pride and contentment.
>
> Kazuo Ishiguro, The Remains of the Day

Advisor(s): I'm very grateful to Dan Work for taking a chance on me as an Master's student in the Cyber-Physical Systems program at Vanderbilt, and supporting my development as a researcher. You've shown me what research leadership looks like. You never showed a moment of uncertainty of success, even when your optimism was surely insane. Many thanks to Jonathan Sprinkle, my second PhD advisor. We started working together at the beginning of 2020, when I was welcomed into the CIRCLES Hardware team as a total newbie. I recall a searing memory from the first days of learning ROS and bash, where I got help in figuring out how to tell if my board was a Raspberry Pi 3 or 4. Turns out you just need to read what is printed on it. Thank you for giving me the opportunities to grow as an engineer, researcher, and leader.

CIRCLES Project Team: Thanks to the CIRCLES PIs, for putting together a large collaborative project that motivated much of the work in this thesis. Some data used in this thesis has been published before its release because thanks to the PIs. Thanks as well to the entire CIRCLES team [https://circles-consortium.github.io/], for being open to collaborative research.

Research Community: To my lab-mates, you went above and beyond the call of duty in helping my research. Thanks to you, who manufactured cables, drove in experimental control vehicles, made many 5am wakeup calls, eviscerated my practice presentations, and more. Special thanks to George, who did not make a habit of assuming the hardware was breaking the car, even if it was my fault. And for all the research done at Villager Tavern and similar venues. Another special call out for Matt B. We achieved some things I thought might be impossible, and made it through some incredibly stressful moments; it has always been a pleasure to work together.

Family and Friends: Thank you to Winona for your love, support, and occasional skeptical glances. Thank you to my friends and family for lending an ear when I needed it, and reminding me about what is more important than submitting more research papers; especially to my parents Amy and Tom Nice, my sisters Maggie and Clara, and my brothers-in-law Dave, and Nadav.

Funding: This material has been supported in part by: the National Science Foundation under Grant Nos. CNS-2135579, 2111688; 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; and the Dwight D. Eisenhower Fellowship program under Grant Nos. 693JJ32345023, 693JJ32245044, and 693JJ32445061. The views expressed herein do not necessarily represent the views of the U.S. DOE, NSF, FHWA or the U.S. Government. I would like to thank the Tennessee Department of Transportation, Nissan, Toyota, and GM, who directly or indirectly supported this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 | Introduction

Perhaps I can find new ways to motivate them.

Darth Vader, Return of the Jedi (1983)

Cyber-physical systems (CPS) are engineered systems in which the computational components and physical components of the system are interdependent and deeply linked. Discovery and development of CPS technologies are changing the way people interact with engineered systems. To demonstrate this point, consider the Global Positioning System (GPS) which revolutionized personal mobility. Its research origins stem from the US Department of Defense in 1973 for military use. By 2000 limitations on civilian consumption of GPS service were lifted. Immediately, personal GPS products were shipped for in-car navigation systems. Soon after, GPS receivers were embedded into mobile phones with the introduction of the Nokia N95 and Apple iPhone for maps and turn-by-turn navigation. With the benefit of hindsight we can see that GPS has become a cornerstone technology of our daily life. Satellite-based navigation has gone from a radical space technology, to a matter of uninteresting availability; it is the typical way that people navigate through roadways now. The combination of the satellite infrastructure, the integration of receiver modules, and powerful maps applications has significantly changed mobility in the 21st century.

Modern transportation systems are increasingly cyber-physical systems. Major investments from the public and private sectors are infusing embedded computing into road-side infrastructure, signalized intersections, roadways, and cars. These investments are motivated by the pronounced potential of connectivity, networking, automation, and data which can be used to inform better operational decisions in real-time, and increase safety and efficiency of mobility. Smart infrastructure and intelligent cars are coming online, but we still await the breakthrough applications which could radically change the transportation system. There are massive opportunities to improve transportation safety and efficiency in US freeways in particular. More than 95% of all transportation-related fatalities in the US occur on freeways – over 40,000 fatalities in 2021[1]. At that rate, more than 4 people die on American freeways every hour in a system that people use daily for commuting and commerce. In light of the fact that CPS technologies can change the way we interact with engineered systems, a motivating question arises: *what technologies can change the way we interact with the freeway systems to improve safety?* In context of the emerging climate change crisis, finding effective ways to reduce greenhouse gas (GHG) emissions are as important as ever. There are also great geo-political ramifications to energy use, and reduced energy use supports national goals in increased energy independence. Transportation accounts for the largest share of GHG emissions in the US[2], with light-duty vehicles taking the lion's share of these emissions. This raises more motivating questions: *what CPS technologies can change the way we interact with freeway systems to improve efficiency?*, and the more general and succinct: *can we change freeways for the better with CPS technology?*

If we could perform *traffic control* on freeway systems, we could diminish these societal-scale problems. Controlling the traffic system would allow us to adjust the system-level characteristics which lead to increased safety risk and energy use. To effectively control traffic on freeways, we need to close the control loop so that evolving and complex changes to the state of the traffic system can inform the actions made by the cars – which in turn change the traffic state, and the loop continues. Each passenger car, truck, or freight vehicle is an individual system with agency that contributes to the gestalt of the greater freeway traffic system. This German word, *gestalt*, refers to something that is made of many parts and yet is somehow more than or

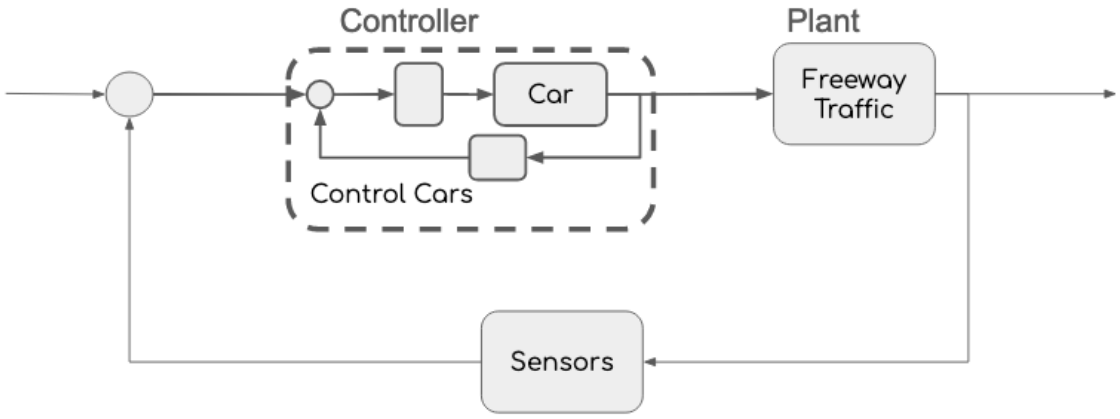


Figure 1.1: **A System of Systems:** To control the freeway traffic system, we need to consider problems of the macroscopic level such as properly measuring the system's state (with ITS fixed infrastructure or mobile sensing networks), and problems of the microscopic level such as transforming individual cars into experimental controllers for actuating the traffic control.

different from the combination of its parts. The base units of actuating control in the freeway system are cars, which in free flow conditions can be similar to the combination of independent cars. In congestion, however, cars are highly interdependent and their complex interactions cause consistent emergent patterns like *phantom traffic jams*. These jams cause frequent traffic waves on freeways, resulting in higher speed variations in the traffic system. This speed variability contributes to increased safety risks and increased energy use on congested highways. These traffic waves are in principle avoidable, however in today's freeway congestion perpetual traffic waves can cost of an extra 40% in energy system-wide. To control the freeway system (a system of systems) in new ways, and curb the safety and energy problems, there is a need to invent new ways for traffic control (a controller of controllers) to close the control loop.

The London Symphony Orchestra famously was recorded for the film scores to 'Star Wars' and the 'Indiana Jones' series, composed and conducted by John Williams. The conductor's critical role, even for the virtuosic London Symphony Orchestra, is to 'play the orchestra' to shape the sound of the ensemble. When Williams flicks his baton he adjusts the pace of play, and when he boldly gestures to different sections of instrumentalists they will finely adjust their playing styles. To control traffic we need effective control mechanisms to shape the emergent behavior of the traffic system. Traditional methods of freeway traffic control like ramp metering, variable speed limits, and tolling, have not been able to broadly achieve substantial progress in reducing fatalities[1] or dampening traffic waves; there is no effective 'baton flick' or 'bold gesture' to diminish these problems. Thus, instead of a focus on infrastructure-driven control, this dissertation focuses on a different paradigm to conduct traffic control called *mobile traffic control*. The key mechanism for mobile traffic control is to actuate the traffic system through an ensemble of cars that partially compose the traffic itself, as they move through the transportation system. Mobile traffic control finds its strength through collectivizing the power of an ensemble of vehicles at-scale. To achieve this vision however, there are several problems to face.

1.1 Problem Statements

Traffic control relying on human drivers suffers from low compliance. A straightforward approach to achieving mobile traffic control is to adjust how human drivers control their cars. Using a human-in-the-loop, i.e. augmenting normal driving with more information to the driver, can be effective at least at small scales to improve driving outcomes [3], [4], but this solution has not gained traction. This is not for a lack of information. Information can be relayed to the driver with high fidelity, such as with a dashboard 'recommended speed display', or to all drivers more coarsely with variable speed limits (VSL). The safety benefits of VSL are generally positive [5], [6], but these systems are known to rely heavily on compliance [7]–[10], and at the same time suffer from low compliance from North American drivers [8], [11]. An alternative to changing human drivers is automated driving.

Experimental connected automated vehicles (CAVs) are inaccessible. The core promise of connectivity and automation is that a car could have access to much more information and act on it in ways that a person at the wheel could not. A great strength of automated systems is their reliable and direct connection to the software and control systems that drive their behavior. In the context of freeway transit, there are promising ideas like ultra-high-density CAV platoons which could have revolutionary implications for traffic engineering in terms of safety, throughput, and energy efficiency. The vast majority of these ideas are limited to study in simulation because of the inaccessibility and great expense of experimental CAVs. Technical challenges in developing CPS technologies with CAVs aside, the capital expenses of instrumentation provide a significant barrier to intrepid researchers interested in system-level effects that CAVs could have.

Researchers able to field test will opt to test on closed courses to simplify the safety constraint that comes with testing 'in the wild', on open roads.

Typical costs are in the hundreds of thousands (USD) just to instrument a single commercially available vehicle into an experimental automated vehicle. To field mobile traffic control in the wild, the capital costs go up a couple orders of magnitude. Without access to testing in real world conditions, development and discovery of CAV technologies is hampered.

New technologies do not always work as intended. Testing in real environments creates opportunities to make technological leaps; in the absence of real world testbeds, the risks of unintended consequences increase. Even with accessibility of CAVs, it is important to pay attention to potential negative knock-on effects of their large-scale adoption. There is a repeated habit of creating transportation CPS technologies with little regard to the consequences of their adoption at-scale. The introduction of on-demand ride sharing made the use of taxis much more convenient, but a byproduct of this is direct contribution to increasing congestion in high-demand areas[12]; personal navigation applications in smart phones changed how people consider travelling, and has encouraged undesirable driving maneuvers to such a degree that it has triggered changes in municipal traffic management [13]; and commercially available adaptive cruise control (ACC) automation has demonstrably negative effects on the emergent properties of traffic[14], [15]. The problem of unintended consequences is one reason driving the need for real-world experimental testbeds. Simulation environments that could diagnose unintended consequences fall prey to the canonical 'sim-to-real' gap. This gap between simulation and reality is very difficult to close in cyber-physical systems which involve complex dynamical systems interactions and novel computing applications. Promising ideas commonly fall flat when real world safety and complexity come into play. Testbeds combat these issues by unlocking the ability to iteratively test and observe experimental technologies in real-world environments.

No testbed exists for real-world investigation of mobile traffic control. Integrating a small proportion of automated driving to achieve mobile traffic control has been shown to be a promising approach to traffic

system-level control[16], [17]. This concept, coined 'mixed autonomy traffic control', is a regime of traffic where driving decisions are made by a mixture of automated vehicles and human-operated vehicles. The resulting complex interactions cannot be studied by isolating the automated behaviors and human behaviors, but rather must be investigated *in situ* on real roadways to account for the rich interactions between ensembles of automated cars and human drivers. Mixed autonomy traffic control can be effective with small proportions of AVs because of the interdependence of cars in traffic. Some proportion of automated vehicles can effect the emergent properties of the entire traffic flow, at least in a ring road[16] and in numerical simulations[17]. Nonetheless, mixed autonomy traffic control has never been fielded, and it is unclear how wide the gap is between simulated settings and the wild reality of the open road. Further, there are no existing testbeds with which mobile traffic control can be tested in high-complexity multi-lane highway contexts. Consequently, it is unclear whether progress on control design for real-world smoothing is being made.

1.2 Contributions

The main contribution of this dissertation is a CPS testbed for experimental connectivity and automation in cars. This testbed enables mixed autonomy traffic control applications, and is compatible with infrastructure integrations (V2I). The testbed is built from a popular commodity vehicle as its base, with low-cost computing hardware and a feature rich software stack. The testbed has an extensive set of capabilities; the design, development, and deployment of which compose this dissertation. These contributions are collected into three parts: Part 1: Establishing new pipelines for experimental control; Part 2: Scaling up for mobile traffic control; and Part 3: Expanding testbed interfaces. Part 1 focuses on establishing novel and experimental interfaces for control in single vehicles, though (a) the driver, i.e. Human CPS; and (b) the car itself, i.e. automated driving. Part 2 focuses on the challenge of scaling experimental control for an ensemble of cars to achieve traffic control. Part 3 focuses on expanding the testbed interfaces to incorporate interactions with smart infrastructure, and associated challenges.

The specific contributions to building a testbed for experimental vehicle connectivity and automation are as follows, presented in this order:

- **Flexible Experimental Vehicle Control with a Human-in-the-loop**
  - This work addresses whether a *human-in-the-loop cyber-physical system* (HCPS) can be effective in improving the longitudinal control of an individual vehicle in a traffic flow. We introduce the *CAN Coach*, which is a system that gives feedback to the human-in-the-loop using radar data (relative speed and position information to objects ahead) that is available on the *controller area network* (CAN).
  - We conclude that (1) it is possible to coach drivers to improve performance on driving tasks using CAN data, and (2) it is a true HCPS, since removing human perception from the control loop reduces performance at the given control objective.
  - *Publications:*
    - M. Nice, S. Elmadani, R. Bhadani, *et al.*, "Can coach: Vehicular control through human cyber-physical systems," in *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, 2021, pp. 132–142
- **A Pipeline for Experimental Automated Vehicle Control**
  - Autonomous vehicle-based traffic smoothing controllers are often not transferred to real-world use due to challenges in calibrating many-agent traffic simulators. We show a pipeline to sidestep

such calibration issues by collecting trajectory data and learning controllers directly from trajectory data that are then deployed zero-shot onto the highway.
  - We construct a dataset of 772.3 kilometers of recorded drives on the I-24. We then construct a simple simulator using the recorded drives as the lead vehicle in front of a simulated platoon consisting of one autonomous vehicle and five human followers. Using policy-gradient methods with an asymmetric critic to learn the controller, we show that we are able to improve average MPG by 11% in simulation on congested trajectories.
  - We deploy this controller to a mixed platoon of 4 autonomous Toyota RAV-4's and 7 human drivers in a validation experiment and demonstrate that the expected time-gap of the controller is maintained in the real world test.
  - *Publications:*
    * N. Lichtlé†, E. Vinitsky†, M. Nice†, *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
- **Enabling Mixed Autonomy Traffic Control**
  - We deploy the first large-scale team of connected automated vehicles (CAVs) for mixed autonomy traffic control.
  - We introduce a hardware and software platform to enable experimental autonomy and connectivity in commercially available SAE level 1 and level 2 automated vehicles. The platform supports experimental automated vehicle control, live mobile sensing with connected vehicles, high fidelity data collection, and scalability.
  - This platform enables an agile develop/deploy cycle for at scale cyber-physical systems research, empowering its field deployment.
  - *Publications:*
    * M. Nice, M. Bunting, A. Richardon, *et al.*, "Enabling mixed autonomy traffic control," *arXiv preprint arXiv:2310.18776*, 2023
- **SAILing CAVs: Speed-Adaptive Infrastructure-Linked Connected Automated Vehicles**
  - We develop and field deploy the first connected automated vehicle with the capability to follow publicly broadcast variable speed limits. We implement this capability with open source hardware and software that extends a stock vehicle's adaptive cruise control.
  - We demonstrate the system on a vehicle in heavy traffic on an open roadway, and compare the performance of the equipped vehicle to a human piloted vehicle driving in the same traffic.
  - *Publications:*
    * M. Nice, M. Bunting, G. Gunter, *et al.*, "Sailing cavs: Speed-adaptive infrastructure-linked connected and automated vehicles," *arXiv preprint arXiv:2310.06931*, 2023
- ***Via Media*: Fielding Connected Automated Vehicles to Follow Variable Speed Limits in Low-Compliance Regimes**
  - We introduce a new notion of safety for cooperative automated vehicle applications to avoid causing controlled vehicles to drive substantially slower than surrounding traffic. Our approach recognizes the necessity for automated vehicles to adhere with the typical driving behavior observed on the roads, even if it requires a deviation from the posted speed limit.
  - Development of a vehicular-based method for measuring prevailing traffic. Since the measurement is done on the vehicle, the vehicle can maintain safety (accurate awareness of with sur-

rounding traffic) locally, even if the vehicle loses communication to external data sources.
  - Field experiments on two control vehicles operating in heavy morning rush hour traffic on the I-24 Freeway near Nashville, TN. We implement our controllers using low-cost hardware, to enable scalability of our approach. Our findings from the experiments show that we spend 16.6% of time following the variable speed limit, 24.0% of time above the speed limit due to prevailing traffic, and 59.4% of the time in a car-following mode to prevent forward collision.
- *Publications:*
  - M. Nice, G. Gunter, J. Ji, *et al.*, "A middle way to traffic enlightenment," *Proceedings of the ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPS)*, 2024

These contributions fit together to enable novel investigations and deployments of experimental control, connectivity, and automation in cars. This testbed was created with mobile traffic control in mind. As a result, this testbed is extensible to service other experimental car research featuring any combination of automated control, human-machine interfaces, large scale, open road deployments, multi-modal networking, and heterogeneous fleets.

The remainder of this dissertation is organized as follows. Chapter 2: A literature review on related research, and works relevant to each successive chapter. Part 1, with Chapter 3: CAN Coach is introduced; a human-cyber-physical system achieves experimental vehicle control objectives by leveraging a human-in-the-loop; and Chapter 4, a pipeline for experimental automated vehicle control, featuring a reinforcement learning-based nominal control algorithm. Part 2, with Chapter 5, which focuses on the challenges of scaling the experimental automated vehicle platform to a team of 100 vehicles and introducing the capability to experimentally control traffic systems via automated vehicles. Part 3, with Chapter 6 which adds to the scalable platform by integrating connectivity with infrastructure-based variable speed limits; and Chapter 7 which adds to the scalable platform by considering and addressing the discrepancy between posted variable speed limits (VSL) and the prevailing traffic speed as a safety concern for prospective VSL-compliant vehicles. Concluding remarks in Chapter 8 discuss the work presented and the future research directions thereof.

2 | Related Work

> No worthy problem is ever solved in the plane of its original conception.
>
> George Saunders, A Swim in a Pond in the Rain

This chapter is separated into a collection of related but distinct sections. These cover related research areas to help characterize the etymology of the contributions of this dissertation, and to aid the reader in understanding the landscape in which this dissertation lies.

Traffic Control Systems on Freeways

Fixed traffic control with computer-based systems have been in operation for more than 40 years, working to improve traffic flow, reduce traffic collisions, and reduce energy consumption [22]. Variable speed limits (VSL) [23], ramp metering [24], and traveler information systems can theoretically reduce traffic shockwaves [25], and they have been deployed to give information to drivers helping them mediate bottlenecks and congestion [26]. Since the initial deployment of VSL several decades ago, the effectiveness of VSL on road safety and mobility has been investigated in both simulation and field tests [27]–[29]. Empirical studies have reported important safety findings. For example, a Belgian study reported an 18% reduction in injury crashes and a 20% decrease in rear-end collisions after the VSL implementation [5]. Similarly, a study conducted over 72 months in Seattle demonstrated a 32.23% reduction in overall crashes, with the most significant impact observed in rear-end collisions [6]. While the safety benefits of VSL are generally positive, their effectiveness is highly sensitive to the rate of driver compliance [7], [8]. Simulation studies affirm that the compliance rate is crucial for the performance of VSL [9], [10]. Challenges in implementing automated speed enforcement in North America contribute to low compliance [8]. Additionally, the large gap between posted and prevailing speeds can further impair driver compliance [11].

Mobile traffic control differs from fixed traffic control because it leverages vehicles as mobile actuators on the traffic flow [17], [30]. Traditional systems have a fixed sensor network and rely on human drivers *in situ* to react to information displayed on large signs. Mobile traffic controllers, which co-locate a sensor network and control decisions, have continuous opportunities for agents to make fine control decisions to adjust the shape of the traffic flow as they travel. Instead of a more blunt tool like road-side signage, mobile traffic control can directly and continuously communicate sensor data and control to a proportion of embedded mobile actuation vehicles. Systematic improvements can be gained [31] wherever vehicles are, instead of being limited to the domains of smart infrastructure installations.

Efficacy of Mobile Traffic Smoothing

To effect traffic phenomena such as traffic waves via mobile actuators, it is necessary to be able to control the velocity and spacing of some individual vehicles. Some related works deploy 10-40 vehicles to study emergent traffic congestion phenomena on a closed course [16], [32], [33], as understanding their nature can lead to improved outcomes. These ring road experiments study string stability in a ring, which is similar to an infinite single lane string of vehicles, on a closed course.

The traffic wave dissipation study [16] builds on empirical studies on traffic waves in the ring road [32], [33] by introducing a single autonomous vehicle (AV) could be used to dampen stop-and-go waves on a ring road with 21 human drivers, yielding sharply improved fuel efficiency. The work in [34] studies traffic smoothing with four connected AVs (CAVs) and demonstrates that the connectivity can be used for more effective dampening of waves on a single-lane, eight-mile-long public road. Another approach to reducing the severity of traffic jams at scale is through vehicular-based [35] variable speed limits, which is an adaptation of classical infrastructure based approaches [22], [25], [26].

Connected and autonomous vehicles will have effects on traffic flow stability and throughput [36], [37], but it remains to be seen if these effects will be positive. Other works have considered the wave dampening properties of existing commercially-available cruise controllers, with [38],[39],[14] all observing that the vehicles they tested were string unstable i.e. driving them in a series results in amplified speed variability, and thus increased traffic waves and fuel usage too. Ciuffo et al. [40] laments this fact, considering ACC was once heralded as a technology to improve mobility in anticipation of its OEM implementation, as CAVs are being anticipated now.

Networking

Using networking technology to improve vehicle control and properties of traffic is a longstanding area of interest. Notable examples include the use of *Cooperative Adaptive Cruise Control* (CACC) [41]–[43] systems, which provide longitudinal control of the vehicle using information exchanged between vehicles via vehicle-to-vehicle communication to improve traffic stability and throughput. Dedicated short-rate communication (DSRC) [44] has been studied in its effectiveness to improve safety[45], [46], vehicle-to-infrastructure testbeds [47], [48], and with vehicle ad-hoc networking (VANETs) [49]–[51]. By their architecture, edge-based decentralized VANETs have challenges with reliable and secure connections; on the other hand a centralized network like the cellular network may have other reliability and security vulnerabilities solvable with an edge-based decentralized network. The roll out of nearly universal cellular networking has allowed for in-depth field tested comparison of vehicle networking modalities (DSRC vs. 4G LTE)[52]. It remains unclear what kinds of networking may be introduced widely in commodity vehicles for inter-vehicle or infrastructure communication.

Human-in-the-loop Control

Designing vehicle control systems via modeling and simulation [25], [35], [53], [54] has drawbacks due to the fidelity of the modeling. To investigate this gap, and reduce the large overhead costs (technical, capital) in experimental vehicle control, researchers have employed a human-in-the-loop to effectively actuate experimental control. Driver assistance systems to promote augmented human driving have been tested on real vehicles in [3], [55]–[57]. These systems consider objectives of eco-driving, or velocity-matching, and using speed-advisory systems. These implementations rely on connectivity, and real-time infrastructure communications to estimate traffic conditions. Using exclusively on-board vehicle data and decision-making would make the overhead for experimental control even lower. The use of on-board vehicle data as a data source for analysis of driving behavior such as lane changing, turning, and driver categorization is [58], [59] becoming increasingly recognized. In the work [60], a driver behavior identification tool is presented that fuses on-board Controller Area Network (CAN) data with sensor data from an inertial measurement unit and

a GPS unit to classify normal and aggressive maneuvers in real-time. These analyses are not used to close the control loop with the human driver in real-time. However, it does establish the potential of using sensor fusion at run-time to understand driver behavior. A concrete step toward a scalable testbed for experimental control is to investigate the extent of the viability of a human-in-the-loop with on-board sensing for achieving experimental control objectives.

Roots of Small Scale Vehicle Technology Research

There are two basic categories to characterize the roots of vehicle technology research: (1) works focused on automation and robotics, and (2) works focused on addressing traffic dynamics questions. There are many works which use a single vehicle with new technology implemented to answer research questions [61]–[65]. The autonomous driving demonstration of the mid-2000s DARPA Grand Challenge is a classic example of research showcasing a vehicle with breakthrough robotic or autonomous control [61]. These breakthroughs did not just pop into existence; they have research roots in robotics in autonomous vehicles in the 1980's [66], [67], which in turn is rooted in graph theoretical work from the 1950's [68]–[70].

Other kinds of small scale works focus on answering an empirical traffic dynamics questions which can be answered through use of deploying an experimental vehicle in the field [62], [64]. Broadly, these stretch back to General Motors researchers studying the empirical research questions on car-following in the 1950s[62] by creating new tools for measuring inter-vehicle distances.

Vehicle Technologies in Many Vehicles: Automation vs. Scale Trade-offs

Generally, increasing the number of fielded vehicles in an experiment comes at the expense of less advanced instrumentation and automation in each vehicle. This creates a "pareto front" for vehicle technology research, where the largest experimental fleets have the least advanced technological capabilities.

On one end of the spectrum are the small-scale works like those from the DARPA Grand Challenge on autonomous driving [61] with a high level of robotic control; in this context the scale was irrelevant, and the scientific contribution resided in demonstration of a groundbreaking technology. At its core, the works pushing the frontier of automation[71], [72] are concerned with getting a new technology demonstrated in one vehicle, not wide adoption. In the middle of the "pareto front" are areas such as vehicle platooning research, which as early as more than 30 years ago [73] has explored installing novel vehicle technologies in a small set of commercially available vehicles [74] to explore the effect of technologies in clusters of vehicles. These vehicles do not strive to have highly automated driving, but enough automation to test technological innovations in small sets of a few vehicles in a single lane or track. A challenge confronted in this dissertation is that pressing research questions need large-scale vehicle deployment and some novel connectivity or autonomous control techniques. This dissertation aims to contribute technologies to enable open road deployment of new technologies borne from and dependent on automation at-scale, and showcase their deployment.

Mobile Sensor Networks

Already, distributed experimental sensing in the wild has been featured and adopted across domains: on roadways [75]–[78], in estuarial and riverine settings [79], pastures [80], and in production-scale fermentation processes [81]; distributed sensing combined with experimental control, at the core of mixed autonomy traffic

control, is not yet pervasive. In both [75] and [76], 5-10 vehicles are deployed with sensing hardware and software to collect data and some form of networking to process and analyze this data. These works build off of the idea that vehicles are inherently a mobile sensory platform for which novel applications can be built. Properly leveraging vehicles as mobile sensor networks is challenging, evidenced by the unsuccessful attempt in [82] where low-cost stationary pollution sensors did not perform well in a mobile sensing pilot. Information is collected in a semi-automated way in [75], [76], [82] and available eventually, but not in real time. In another domain, hydrology, mobile sensor networks were created at a scale of 100 [79] to track flows and quantify hydrodynamics in unknown estuarial and riverine settings. The work [77] demonstrates a mobile sensing network measuring emergent traffic phenomena [78] at a scale of 100 instrumented vehicles. Mobile sensing networks have the unique ability to sense wherever the network they reside within flows; their perspective is orthogonal to fixed sensor networks which have better use to understand the nature of a set of specific locations.

Naturalistic Driving Datasets

Naturalistic driving studies, which vary in fleet size from around 10 [83], to 100 [84], and then thousands [85]–[88] of deployed vehicles, are not pointed at a specific research application or question *per se*. They aim to provide a valuable resource to the research community: volumes of naturalistic driving data from a broad scope of drivers, vehicles, and locations. Outside of the research community, vehicle OEMs collect data from their fleets of vehicles which are in the millions; this data is kept private for competitive advantage. A limitation of the large research data sets is that they cannot be used to evaluate new vehicle technologies that have since and will continue to emerge. On-board vehicle sensor datasets are used for research works [58], [89] which derive scientific contribution from the collection of high fidelity data, which can capture finely what is going on inside the vehicle from the perspective of control systems as opposed to very coarse fixed infrastructure sensing (e.g. induction loops).

Vehicle to Everything

Vehicle-to-everything (V2X) is an overarching term that encompasses various forms of vehicle communication, including vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I). V2V communication is pivotal for the future of intelligent transportation systems, particularly for CAVs. V2V has been applied to reduce time headway for platooning of connected vehicles, thereby enhancing traffic flow [90]. In addition, [91] presents a cooperative dynamic intersection protocol for CAVs, utilizing V2V communications and perception systems, to safely and efficiently navigate these intersections. The proposed protocol significantly improves traffic throughput and minimizes trip delays when compared to baseline models. Complementing V2V, V2I focuses on the interaction between vehicles and road infrastructure. There is a growing body of simulation studies that explores the integration of VSL and CAV within the broader context of V2I communication. For instance, Li et al. [92] demonstrated that integrating V2I with VSL and cooperative adaptive cruise control (CACC) can effectively reduce rear-end collision risks. Furthermore, Grumert et al. [93] showed that the benefits of V2I communication, autonomous vehicle control, and individualized speed limits for VSL systems result in harmonized traffic flow and reduced exhaust emissions.

There are few studies of vehicle-to-infrastructure (V2I) field experiments mainly because of the inaccessibility CAVs and communication gaps between the infrastructure operators and the vehicle automation

systems. Ma et al. [94] conducted a field experiment on an active freeway with recurring congestion, employing three V2I-equipped vehicles to implement a simple speed recommendation algorithm. The study used probe vehicles to measure the impacts on the overall traffic flow and found that the V2I-enabled speed recommendation algorithm reduced oscillatory behavior in the instrumented vehicles without negatively affecting travel times. The control effectiveness from a small portion of automated vehicles has been further demonstrated in simulation [10], which shows that a small number of vehicles complying with the speed limit has a greater *effective compliance rate* since non-complying vehicles have limited ability to maneuver around complying ones.

Part I

Establishing New Pipelines for Experimental Control

3 | Flexible Experimental Vehicle Control with a Human-in-the-loop

> Ah, ah, I almost forgot... I'm also going to need you to go ahead and come in on Sunday, too. We, uhhh, lost some people this week and we sorta need to play catch-up. Thaaaaaanks.
>
> Bill Lumbergh, Office Space (1999)

This section includes material from a publication:
M. Nice, S. Elmadani, R. Bhadani, *et al.*, "Can coach: Vehicular control through human cyber-physical systems," in *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, 2021, pp. 132–142

3.1 Introduction

The goal of Lagrangian control is to control a few particles within a flow in order to effect the overall flow. In the domain of transportation, this may be carried out by control of a few vehicles in the flow of traffic in order to achieve a control objective. In order to regulate certain kinds of traffic phenomena, it is necessary to go beyond velocity control (which might leave space that encourages lane-changing) in order to control the spacing between the control (ego) vehicle and the lead vehicle.

Research has been done to evaluate the idea of Lagrangian control of traffic with automated vehicles, demonstrating that vehicle automation can smooth traffic waves both in simulation [95] and in full-scale experiments [16]. These controllers dampen traffic waves through prescribed time-gap following, but they depend on advanced sensing and computation from the vehicles of tomorrow in order to close the loop with an accuracy and dependability that is on par with human-driven vehicles, with a requirement that 5% or more of the vehicles must be controlled vehicles.

This raises a motivating question: can a human driver drive in a manner that dampens traffic waves, and thus accelerate the potential to adopt Lagrangian control by having the human perform some tasks that might otherwise require advanced sensors? If a human driver can be enabled to accurately follow commands that would enable Lagrangian control, then it may be possible to rapidly deploy technology in the vehicles of today that combines the advanced perception found in human drivers with advice from a supervisory controller that provides updated time-gap set points to carry out a control objective.

In this chapter, we ask drivers to follow a vehicle (the lead vehicle) in their vehicle (the ego vehicle) in order to determine how to positively improve time-gap following through the use of a CAN Coach. This CAN Coach uses information from the CAN bus such as instantaneous velocity along with estimates of the relative position and velocity of the lead vehicle. These data were collected using only data from the CAN in the ego vehicle.[1] Drivers are asked to follow a lead vehicle with several different objectives, as well as to follow an imaginary vehicle (a ghost vehicle) by using the CAN Coach to follow a time-gap (a measure of time separation between the vehicles), but without the visual feedback to the human-in-the-loop from the lead vehicle.

[1] This work was conducted under IRB approval #200343.

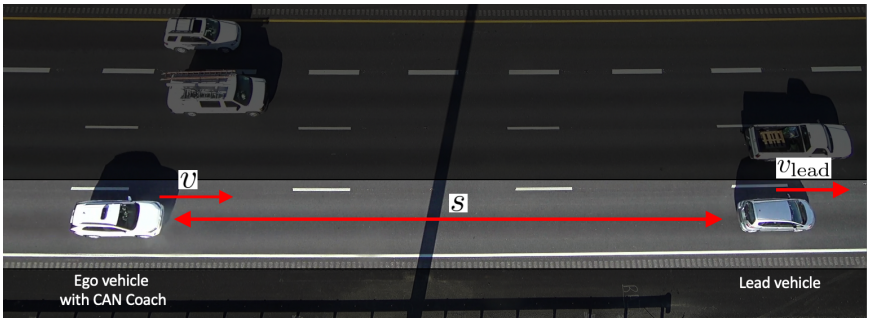


Figure 3.1: Ego vehicle with CAN Coach (White Toyota RAV4) and lead vehicle (Silver Honda Fit) in formation in the right lane.

Our results show that when coached by the CAN Coach, human-in-the-loop drivers can closely control the time-gap between the lead vehicle and the ego vehicle, i.e. space as a function of speed. The driving behavior with the CAN Coach is significantly improved compared to when subjects are instructed to use their own perception to achieve the control objective. When following the ghost vehicle, performance degrades due to the lack of a human-perceivable vehicle.

The contribution of this paper is the demonstration of an HCPS that is effective in controlling an individual vehicle which could be used for Lagrangian control of traffic flow. The CAN-based feedback system that is used (CAN Coach) enables drivers to achieve precise control objectives they could not otherwise achieve by giving extrasensory feedback informing adjustments of velocity and spacing. In other words, the CAN Coach can coach drivers to achieve "superhuman" driving tasks, and enables a human-in-the-loop to drive in a manner that could dampen traffic waves.[2] If successful, the application could scale widely given the low cost of the technology and the pervasive availability of the vehicle sensors; this would demonstrate the feasibility to deploy Lagrangian control techniques with the vehicle fleet of today.

The remainder of this chapter is organized as follows. In Section 3.2, we provide an overview of CAN data, the hardware platform, and the software platform. In Section 3.3, we describe the feedback design and the experimental design. Section 3.4 describes CAN data validation, and results of the CAN Coach system experiments. Finally, Section 3.5 outlines the planned extensions from our work presented here.

3.2 CAN Coach System Background

This section provides an overview of the CAN data, hardware platforms, and the software platforms used in our study.

3.2.1 CAN Data

To use the CAN Coach, depending on the control objective, we need measurements of the velocity of the ego vehicle, v, velocity of the leading vehicle, v_{lead} and the space-gap, s, the distance between the front bumper of the ego vehicle and the rear bumper of the lead vehicle (see Figure 3.1). By accessing the CAN data in the ego (following) vehicle, this information can be found via radar sensors and wheel encoders.

Vehicles with adaptive cruise control typically have front-facing radar, which directly measures information we need to compute the space-gap and relative velocity ($v_{\text{lead}} - v$). Note that for feedback in real-time,

[2] Here, we refer to superhuman in a colloquial sense, i.e., improving on what human drivers can do without assistance.

there are two related challenges with radar data. First, incoming radar data tracks many objects in a wide field of view and thus needs to be processed to extract a high quality estimate of the location and relative velocity of the lead vehicle. Second, any processing on the radar data must be done quickly so that the feedback to the human-in-the-loop remains relevant. These challenges—briefly discussed in Section 3.2.3—are otherwise outside the scope of this paper.

3.2.2 Hardware

The base vehicle used in this work is a stock 2020 Toyota Rav4 Hybrid vehicle. The vehicle has as standard equipment an *Adaptive Cruise Control* (ACC) system and *Lane Tracing Assist* (LTA) system. The ACC system depends on a stock forward looking radar unit that transmits relevant data on the CAN. To access this data, we used a Gray Panda manufactured by Comma.ai [96] as a data logging device. The Gray Panda is connected to a Raspberry Pi 4 via USB. The Pi decodes the messages, transforms them into Robotic Operating System (ROS) messages, executes the CAN Coach to generate an auditory feedback for the human-in-the-loop, and records all data from the car and from all ROS nodes.

3.2.3 Software

In this section, we discuss software that was developed to implement the CAN Coach System, execute the driving scenarios in a repeatable manner, and perform analysis of the captured data. Figure 3.2 shows how the information flows through the system.

Comma.ai Panda devices are used as interfaces to allow for reading directly from the CAN bus. Libpanda [97] is a C++ based library to aid in the construction of custom software, abstracting the USB interface for easy reading and writing to the CAN bus and for easy reading of the GPS module (if installed). The libpanda library was developed in order to capture all data available from the connected CAN buses, rather than the subset of data selected by Comma.ai for use by their open-source control packages.

The Robotic Operating System (ROS) is an open source framework for robotics. It provides the necessary tools and libraries to create and run peer-to-peer processes called nodes. ROS enables modular software development by utilizing reusable code packages [98].

The CAN Coach System consists of these ROS nodes:

- CAN Coach node: the core node in the CAN Coach System, it processes relative vehicle velocity and distance and generates feedback for the human-in-the-loop;
- Ghost Mode: uses a simulated velocity of a lead vehicle in order to create a virtual vehicle's position in front of the ego vehicle;
- Director: facilitates communication between the human-in-the-loop, mode changer, and CAN Coach, and ensures consistent execution of the experiment for each subject *without* the need to have an additional researcher advancing each test mode;
- Mode Changer: allows the human-in-the-loop to advance (or reverse) the mode of the test while driving.

CAN Coach

The CAN Coach, the eponymous ROS node, is what puts the human into the loop with the vehicle sensors, with audio feedback derived from CAN data in real time. The CAN Coach subscribes to velocity, relative distance, and relative velocity data obtained from the CAN bus, and when in the Ghost Mode subscribes to that node for relative distance and velocity of the virtual lead car.

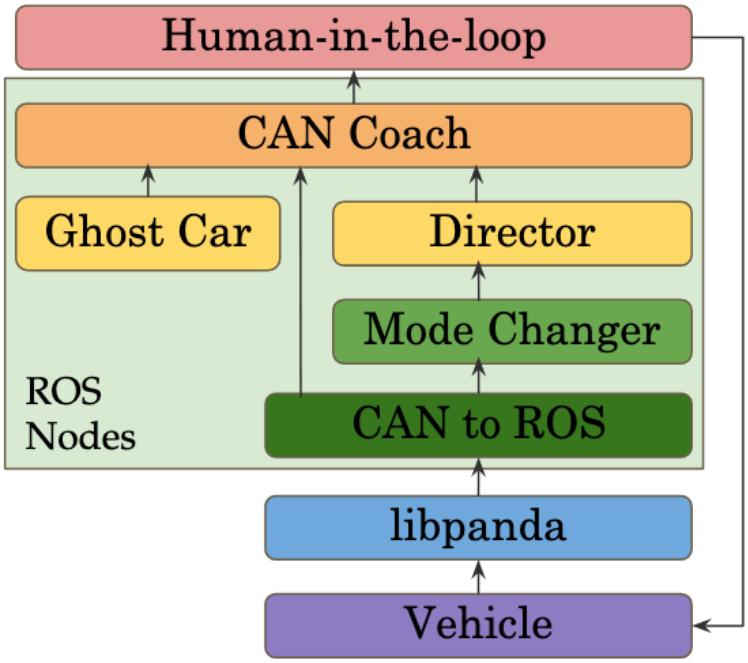


Figure 3.2: Diagram showing how information flows through the system.

With this information, the CAN Coach uses the 16 hi-frequency radar signals (20 Hz) that correspond to all tracked objects, along with a low-frequency radar trace (1 Hz) that corresponds to the lead vehicle's distance, in order to determine the relative distance s and relative velocity $v_{\text{lead}} - v$ between the ego and lead vehicles. This is achieved by finding a match between the most recent location of the lead vehicle and a buffer of recent raw radar data. With a match to the raw radar position, there is a corresponding relative velocity measurement. The CAN Coach can then communicate this information to the human-in-the-loop by producing the sound apt for the current control objective and feedback type.

Ghost Mode

The Ghost Mode is designed to measure how well the human-in-the-loop can track a lead vehicle through feedback from the CAN Coach, but without the ability to see the lead vehicle. Additional information exploring the motivation and merit of the Ghost Mode is provided in Section 3.3.2.

In order to carry out this mode of the experiment, relative distance and relative velocity are measured from the ego vehicle to a virtual 'ghost' vehicle—rather than obtained from the CAN bus. Through code generated from a Simulink model, the ego vehicle's velocity signal and a simulated constant velocity of the ghost vehicle are integrated to simulate the relative distance and relative velocity between the ego vehicle and lead (ghost) vehicle.

The CAN Coach calculates the time-gap in Ghost Mode from the ego velocity, and the difference in the distance traveled for both vehicles during ghost mode plus a constant of 65 m. A companion Stateflow model periodically determines whether the virtual space gap between the ego and ghost vehicles has either grown too large (which may result in unsafe high speed by the human-in-the-loop to catch up) or too small (which may result in unsafe low speed to allow the ghost vehicle to pass). If the virtual space gap $s > 100$ or $s < -30$, then the virtual distance is reset to 0 m (plus the constant 65 m) to permit safe execution of the experiment.

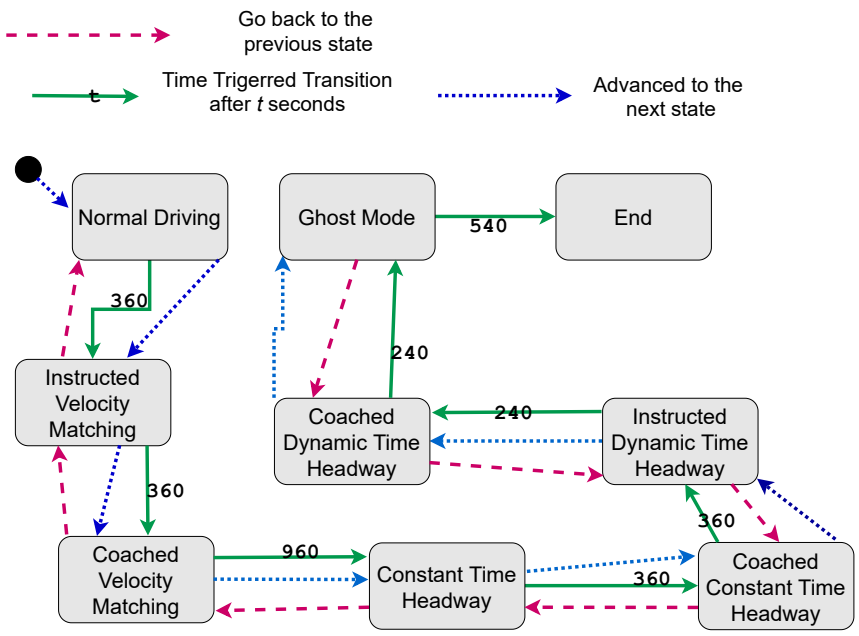


Figure 3.3: The Director node: Each experiment state lasts for a precise duration, or can be advanced or reversed based on an input.

The constant is added because at the initialized speed, 29 m/s, and time-gap set point, 2.25 s, the target space-gap is 65 m. This ensures that in the first minutes of Ghost Mode, if the ghost vehicle is reset it will return to the target state.

Director and Mode Changer Nodes

Due to ongoing health and safety practices (fielded in Fall 2020), only one person is allowed in a vehicle at a time—which introduces challenges in carrying out driving experiments where a passenger might otherwise control the experiment mode. We address this challenge through these supervisory nodes that require no synchronous interaction from the driver: a state-based Director, and a Mode Changer.

An abridged schematic of the Director node, simplified for readability, is provided in Figure 3.3. The Director advances modes based on elapsed time, yielding a precise, repeatable sequence with a fixed duration for each segment of the CAN Coach experiment. The Director node publishes the set point, feedback type, and current mode to the CAN Coach node every 0.5 s.

The Director subscribes to the Mode Changer, allowing the human-in-the-loop to advance (or reverse) the experiment mode. For example, if the lead driver thinks that the data would not be useful for analysis, that driver could request the human-in-the-loop subject to reset the mode using the vehicle's light stalk. The experiment's modes are the tested pairs of control objectives and feedback types.

3.3 Experimental Methods

This section describes the decisions made to design the feedback within the CAN Coach, and to design the experiments.

3.3.1 Feedback Design

The CAN Coach provides simple feedback to the human-in-the-loop, because complexity can be a burden [99]. In CAN Coach, when the human-in-the-loop is a threshold amount away from the control target, an audible signal is output to inform the human-in-the-loop of an action to take. We choose audio feedback because [100] shows auditory driving warnings correspond with better driver reaction times than visual warnings. Some commercial systems use discrete audible feedback, such as Forward Collision Warning Systems. They take into account both the physical driving context and human interaction; the precise timing of the discrete intervention is key to the system's success [101], [102]. Literature suggests, however, that continuous feedback [103], [104] is preferred over discrete feedback [105] for CAN Coach, because continuous feedback allows for greater control of precision over an extended period of time.

Under time-gap control, if the current time-gap differs from the desired time-gap (the set point) by more than ± 0.05 s, a sound is output to increase or shorten the time-gap. Under velocity matching control, if the relative velocity to the vehicle ahead differs by more than ± 0.4 m/s, a sound is given as output to increase or decrease the velocity.

There are only three messages the feedback communicates: speed up, slow down, or do nothing. A high pitched sound is used to indicate the human-in-the-loop should accelerate; a low pitched sound is used to indicate the human-in-the-loop should decelerate. No sound is emitted when the human-in-the-loop is near the set point.

3.3.2 Experimental Design

Description of the Driving Environment

The experiments conducted in this work are conducted with two vehicles on a freeway. A lead vehicle is used during the experiment to provide consistency in the testing environment, and to trace the route for the ego vehicle, at a speed of 65 mph ($\approx 29.0\,\mathrm{m/s}$). The ego vehicle is instrumented with the CAN Coach, and it follows immediately behind the lead vehicle while executing desired control objectives. Each drive begins and ends in the same locations for each tested driver, for a consistent roadway grade along the route on which the drivers are tested. The tests occurred on a 55 mile freeway route (Figure 3.4) at off peak hours to reduce interactions between the vehicles involved in the experiment and other road users. Each experiment lasts about 57 minutes per driver.

Prior to the start of each experiment, each driver is given an overview of the experiments, the route, the technologies used in the experiments, and the safety protocols. Drivers are informed to follow instructions given during the experiment to the best of their ability without compromising safety. Additional instructions are provided to the driver automatically using a standardized script prompted from the Director (Section 3.2.3), eliminating the need for any experimental staff to be in the vehicle with the human-in-the-loop subject.

At the start of the experiment, each driver is given a period of time to operate the vehicle following the lead vehicle under a *Normal Driving* regime. The driver is given instructions to follow the vehicle ahead without changing lanes.

Recall that the lead vehicle velocity is denoted by v_{lead} and the ego vehicle velocity be denoted by v (Figure 3.1). The relative velocity is $\Delta v := v_{\text{lead}} - v$, where a negative relative velocity indicates the ego vehicle is catching up, while a positive relative velocity indicates the ego vehicle is falling behind. The *space-gap*, denoted by s, is the distance between the front bumper of the ego vehicle and the rear bumper of

Figure 3.4: Route used to conduct the CAN Coach experiments.

the lead vehicle. The *time-gap*, denoted $\tau := s/v$, is the time required to travel the space-gap distance s when traveling at the velocity v [106].

In all experiments, the lead vehicle speed is set at 29.0 m/s (65 mph) using a standard cruise control system. Variations in the lead vehicle speed occur due to variations in the grade of the roadway as well as traffic conditions. Six drivers consisting of university students and staff were recruited to conduct the experiments. All drivers passed a university-required driver safety course.

Description of Control Objectives

During each experiment, the human-in-the-loop driver of the ego vehicle is asked to adjust the speed of the vehicle to achieve a desired control objective. Only one objective is used at a time. The control objectives are as follows.

- *Constant Time-gap*: Drive the ego vehicle such that the time-gap to the vehicle ahead is $\tau = 2.25$ s.
- *Velocity Matching*: Match the velocity of the lead vehicle (i.e. so that $\Delta v = 0$). The set speed of the lead vehicle is not announced to the driver (i.e., they are not informed that the lead vehicle is traveling at a fixed speed of 29.0 m/s). This meaningfully differs from Constant Time-gap because there is no fixed target for following distance.
- *Dynamic Time-gap:* Drive the ego vehicle such that the time-gap is $\tau = 2.25$ s or $\tau = 1.8$ s. The desired time-gap changes every 60 s.

Feedback Type

To meet the control objective, the driver is given varying degrees of information:

- *Instructed Driving (human visual perception feedback)*: The human-in-the-loop is verbally given the specific control objective (e.g., drive with a specified time-gap). In the case of the dynamic time-gap control objective, the human-in-the-loop is told when the time-gap changes, and the value of the new time-gap.

- *Coached Driving (human visual perception and CAN feedback)*: The human-in-the-loop is given all of the information provided in the Instructed Driving setup, as well as additional feedback using CAN data.
- *Ghost Mode (CAN feedback but without human visual perception)*: Ghost mode is a special setup in which the lead vehicle is replaced by a virtual (i.e., ghost) vehicle. CAN Coach is used to provide feedback to help the human-in-the-loop of the ego vehicle achieve a control objective relative to the ghost vehicle. The point of the ghost mode is to remove the possibility of the human-in-the-loop from visualizing the space-gap or the velocity of the lead vehicle. The human-in-the-loop is given the same instructions and feedback as for Coached Driving, with the major modification that the real lead vehicle is replaced by the ghost vehicle.

3.4 Results

This section presents the results from each human-in-the-loop subject under different control objectives and feedback types. We first consider the constant time-gap control objective, and compare the performance of humans-in-the-loop using Instructed Driving to the performance when CAN Coach is used to provide feedback. We show that CAN Coached driving improves the ability of the human-in-the-loop to achieve the control objective. Next, we consider the importance of the the ability of the human-in-the-loop to see the vehicle ahead by comparing driving under Coached Driving, to driving in Ghost Mode. Finally we consider additional control objectives including velocity matching and dynamic time-gap matching.

3.4.1 Data Preprocessing

CAN velocity and space-gap data are validated using GPS devices following the process described in [107]. Irrelevant data from the experiments must be removed in preprocessing. These data are collected due to experimental conditions on an open roadway. For example, we discard data near and on the ramp connecting the two freeways that form the driving route. For some drivers during some tests, a vehicle not part of the experiment can have undue influence the experiment, e.g., by cutting in between the lead vehicle and the instrumented ego vehicle. These data are removed prior to analysis. Through manual inspection of dash-mounted video data recorded during the tests, we determine simple thresholds that remove data corresponding to these anomalies outside of the intended testing environment. Specifically, we observe that we can eliminate data corresponding to these events by discarding data below the 10th percentile velocity, and when the relative velocity is below the 5th percentile or above the 99th percentile.

3.4.2 Constant Time-gap Control

In the first set of experiments, we compare the ability of a human-in-the-loop to achieve the constant time-gap control objective. We first present the results under Instructed Driving (visual feedback) compared to Coached Driving (visual and CAN feedback) conditions. Then we remove the possibility of visual feedback by considering a constant time-gap control to a ghost vehicle.

We consider the following performance measures. The control objective is to achieve a desired time-gap of $\tau_{\text{desired}} = 2.25$ s. The time-gap error at each measurement time instant t is computed as $\varepsilon_\tau(t) := \tau_{\text{desired}} - \tau(t)$. The empirical distribution of the time-gap error over the duration of the test can be computed,

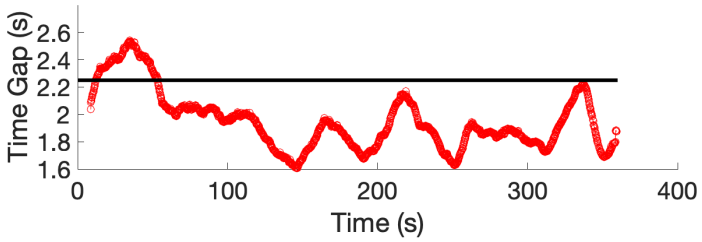


Figure 3.5: Instructed Driving for the Constant Time-gap control objective. Horizontal line indicates the desired time-gap of 2.25 s.

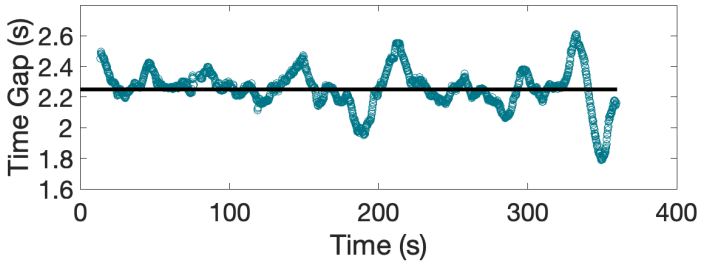


Figure 3.6: Coached Driving for the Constant Time-gap control objective. Horizontal line indicates the desired time-gap of 2.25 s.

as well as the mean error and the standard deviation of the error. Similarly, we can compute a corresponding space-gap error as $\varepsilon_s(t) = v(t)\tau_{\text{desired}} - s(t)$, and we can compute the space-gap error statistics.

Instructed vs. Coached Driving: Driver 1

We compare the results of a single driver under Instructed Driving to Coached Driving. Under the Instructed Driving (where feedback is based only on human visual perception), Driver 1 does not achieve the control objective. Figure 3.5 shows 6 minutes (360 seconds) of instructed driving. The driver has a time-gap that is consistently too low. In contrast, Figure 3.6 shows the same driver and control objective under the Coached setting in which visual and CAN feedback are provided. It is clear from the Figures that CAN Coach helps the driver correct the time-gap to achieve the control objective. To summarize the time-series shown in Figures 3.5 and 3.6, histograms of the time-gap error with and without CAN feedback is provided in Figure 3.7. Under Instructed Driving, the driver has a mean time-gap error of -0.3 s, and a standard deviation of 0.22 s. Under Coached driving, the mean time-gap error is 0.02 s, and the standard deviation is 0.13 s. In summary, the addition of feedback based on the CAN reduces the mean time-gap error by 93% and the standard deviation by 41%.

It is also possible to explore the corresponding space-gap error and the relative velocity to the lead vehicle under Instructed and Coached Driving. Comparing 3.8 and 3.9, we see that though there are marginal differences in relative velocity between Instructed and Coached Driving, there are important differences in the space-gap error. Coached driving substantially reduces the mean space-gap error (from -8.71 m to 0.088 m) as well as the standard deviation of the space-gap error (from 6.15 m to 3.65 m).

Instructed vs. Coached Driving: All Drivers

Figures 3.10 and 3.11 show the results comparing instructed driving to coached driving for all six drivers. Examining Figure 3.10, it is clear that the drivers had wide performance variation under Instructed Driving.

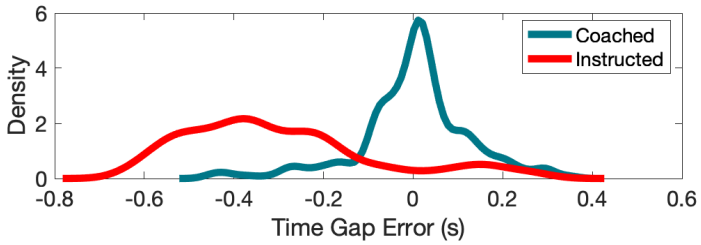


Figure 3.7: Driver 1 time-gap error distribution for Instructed and Coached Driving under Constant Time-gap control.

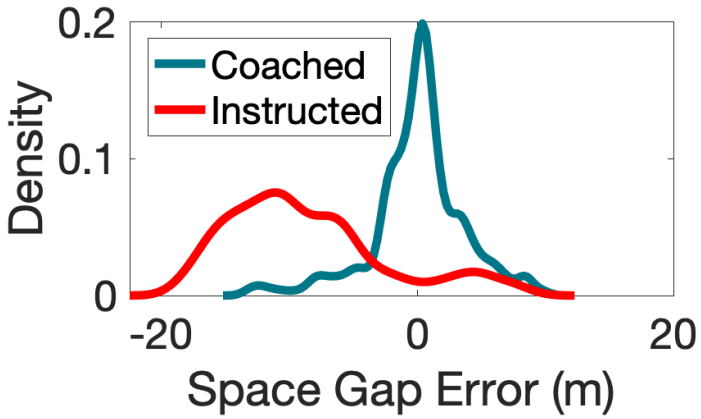


Figure 3.8: Distribution of space-gap error to the lead vehicle under Constant Time-gap control.

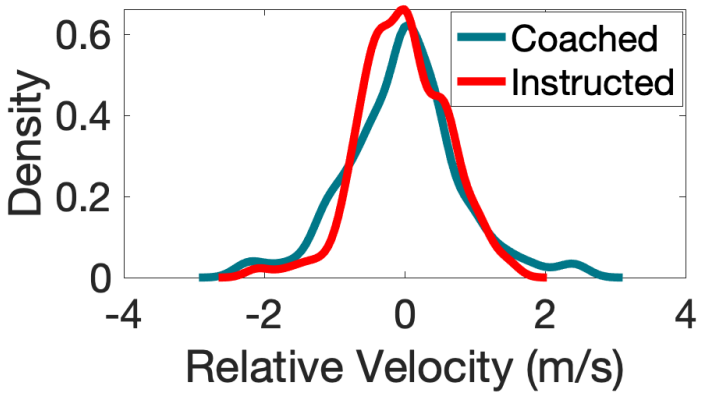


Figure 3.9: Distribution of relative velocity to the lead vehicle under Constant Time-gap control.

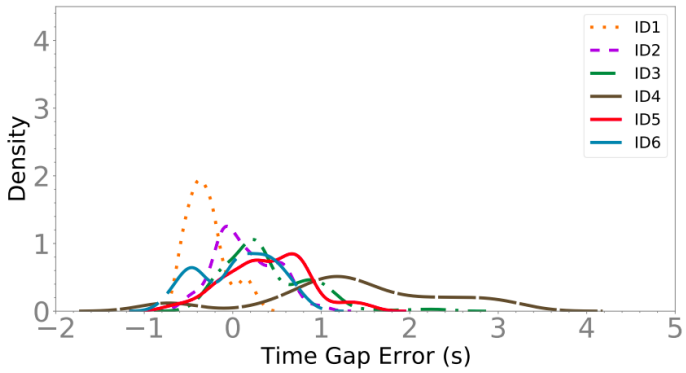


Figure 3.10: Time-gap error distributions for all drivers under Instructed Driving under Constant Time-gap control.

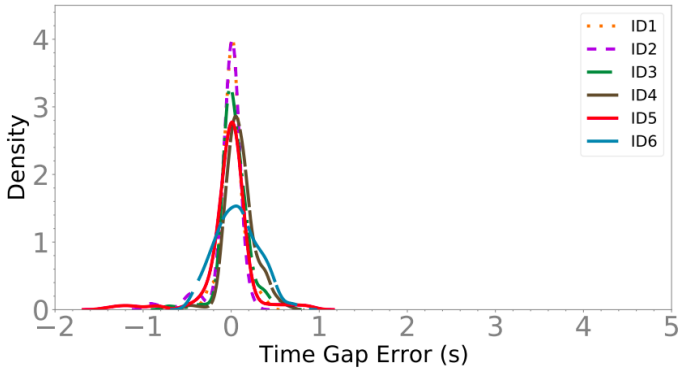


Figure 3.11: Time-gap error distributions for all drivers under Coached Driving under Constant Time-gap control.

This is true both in terms of average error from the set point, as well as the standard deviation. While aiming for a 2.25 s time-gap, the observed average time-gaps range from from 1.95 s (for Driver 1) to 3.61 s (for Driver 4). The standard deviation of the time-gap range from 0.22 s (Driver 1) to 1.01 s (Driver 4).

Figure 3.11 presents the time-gap errors for all six drivers under Coached Driving. All drivers are able to obtain a low mean error on the time-gap. The smallest time-gap error is 0.02 s (Driver 1) and the largest is 0.1 s (Driver 4). The variance of the errors are also substantially reduced. The standard deviations range from 0.13 s (Driver 1) to 0.28 s (Driver 5). To put the standard deviations into perspective, note that the standard deviations of all six drivers under Coached Driving are lower than the second most consistent driver (Driver 2; standard deviation of 0.34 s) under the Instructed mode.

Across all drivers (Table 3.1), the mean time-gap error reduced from 0.44 s under Instructed Driving to 0.05 s under Coached Driving, which is a 73% reduction in mean error. The drivers on average are more consistent, with the error standard deviation reduced from 0.49 s to 0.2 s (a 53% reduction). Only Driver 6 saw no reduction in mean error when operating under coached mode, which is due to the fact that Driver 6 had a very low 0.06 s mean error in instructed mode. All other drivers saw mean error reductions above 70% as a result of the feedback from the CAN bus.

	Instructed		Coached		% reduction	
Driver	mean	std	mean	std	mean	std
1	0.30	0.22	0.02	0.13	93%	41%
2	0.14	0.34	0.04	0.18	71%	47%
3	0.38	0.48	0.04	0.18	89%	63%
4	1.36	1.01	0.10	0.16	93%	84%
5	0.39	0.46	0.03	0.28	92%	39%
6	0.06	0.42	0.06	0.24	0%	43%
Avg	**0.44**	**0.49**	**0.05**	**0.2**	**73%**	**53%**

Table 3.1: Performance summary of Instructed and Coached driving under Constant Time-gap control. Units in sec.; Percent reduction is computed using Instructed as the baseline.

Removing Visual Feedback via Ghost Mode

In this section we consider the importance of human-based visual perception in the control loop. The expectation is that by removing the ability of the human-in-the-loop to also sense the vehicle ahead, the ability to follow the vehicle will degrade. The use of Ghost Mode allows for an ablation analysis because it removes the human perception of the lead vehicle by replacing it with an abstract ghost vehicle.

We first highlight the results of Driver 1 and then present the summary results for the remaining drivers. Figure 3.12 shows the time-gap error under Coached and Ghost Mode driving conditions. There is a drop in performance of the human-in-the-loop when following a ghost vehicle compared to a real vehicle, as quantified by the mean time-gap error. The mean time-gap error of 0.02 s under Coached Driving conditions increases to 0.15 s in Ghost Mode. Similarly, the standard deviation increases from 0.13 s in Coached Driving to 0.26 s when in Ghost Mode.

Next we consider the range of performances across the drivers. Figures 3.13 and 3.14 summarize the mean time-gap error and standard deviation of the time-gap error under Coached Driving and Ghost Mode driving. Figure 3.13 shows the the mean time-gap error increases for Drivers 1, 4, 5, and 6 when following a virtual vehicle rather than a real one. The mean time-gap error is identical for Driver 3 under both feedback modes, while the mean error for driver 2 is slightly reduced under Ghost Mode relative to Coached Driving. The mean time-gap error across all drivers is 0.05 s during Coached Driving, compared to 0.13 s in Ghost Mode. The standard deviation of the time-gap error distributions under Coached and Ghost driving is shown in Figure 3.14. The standard deviation increases for all drivers except Driver 2 when following a ghost vehicle compared to a real one. Combined, all drivers except Driver 2 saw either an increase in the mean error, standard deviation of the error, or both. This indicates that most of the humans-in-the-loop contribute to achieving the control task by seeing the vehicle ahead, even with CAN Coach feedback.

Note that performance under Ghost Mode is still improved compared to driving using only visual perception in the Instructed mode.

3.4.3 Velocity Matching Control

Next we consider a new control objective, in which the human-in-the-loop is asked to match the velocity of the vehicle ahead. There is a moderate reduction in standard deviation of the relative velocity for Coached Driving over Instructed Driving for the velocity matching control objective. Coached Driving feedback reduces mean relative velocity by 0.03 m/s, and reduces standard deviation by 0.16 m/s compared to Instructed

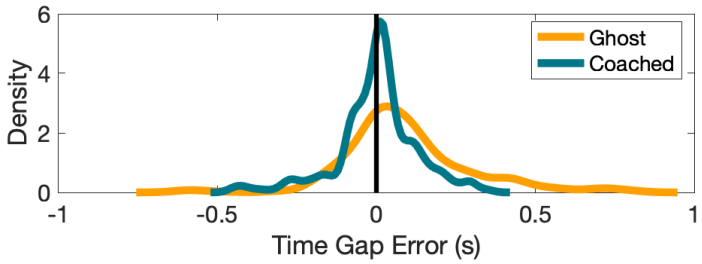


Figure 3.12: Driver 1. Time-gap error distribution for Coached Driving and Ghost Mode feedback under Constant Time-gap control.

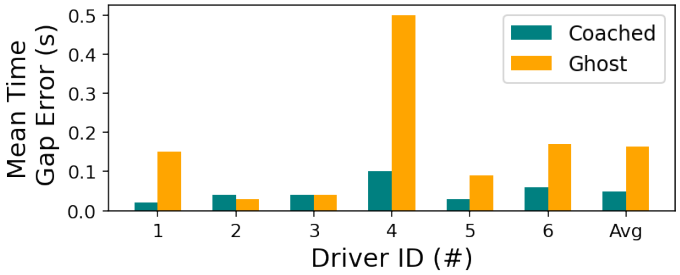


Figure 3.13: Mean time-gap error for Coached Driving and Ghost Mode.

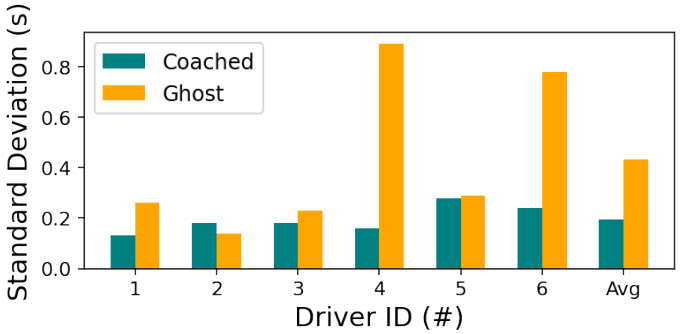


Figure 3.14: Standard deviation of the time-gap for Coached Driving and Ghost Mode.

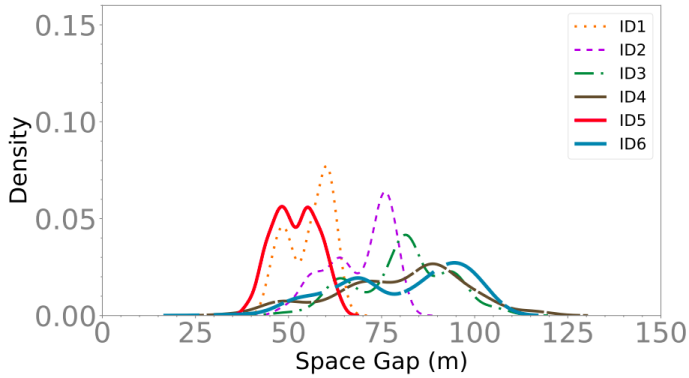


Figure 3.15: Space-gap distributions with Coached feedback for Velocity Matching

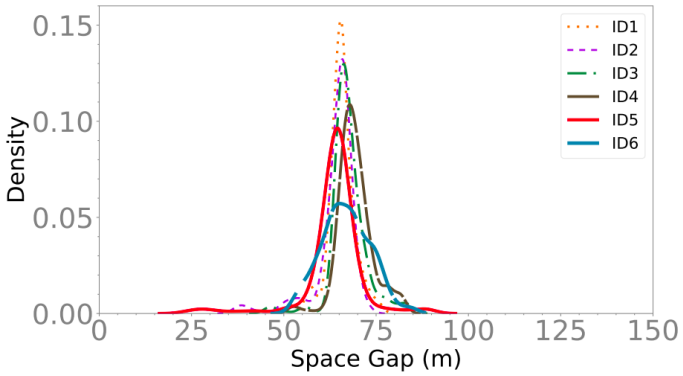


Figure 3.16: Space-gap distributions with Coached feedback for Constant Time-gap control.

Driving. The improvement is modest because the drivers under Instructed Driving were able to match the velocity of the lead vehicle reasonably well. As a consequence, CAN feedback resulted in only a slight reduction in error.

The most important and intuitive observation is that Velocity Matching does not standardize the space-gap across the drivers. The space-gap distribution under Coached Velocity Matching is shown in Figure 3.15, which can be compared to the space-gap variation under the constant time-gap control objective (Figure 3.16). When under velocity matching control, mean space-gaps ranged from 51.8 m (Driver 5) to 80.5 m (Driver 3). Under time-gap control similar relative velocity errors are observed, but the mean space-gaps are more tightly bounded. They range from 64.0 m (Driver 2) to 69.2 m (Driver 4). In summary, controlling the time-gap provides similar velocity control and better space-gap regulation than velocity based control.

3.4.4 Dynamic Time-gap

Finally, we consider the consequences of changing the time gap between two fixed values in a Dynamic Time-gap objective. Figure 3.17 shows the time-gap error distribution under the Instructed Driving conditions, while Figure 3.18 shows the time-gap error distribution under the Coached Driving condition. The mean time-gap error decreases from 0.21 s (Instructed) to 0.08 s (Coached). The standard deviation is decreased from 0.40 s (Instructed) to 0.30 s (Coached). This shows that CAN-based feedback can more accurately and precisely control the vehicle than without CAN-based feedback, even as the control objective increases in

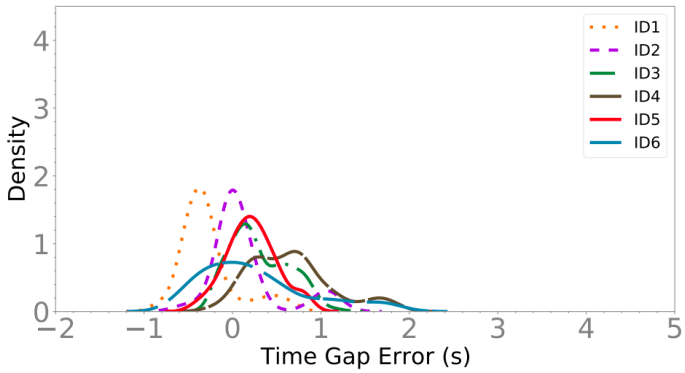


Figure 3.17: Time-gap error distributions for all drivers under Instructed Driving with Dynamic Time-gap control.

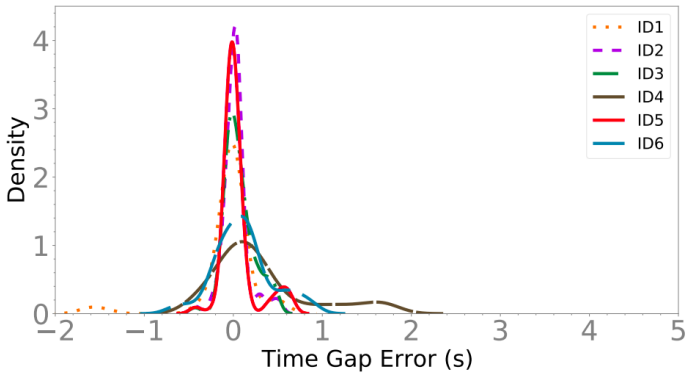


Figure 3.18: Time-gap error distributions for all drivers under Coached Driving with Dynamic Time-gap control.

difficulty.

When comparing Dynamic Time-gap Coached Driving to the Constant Time-gap Coached Driving, we see that performance was slightly worse with Dynamic Time-gap than with Constant Time-gap. From Constant Time-gap Coached Driving to Dynamic Time-gap Coached Driving, the mean time-gap error increased from 0.05 s to 0.08 s; the mean standard deviation increased from 0.19 s to 0.30 s.

3.5 Conclusions and Future Work

With an eye towards scalable Lagrangian traffic control, this work considered the possibility of vehicular control through a human-in-the loop system. By providing the driver feedback using sensor data reported on the CAN, drivers of varying skill levels are able to consistently achieve a desired time-gap control objective which effectively regulates both the velocity and the space-gap of the ego vehicle. The CAN Coach can coach drivers to achieve driving tasks that were not possible without feedback, opening a door to modify driver behavior for the benefit of the overall traffic flow. Though changing time-gap affects safety and efficiency, this work focuses on the successful implementation of a control objective, not the merits of the control objective itself.

In our future work we are interested in understanding the limits on the complexity of the control objective. For example it may be possible for drivers to explicitly follow a complex desired trajectory using time-gap

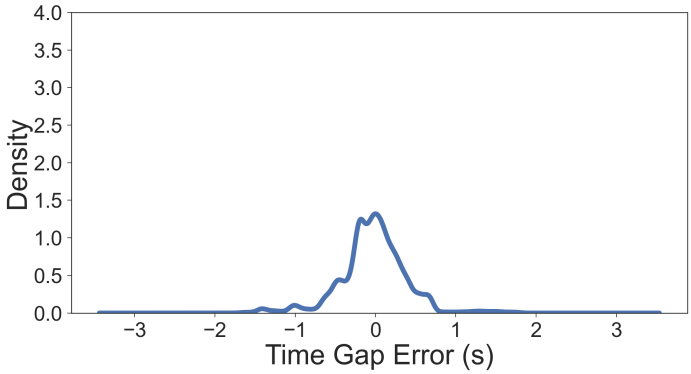


Figure 3.19: Time-gap error distributions for the stock ACC system under Dynamic Time-gap control. This distribution includes cut-in events.

based feedback to the desired trajectory, or for drivers to be coached to explicitly achieve a traffic wave stabilizing control policy. Enhanced driver assistance interfaces and experiments with a larger cohort of drivers will be important to generalize the findings presented here. Personalized feedback to drivers could improve performance and minimize intervention.

3.6 Appendix

This section contains an analysis of a stock ACC system tested under the same protocol as the Dynamic Time-gap, and comments on cut-ins that occur during data collection.

3.6.1 ACC Dynamic Time-gap Test

Though the aim of this work is not to recreate ACC with human drivers, it is interesting to compare between ACC driving and human driving with CAN Coach. In Figure 3.19 we can see how the stock ACC system compares to the human drivers with CAN Coach in Figure 3.18.

The same dynamic time-gap protocol was run, but this time with the stock ACC active. The ACC has a mean time gap error of -0.03 s and a standard deviation of 0.40 s. As a side note, the ACC error calculation is based on an assumed constant time-gap strategy of the stock ACC system and the stated time-gap settings in the owner's manual of the vehicle. The ACC driving shows a smaller mean error (0.03 s < 0.08 s), and a higher standard deviation (0.40 s > 0.30 s) in comparison to the average human driver under Dynamic Time-gap Coached Driving.

3.6.2 Discussion on Cut-Ins

The planned experiments were not structured to account for the semi-random nature of vehicle cut-ins. If, for example, during one driver's test there were two cut-in events and during another driver's test there were none, the results could be skewed. Broadly, the number and nature of cut-ins across drivers could be very different and bias the results. It would be interesting to do an analysis testing the velocity and spacing thresholds for cut-ins, and even make an online predictive model based on the state-space sensed by a vehicle in real-time.

In practice, the number of cut-ins were minimal. Each driver had 2.6 cut-in events on average over the course of the nearly hour-long drives. Some of these cut-ins are during the Ghost Mode, where the driver is

following a Ghost Car. In this setting the cut-in is even more abstract: there is a real vehicle cutting in front of the CAN Coach equipped vehicle which is following a virtual vehicle. Drivers were not given any specific guidance on how to respond to cut in events, and the sample size is small by construction (i.e., the selection of a remote freeway). Consequently we do not analyze the cut in data rigorously in this article, rather we discard the data surrounding the events.

4 | A Pipeline for Experimental Automated Vehicle Control

> Once you free your mind about a concept of harmony and of music being "correct" you can do whatever you want. So, nobody told me what to do and there was no preconception of what to do.
>
> Giogio Moroder

This section includes material from a publication:
N. Lichtlé†, E. Vinitsky†, M. Nice†, *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

4.1 Introduction

The increased availability of automated lane and distance keeping in modern vehicles has rapidly transitioned our roadways into the mixed autonomy regime where autonomous and human drivers all operate together. With the increased availability of automation in vehicles, i.e. mobile traffic actuators, it is becoming possible to perform Lagrangian traffic control in which control of the highway is dispersed amongst many vehicles in the flow. The ability to perform distributed control has brought closer the long-standing goal of AV research [108]–[112]: to use the programmability and fast reaction time of automated vehicles (AVs) to improve socially desirable highway metrics like congestion and energy efficiency for both humans and AVs.

In particular, prior work [16] has shown that even at low penetration rates of less than 4%, empirical and theoretical evidence suggests that AVs can significantly reduce stop-and-go traffic, a pernicious transitory phenomenon in which vehicles alternate between starting and stopping, consuming extra fuel in the process. However, prior approaches have a unifying problem: they are developed and analyzed in simplistic settings such as rings or hand-designed input perturbations. Testing on more complex settings is difficult as: 1) real-world highway sensor data are sparse and lack required resolution and detail needed for accurate modeling; 2) developing simulators that properly reproduce emergent structures from many-vehicle-interactions is challenging.

Building more complex models is heavily data constrained. Loop detectors only yield macroscopic statistics, while cameras tend to cover only a small portion of the roadway. This lack of available data is a fundamental issue as the trajectories of vehicles traveling through waves depends on the wave speed [113], and yet the wave speed is difficult to estimate with available stationary sensors. However, without an accurate means of reconstructing the stop-and-go traffic that is likely to occur on a particular highway, it is difficult to validate how a controller will perform when deployed on that highway. Consequently, it is unclear whether progress on control design for real-world smoothing is being made.

The contribution of this paper is a pipeline which avoids these aforementioned modeling challenges and produces a reinforcement learning (RL) controller that is then successfully deployed on four vehicles in dense highway traffic. This pipeline has three parts: (i) the data collected from human driving trajectories, (ii) the RL controller, and (iii) the deployment of the controller on physical vehicles.

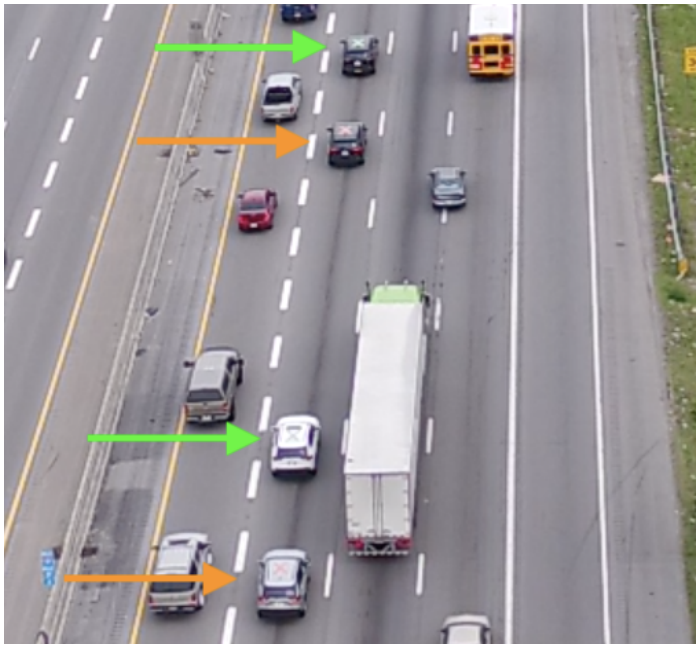


Figure 4.1: 4 of 11 vehicles in formation on the roadway. Green arrows and green X on roof indicate AV (AV), orange arrows and orange X on roof indicate human driven sensing vehicle (H). During experiments platoon formed in this order: [H, H, AV, H, AV, H, AV, H, AV, H, H], with no control over traffic flow consistently cutting in and out.

We are able to avoid modeling challenges by learning a traffic-smoothing controller directly from data collected from human driving trajectories. Instead of attempting to build a high fidelity simulation, we evaluate and train our controllers on collected highway trajectory data, ensuring that our controllers are learning to smooth a realistic representation of waves from the particular highway on which we intend to deploy AVs. We construct a simplified controller evaluation procedure in which a simulated mixed platoon of AVs and human drivers follows directly behind trajectories collected by a human driver on I-24, an interstate highway in Tennessee, scoring the controllers by their ability to improve energy consumption while maintaining traffic throughput. This approach sidesteps the aforementioned difficulties in calibrating both the waves and the microscopic car following dynamics. Using Proximal Policy Optimization[114], an RL policy gradient algorithm, we learn a controller that decreases the fuel consumption of the platoon in simulation by 16% for the AV and 10% on average for the platoon vehicles. Finally, we deploy the controller on real vehicles in highway traffic, showing the viability of this controller to create real-world energy savings and use of the complete pipeline.

The rest of this chapter is organized as follows: in Section 4.2 we discuss the data collection, cleaning, and analysis, in Section 4.3 we discuss the controller design and structure, algorithm, training details, and deployment pipeline, in Section 4.4 we discuss the simulation results, and experimental results, and finally in Section 4.5 we discuss and provide practical considerations to be considered in future work.

4.2 Training Set

Here we detail the human driver data collection procedure. The data serve as the basis for which we train wave smoothing controllers. We then briefly describe the data cleaning process and analyze the distribution

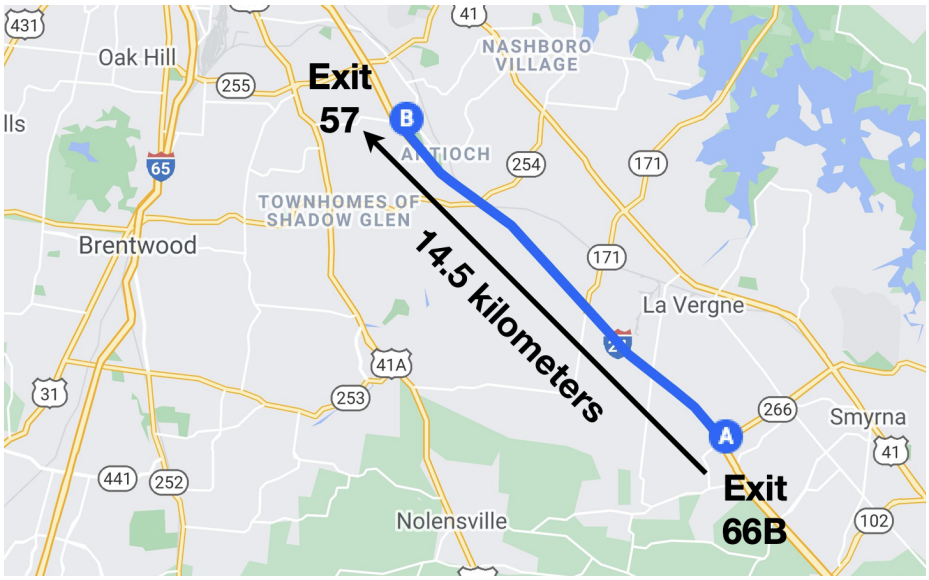


Figure 4.2: Portion of the I-24 highway on which we collected most of the dataset (section 4.2.1) and where we ran the experiments described in section 4.4.

of trajectories collected.

4.2.1 Data Collection

We collect data by recording trajectory data on a 14.5-kilometer-long segment (displayed in Fig. 4.2) of I-24 located southeast of Nashville, Tennessee. Each drive is conducted in an instrumented vehicle that logs CAN data via libpanda [97] and GPS data from an onboard receiver. Collected measurements from the vehicle CAN data include the velocity of the *ego vehicle* (the vehicle being driven), the relative velocity of the *lead vehicle* (the vehicle in front of the ego vehicle), the instantaneous acceleration, and the *space-gap* (bumper-to-bumper distance).

The drives are varied in the time of day, day of the week, direction of travel on the highway, and level of congestion. Each drive is made up of one or more passes through the highway stretch of interest. The data used to train the algorithm in this work are made publicly available at [115], along with more details on the data.

4.2.2 Data Cleaning

The raw data for a given drive were recorded in two files: a CAN data file and a GPS file. The pertinent data are pulled from the CAN data and interpolated to the GPS time, which is measured at 10 Hz. High-frequency CAN data are down-sampled and linearly interpolated to match the GPS time, and low-frequency CAN data undergo linear interpolation to match the 10 Hz GPS time as well. Distance traveled and direction of travel are computed using the GPS position data. Since the westbound data contain more regular congestion, we focus on westbound data for training. The westbound data contain 60 trajectories, representing 8.8 hours and 772.3 kilometers of driving.

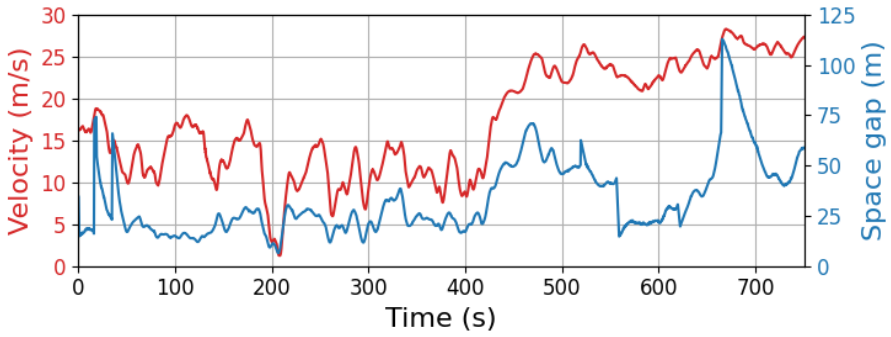


Figure 4.3: Velocity of the ego vehicle (blue) and space-gap to the lead vehicle (red) for a single trajectory in the dataset, containing sharp variations in both velocity and space-gap.

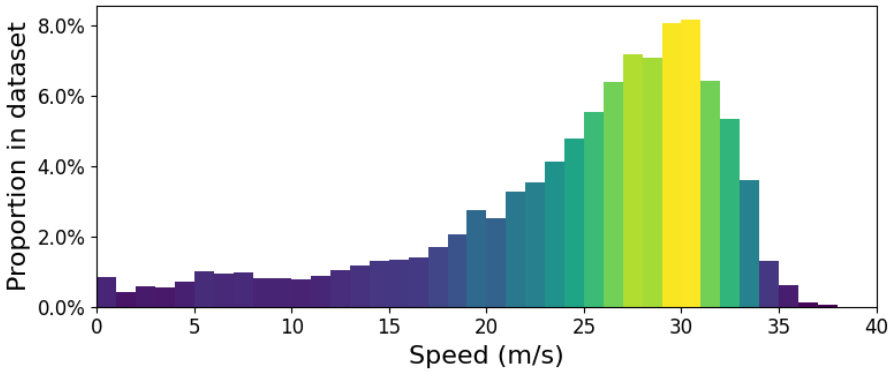


Figure 4.4: Histogram showing the distribution of velocities of the ego vehicle in the dataset.

4.2.3 Dataset Analysis

The data are collected over a wide range of traffic conditions ranging from congested traffic that is nearly stopped to free-flow, max-speed traffic, including many acceleration and deceleration patterns corresponding to stop-and-go traffic. Fig. 4.3 shows an example velocity and space-gap profile from a trajectory in the dataset, where we can observe the ego vehicle going quite rapidly from low to high speeds. While our main interest is in smoothing high-frequency waves, which occur primarily in congestion, the distribution of speeds in the training dataset, shown in Fig. 4.4, tends towards higher speeds. While we could filter the dataset to only contain low speeds, likely making the learning problem simpler, Fig. 4.3 suggests that regions of congestion are often quickly followed by regions of high speed. To ensure our controller behaves appropriately at high speeds and in transitions between high and low speed regions, we keep both low and high velocities in the training dataset.

4.2.4 Constructing the Training Environment

In order to use the collected data, we build a one-lane training environment where the AV follows behind the trajectory collected from the human drivers. The human driver is placed at the front of a simulated platoon, followed by the AV, followed by five vehicles driving according to the Intelligent Driver Model (IDM) [116] with a set of parameters that are string unstable below $18 \frac{\text{m}}{\text{s}}$, which ensures that the waves grow in congestion.

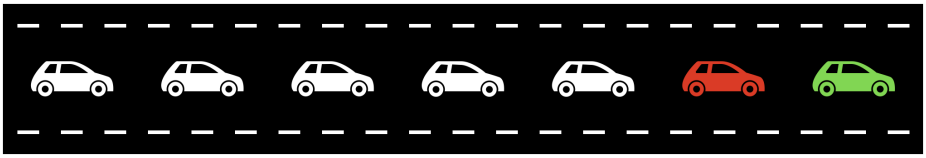


Figure 4.5: Vehicle formation used in simulation. A trajectory leader (in green) driving a speed profile drawn from the dataset is placed in front of an AV (in red) which is followed by a platoon of 5 human vehicles (in white), modeled using the Intelligent Driver Model.

Although having a full micro-simulation of the I-24 would allow for training on a model with complex long-range interactions between the vehicles, the simulator proposed here allows us to train on realistic driving dynamics that are representative of both the types of waves on this highway and how drivers react to wave formation. As an additional benefit, this single-lane simulation using half a dozen vehicles achieved 2000 steps per-second while a comparable micro-simulation of the full 14 kilometer road section would have thousands of vehicles in congestion and would be very computationally costly to evaluate.

Our collected dataset contains both the trajectory of our drivers and the vehicles in front of them (via space-gap and relative velocity data logged on the CAN). We discard the lead trajectories and do not use them for simulation as the lead trajectories contain both cut-ins (a vehicle cuts in between the lead vehicle and the ego driver) and cut-outs (the lead vehicle changes lanes). While cut-outs are likely unaffected by the behavior of the ego driver, cut-ins are likely a function of the spacing between ego driver and lead vehicle. Since our trained controller will have different space-gap keeping patterns, it is possible that the observed cut-outs would not occur given the controller's choice of space-gaps; to avoid dealing with this counterfactual we simply do not use the leader data for training and only keep ego vehicle data for our lead trajectories. Since the human drivers who collected the dataset intentionally rarely change lanes, our simulator consequently does not contain lead-vehicle lane changes. Finally, we note that we do not split the data into a train and test set; we train our controller on all of the available trajectories and instead use the deployment as our test set.

4.3 Method

In this section we describe the control design and structure, the details on how the controller is trained on the trajectory data, and the deployment pipeline that enables experiments to be conducted on a real vehicle platform.

4.3.1 Controller Design

For the model of the system dynamics, controls, and inputs we adopt the following system. As it is unclear whether the Markov property holds for this system [117], we will assume that the system described below may be slightly non-Markovian.

State space $[v, v_{\text{lead}}, h]$ where v is the AV speed, v_{lead} the speed of the vehicle right in front of it, and h the space-gap. All of these features can be acquired by using the forward-facing radar and the data collection software[97], [118] that we place on our vehicles.

Action space an instantaneous acceleration a, bounded between $[-4.5, 2.6]$ $\frac{\text{m}}{\text{s}^2}$, to be applied to the AV. Note that we do not allow the AVs to lane-change in this work.

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

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

Here E_t is the instantaneous gallons of fuel consumed by the AV (given by a piece-wise polynomial energy model calibrated to a RAV-4 Toyota vehicle; the fitting procedure and function coefficients are given in [119]), a_t the AV's instantaneous acceleration in $\frac{m}{s^2}$ and P_t its gap penalty, all at time-step t. The first term is intended to discourage fuel consumption, the second to encourage smooth driving, and the third to discourage the formation of large gaps that induce cut-ins or small gaps that might lead to driver discomfort. For our reward functions, we use coefficients $c_0 = 1.0\frac{1}{\text{Gal}}$, $c_1 = 0.002\frac{s^2}{m}$ and $c_2 = 2$, and penalize with $P_t = 1$ when the gap is below 7m, above 120m or when the time-gap (i.e., space-gap over speed) to the leader is below 1 second. These particular values were selected via an informal hyperparameter search and found to yield improved fuel consumption of the platoon while maintaining all constraints that might set off the penalty term P_t.
Finally, we note that our reward function does not include the energy consumption of the following platoon. While we experiment with such a reward, we observed more improvement by only optimizing for the energy consumption of the AV.

4.3.2 Controller Structure

The RL controller $G(\cdot)$ takes as inputs the current vehicle speed, the speed of the lead vehicle, and the space-gap provided by recorded or real-time CAN-to-ROS translation[118], and outputs a desired acceleration to a supervisory FollowerStopper[120] wrapper controller. The FollowerStopper leverages reachability analysis to verify safety and allows for total avoidance of a collision with the lead vehicle by taking in a desired velocity and returning a safe commanded velocity v_{safe}. The controller output during learning is acceleration-based; to convert it into a desired velocity we return $v_{\text{des}} = v_t + 0.6 \cdot G(\cdot)$ where v_{des} is the speed passed to the FollowerStopper and $G(\cdot)$ is the acceleration output by the RL controller. This desired velocity is then sent via CAN[97] to the vehicle's ECU for actuation. The particular "integration constant" 0.6 corresponds to the vehicle's responsiveness of about $\tau = 0.6s$, which is found by making the mapping from desired speed to realized speed as close as possible to the identity function, a mapping that we get from a transfer function approximating the vehicle's dynamics.

4.3.3 Algorithm

We train our policy using Independent Proximal Policy Optimization [114] (PPO), a policy gradient algorithm. We modify the standard PPO algorithm by providing the value function with a few additional inputs: the total distance traveled from start to time t, the total energy consumed by the agent at time t, and time t. The value function V^π estimates the discounted cumulative reward from a given state s_t and a particularly controller π. This quantity is difficult to estimate without the additional information we provide due to the partially observed state described in Sec. 4.3.1. The non-local information provided to the value function is used exclusively during training for variance reduction (see [114] for details), and these additional inputs are neither available nor needed by the controller during evaluation.

Training was done using the PPO implementation provided in Stable Baselines 3 [121] version 1.0, a Pytorch-based deep RL library. Training details and hyperparameters are provided in the linked code-base.

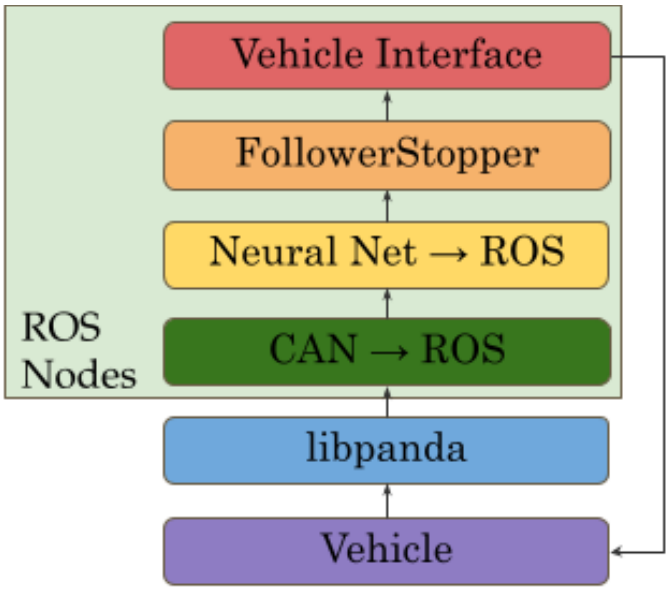


Figure 4.6: Diagram showing how information flows though the HWIL system when deployed. The vehicle sensors send data on the CAN bus. Libpanda[97] records the data and data are translated into ROS [118]. The neural net is embedded in a ROS node subscribing to pertinent data, and its output is filtered through a supervisory safety controller to get v_{safe}. This value is sent to the vehicle interface which takes a desired ROS command and sends it via CAN to the vehicle.

4.3.4 Deployment Pipeline

An initial software-in-the-loop (SWIL) step is taken to check functional correctness and interface testing in a 2-vehicle Gazebo simulation [122]. Structured velocity profiles (e.g., constant acceleration, sinusoidal, trapezoidal) are input to the controller to ensure outputs are not unusual. This also checks for software correctness. For hardware-in-the-loop (HWIL) deployment on the physical vehicle, there is a series of three tests to mitigate safety risks from the transition from simulation to physical vehicle before testing the controller on the I-24 segment. All three tests have varied input from the leader vehicle to ensure performance in non-equilibrium states.

First, the controller is tested in a 'Ghost Mode' as in [4] where the vehicle follows a simulated 'Ghost' vehicle as its leader. This provides the opportunity for a bad implementation to fail and crash into a virtual vehicle instead of a real one. The full HWIL setup is used with the modification that the real sensing done by the vehicle is replaced by a spoofed recording of a lead vehicle ahead using [118]. Second, the controller is tested in a 'CAN Coach Mode' as in [4] where the controller feedback is sent through a human-in-the-loop (HIL) for actuation. This second test occurs on a low-traffic, high speed route. Here the vehicle sensors feed real-time data into the controller, and the controller gives feedback to the HIL to indicate what input should be provided to the vehicle, but if the controller provides unsafe input to the HIL it is rejected to maintain safety and replaced with human control.

Finally, the controller is used on a low-traffic, high-speed route testing the complete HWIL control loop. Once these are successfully finished, the controller is ready to be tested on the heavy traffic, high speed I-24 roadway segment.

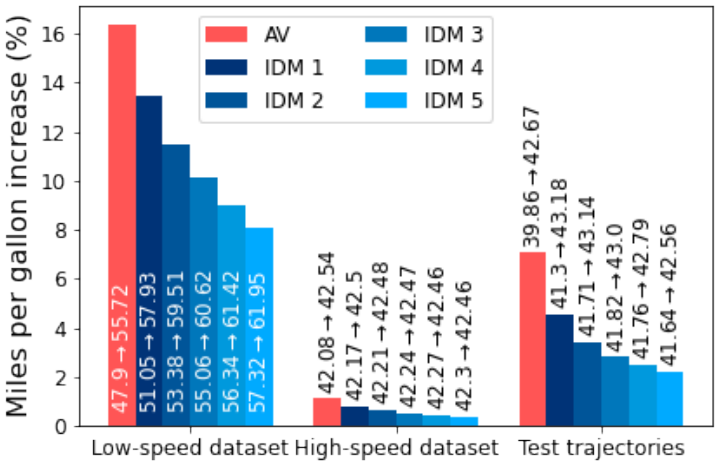


Figure 4.7: Percent improvement in MPG relative to a baseline in an IDM vehicle leads the platoon in Fig. 4.5. Each column contains both percent improvement on the y-axis and MPG values used to compute this improvement inside each column with IDM (AV) on the left (right) of the arrow. High and low speed columns are over the training set. The "Test trajectories" column is the controller evaluated on data from the physical test.

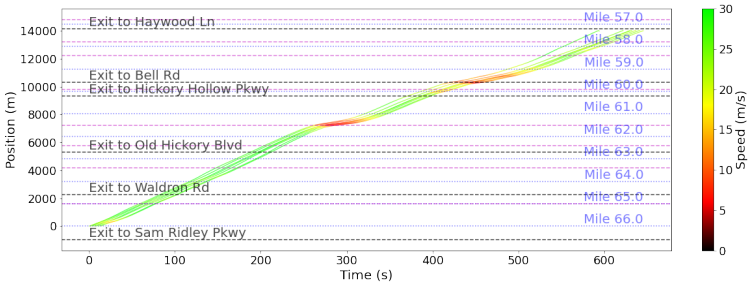


Figure 4.8: Time-space diagram showing the trajectories of our platoon of vehicle during the first test. We can observe two low-speed regions of congestion where vehicles following behind the AV could experience wave smoothing.

4.4 Results

4.4.1 Simulation Results

Here we analyze the performance of the controller in terms of energy efficiency improvements in miles per gallon (MPG) observed in our simulator. In Fig. 4.7, we compare the energy consumption of the AV and all vehicles in the platoon (as shown in Fig. 4.5) when the AV is using our RL controller compared to an IDM controller, over the whole training dataset. We split the trajectories by leader speed, computing the energy savings at leader speeds above and below $18 \frac{\text{m}}{\text{s}}$, which is the speed boundary beyond which IDM vehicles with the parameters used in this work go from being string-unstable to string-stable. The results in the left and middle columns indicate that most of the expected energy improvements from the controller will come at low speeds. While these savings are significant, in more complex settings imperfections in actuation, modeling of human drivers, and cut-ins would likely lower the actual improvement. The rightmost column is described in Sec. 4.4.2.

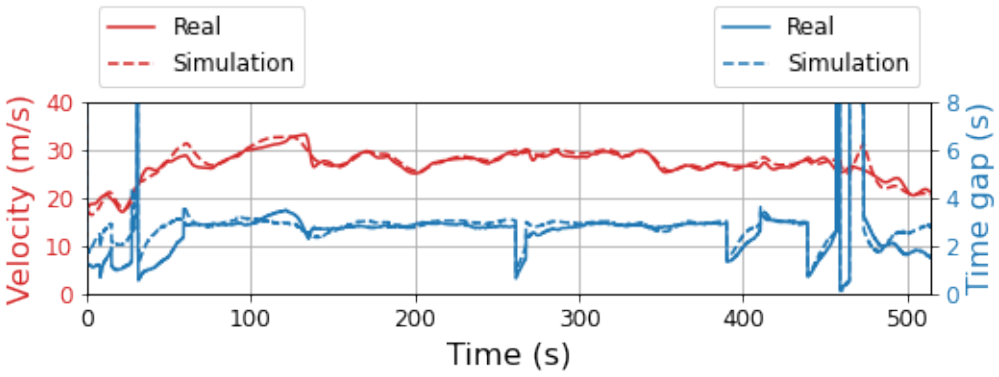


Figure 4.9: Comparison of velocity and time-gap between a real (solid) and simulated (dashed) roll-out. There are small divergences that occur around the cut-ins but the car mostly maintains a three-second time-gap in both cases.

4.4.2 Experimental Results

In this section, we describe the validation experiment conducted on the segment of I-24 shown in Fig. 4.2. We assess the success of the controller deployment onto AVs by showing an accurate match between simulation and reality. Finally, we seek to determine whether our controller improved the energy efficiency of its platoon.

Fig. 4.1 shows four vehicles from the eleven-vehicle platoon of alternating humans and AVs that we deployed on I-24.

For each test, we got the platoon onto the highway without any non-platoon vehicles lane-changing into it. Once on the highway, non-platoon traffic cut in and out of our platoon. Since our vehicles were only instrumented to sense the vehicle in front of them, the number of vehicles that managed to enter into our platoon is unknown. We ran experiments on August 2nd, 4th, and 6th of 2021, each day launching the platoon of vehicles three times and bringing the vehicles back to the start of the highway section in between each run. The controller presented here was only actuated on 08/06, over three tests that occurred at 6:45, 7:29 and 8:36 AM. Fig. 4.8 shows individual vehicle trajectories on a time-space diagram from the 6:45 AM test; the two regions of red correspond to congestion events. The deployment of the controller from simulation to real vehicles was overall successful as all tests ran safely and smoothly.

We investigate the effect of the sim-to-real gap induced by the presence of cut-ins and cut-outs, which we did not have when training our controller, as well as imperfect modeling of the transfer function of the AV. First, we attempt to compute a counterfactual baseline in which we replay our controller in simulation behind a trajectory collected during the tests. This mechanism is imperfect as the real-world trajectory has cut-ins and replaying a different controller behind it might affect the cut-in frequency. Without a model of lane changing, we cannot perform this counterfactual perfectly so instead we make the calculations assuming that both the times when cut-ins occur and the space-gap directly after the cut-in are unchanged. Occasionally, we choose to relax this latter condition in order not to experience, in simulation, cut-ins that would be more aggressive than what the real-world AV experienced. To that end, at each time-step t where a cut-in would leave the AV with a space-gap h_t^{sim} while the real-world AV experienced a space-gap h_t^{real}, we set $h_t^{\text{sim}} = \max(h_t^{\text{sim}}, \min(h_{t-1}^{\text{sim}}, h_t^{\text{real}}))$.

Fig. 4.9 shows the velocity and time-gap (space-gap divided by velocity) of an AV from the validation experiment as well as the replay of the trajectory in our simulation using the counterfactual cut-in mechanism

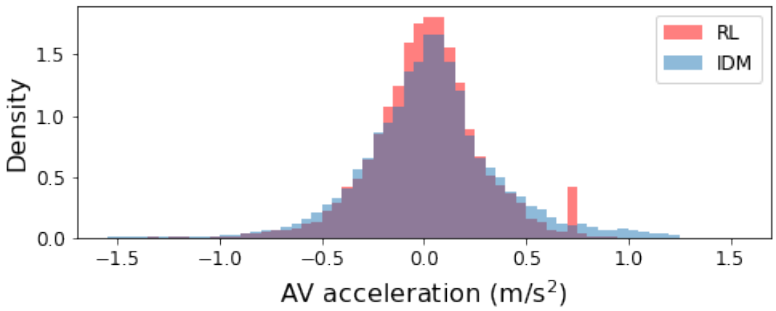


Figure 4.10: The density of the AV acceleration when simulating an AV or an IDM vehicle behind leader trajectories from the tests. The AV case places less mass at high, energy-consuming accelerations. The peak observed at 0.75 corresponds to the lead vehicle being out of range due to cut-outs.

mentioned above. The velocity profile of the vehicle closely matches its expected behavior computed in simulation. Although there are mismatches around cut-ins and cut-outs (regions where time-gap changes discontinuously), the time-gaps are relatively close and we can observe the vehicle roughly tracking a three-second time-gap in both cases. We observe similar results on the other trajectories we collected during the tests.

Finally, we analyze the potential fuel efficiency improvements from the validation experiment. The third column in Fig. 4.7 depicts the energy savings obtained when replaying in simulation using the trajectories collected during the experiments using the counterfactual cut-in mechanism mentioned earlier. We observe that the fuel efficiency of the AV has improved by 8% with additional small gains for the IDM vehicles. Fig. 4.10 shows the density of accelerations taken by the IDM vs. the AV; the higher density of large accelerations of the IDM vehicle are likely the reason for the improved fuel efficiency of the RL AV over the IDM AV. Unfortunately, the day of the deployment featured limited congestion so potential improvements are smaller than might be observed in heavier traffic conditions. More experimental testing on a number of days are needed to provide conclusive experimental energy savings results.

4.5 Conclusions and Future Work

In this work we propose and test a pipeline that allows for effective validation and training of traffic smoothing controllers. We collect over 700 km of training data that is used to build a controller validation system. This system avoids the fundamental modeling issues that have restricted the learning or design of traffic smoothing controllers to relatively simple settings, or prevented them from deployment on real cars. In our validation system, we use Policy Gradient methods to train a controller that improves the MPG of an AV by 16% and has benefits for the following human vehicles. We then construct a pipeline for porting these controllers to four AVs and perform physical validation experiments over three days. The behavior of the vehicle on the validation experiment closely matches its expected simulation behavior, suggesting that our pipeline is an effective mechanism for validating controllers.

There are a few missing features in our environment that merit further work. First, our simulator lacks counterfactual lane-changes. In future work, this can be addressed using the observed lane changes in the data to build a single-lane lane changing model that can be used to extend our simulation. In terms of the Markov Decision Process we design, our controller is memory-free, which may prevent the agent from

learning a predictive model of downstream speeds that can be used for further smoothing. Additionally, we do not penalize the energy consumption of the platoon; the addition of this penalty may lead to qualitatively different behavior. Finally, additional field experiments can support the assessment of our approaches in a range of traffic congestion levels.

Part II

Scaling Up for Mobile Traffic Control.

5 | Enabling Mixed Autonomy Traffic Control

> I tell this story to illustrate the truth of the statement I heard long ago in the Army: Plans are worthless, but planning is everything.
>
> Dwight D. Eisenhower

This section includes material from a publication:
M. Nice, M. Bunting, A. Richardon, *et al.*, "Enabling mixed autonomy traffic control," *arXiv preprint arXiv:2310.18776*, 2023

5.1 Introduction

This chapter introduces a new capability for connected automated vehicles (CAVs): mixed autonomy traffic control. Mixed autonomy means that a proportion of the vehicles are automated, and the remainder are not. Even when automated vehicles are a small fraction of the flow, mixed autonomy traffic control has the potential for significant societal-scale benefits [16], [35], [37], [53], e.g. reducing energy in stop-and-go driving by up to 40%[16].

In this work we deploy a fleet of CAVs 'in the wild', i.e. in a mixed autonomy setting on the freeway. The CAV fleet exchanges information to work collaboratively on a shared goal. Existing objectives for CAV fleets include platooning [73], [74] to reduce aerodynamic drag for the vehicles within the platoon, and operating as efficient automated ride sharing networks [123]. This chapter takes CAV fleets in a new direction, by deploying a CAV fleet with the objective to reduce phantom traffic jams [62], [124] by controlling traffic flow in the mixed autonomy setting.

Researchers want investigate the promise of robotics in mixed autonomy traffic settings, but there is a fundamental conflict. These technologies need to be investigated at scale *in situ* because there is no holistic replacement for the real physical traffic environment. Simultaneously, potential technologies have not yet been investigated *in situ* because of the issues of scale necessarily introduced to create and measure their impact; there is a gap in the tools and technologies to deploy CAVs at scale. To address this gap, we develop new scalable hardware and software that endows commodity vehicles with the ability to sense the traffic environment, perform vehicle control in a coordinated manner, and adapt to a complex field environment.

There are relatively few large-scale robotic passenger vehicle deployments in the field, because of the inherent difficulty of deploying experimental technologies at scale. The most visible examples are the highly automated vehicle fleets that use expensive suites of sensors and onboard computation in pursuit of SAE level 4+ driving. Our work differs in that the goal of our fleet is to change the emergent properties of traffic. We demonstrate deployment can be achieved at scale with an approach using low cost computation and stock sensors on today's commodity vehicles. The capability of mixed autonomy traffic control has been anticipated for decades [4], [18], [78], [95], [125]–[127]. Our low cost approach addresses the gap in tools and technologies, and enables the deployment of mixed autonomy traffic control with exchangeable candidate experimental control.

Figure 5.1: **Movie 1**: Deploying mixed autonomy traffic control with a fleet of 100 connected automated vehicles. Available at https://youtu.be/slH9nimpaY8

5.1.1 Contributions

1. We deploy the first large-scale team of connected automated vehicles (CAVs) for mixed autonomy traffic control.
2. We introduce a hardware and software platform to enable experimental autonomy and connectivity in commercially available SAE level 1 and level 2 automated vehicles. The platform supports experimental automated vehicle control, live mobile sensing with connected vehicles, high fidelity data collection, and scalability.
3. This platform enables an agile develop/deploy cycle for at scale cyber-physical systems research, empowering its field deployment.

5.2 Results

This work introduces the tools and technologies needed to transform a fleet of commodity vehicles into a team of connected automated vehicles. With these tools in place, we are able to deploy mixed autonomy traffic control in the wild with our CAV fleet. The demonstration of the CAV fleet circulates on a 5-mile loop of Interstate 24 (I-24) near Nashville, TN, a multi-lane freeway with heavy congestion. Each of the SAE Level 1-2 vehicles is augmented with an experimental control system that driver operators activate on the freeway in the same manner as existing cruise control systems. Leveraging their connectivity, the vehicles run collaborative control algorithms with the objective to smooth traffic waves in morning congestion. The results evaluate the effectiveness of our platform to create a scaled CAV fleet; we do not evaluate the effectiveness of specific exchangeable algorithms. First, we consider the fleet as a coordinated CAV team actuating on the congested traffic flow. Then, we analyze the deployment for control vehicle density and approximate penetration rate. Last, we introduce the novel tools and technologies which enable a single vehicle to become cohesive with others at scale.

5.2.1 A Fleet of 100 Mobile Agents on the Traffic Flow.

We deploy the fleet of CAVs on I-24 during heavy morning congestion from November 14-18, 2022. The CAV fleet runs experimental control algorithms distributed densely on a five mile stretch of roadway as shown in Figure 5.2. By extending commodity vehicles into a collaborative fleet of 100 connected automated vehicles, we are able to test in the wild for the first time mixed autonomy traffic control at scale. Our fleet

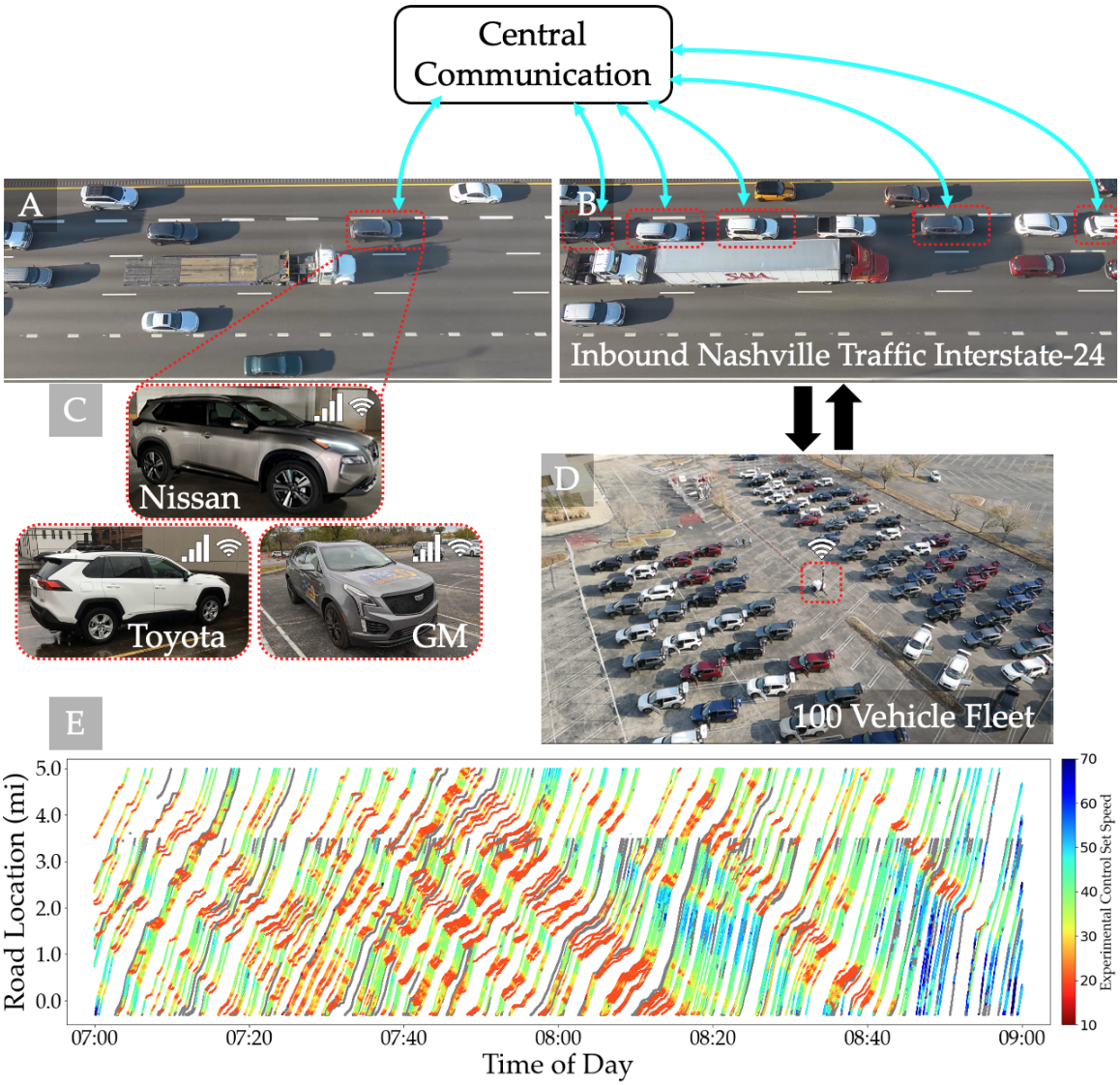


Figure 5.2: **Collaborative Automated Fleet as Mobile Agents on Traffic Flow.** The ephemeral fleet of 100 experimental vehicles (**D**) was deployed in varying traffic densities (**A-B**). Drivers circulated in congested morning traffic, and when they needed a break the vehicle had a fresh driver swapped in at our operational HQ (D). The heterogeneous fleet had 3 constituent vehicle platforms from 3 different major manufacturers (**C**). (**E**) shows the series of inbound Nashville trajectories from our large-scale experimental fleet over a 5 mile stretch of Interstate-24 in morning congestion. When experimental control is engaged, the commanded vehicle speed is shown according to the color bar; otherwise, the data point is colored gray.

drove a cumulative 22752 recorded miles (36616 kilometers) over a combined 1022 hours. In the field, tens of versions of software are deployed, with 6 deployed on 50+ vehicles and 19 deployed on 10+ vehicles. The deployed controllers are designed for distributed traffic wave smoothing control.

Our vehicle fleet faced varied traffic densities (Figure 5.2 A-B) in recurring congestion while running experimental mixed autonomy traffic control algorithms. The 100 vehicle fleet consists of 3 vehicle models from 3 original equipment manufacturers (OEMs) (Figure 5.2 C). The variation in OEMs required cross-platform support from our hardware and software tools. Vehicle-specific hardware was abstracted away with our lower-level software, providing a uniform interface for upper-level software modules [97], [128], including modular ROS integration for control developers and live tracking [129] for field test administrators. Vehicles circulated in traffic flow (Figure 5.2 A-B) until driver operators elected to rest at our operational headquarters (HQ) (Figure 5.2 D), providing an opportunity for fresh operator to continue circulating the vehicle. The HQ parking lot WiFi access point (Figure 5.2 D, in red), from a spliced extension of the building's fiber optic network, enabled high volume data transfer each time the vehicles are powered off. This network connection allowed for immediate data backup and data access to analyze control performance, which informed new software versions installed over-the-air on the next startup of the vehicles. Strym [130] and ROS [98] were key data analysis tools in ad-hoc and urgent analyses of in-vehicle network data informing field decisions. The CAV fleet was distributed throughout the inbound Nashville morning congestion in a 5 mile stretch of freeway. Figure 5.2 E shows 323 control vehicle trajectories, and where experimental control was actively commanding the vehicle, from 7AM-9AM on the test day. The varied colors indicate the commanded velocity, and gray indicates where the system was not active. Notice that for driver operators which turn around at MM 3.5 there is a consistent gray tip of their trajectory; this is a result of electing to disengage the experimental control system as the operator prepares to exit the highway. As they drive around, each CAV in the team shares information with our central server, which allows central control whitelisting, live tracking, control heartbeats, local sensor message passing, control recommendations and more; see the materials and methods section for more details.

5.2.2 A Deployment of Mixed Autonomy Traffic Control Examined.

To examine the mixed autonomy traffic control deployment, we examine the control vehicle density, control vehicle penetration rate in vehicle flow, vehicle volume, and looping I-24 driving routes. Vehicle density is a measure of the number of vehicles per roadway length, vehicle flow is a measure of the number of vehicles moving per unit of time, and the penetration rate is the proportion of vehicles of interest. Figure 5.3 A shows the density of control vehicles per mile throughout the 5 mile experimental corridor and morning of congestion. Density is calculated per 5 minute and 0.1 mile (528 ft) window. Vehicle flow during heavy congestion on I-24 ranges from 6000-8000 vehicles per hour across all lanes, which translates to a 2.0%-2.7% average penetration rate for our control vehicles from 7AM-9AM on Friday 11/18/2023 across all lanes of traffic. If we consider just the three lanes of four total on the roadway in which control vehicles were operating, the average penetration rises to a range of 2.7%-3.6%. This penetration rate could be sufficiently high for a distributed control system to effect the flow of traffic for all vehicles in the 5 mile experimental corridor. Previous work on a closed course ring road used 1 automated control vehicle and 20 non-automated vehicles (4.8%), i.e. 1 control vehicle per 260m (853 ft) ring circumference and achieved dramatic traffic smoothing behavior [16]; this density indicator to helps inform our understanding of approximate effective densities, but can not be compared directly to effects in large-scale deployment in the wild.

For reference on the scale of distance on the interstate, at highway speeds (70 mph) the vehicle a driver

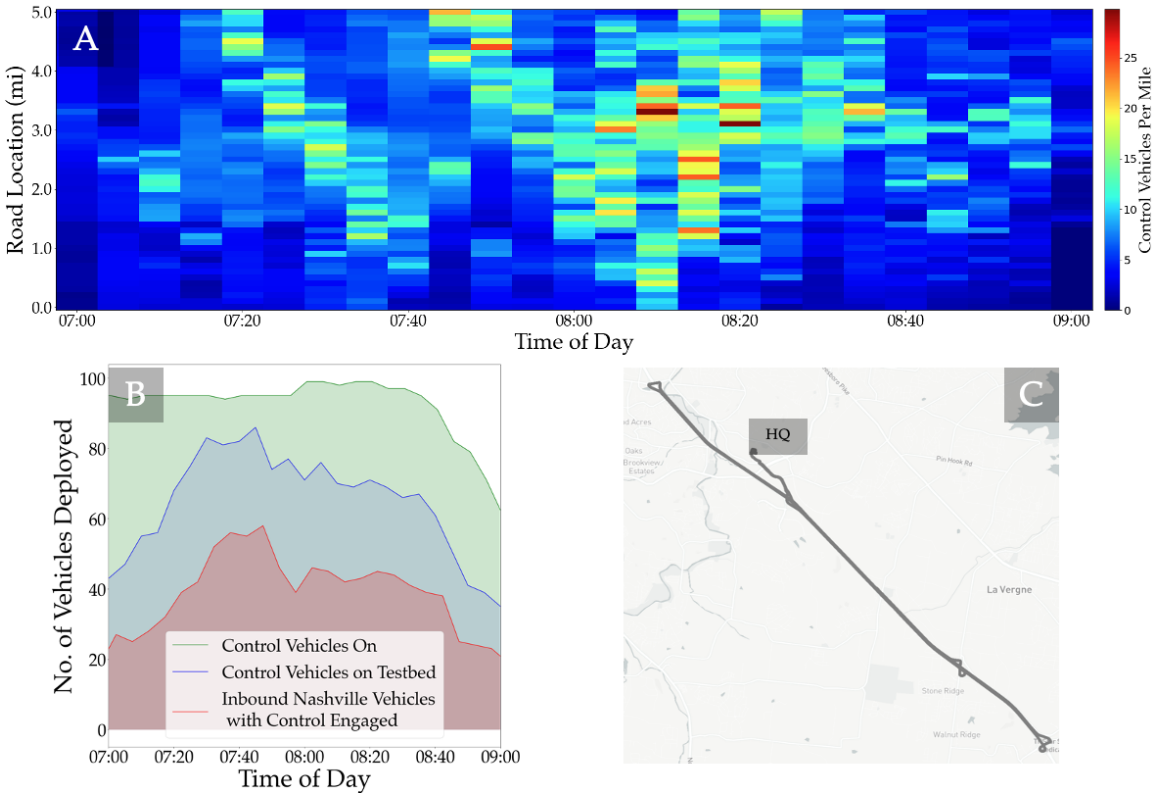


Figure 5.3: **Control Fleet Deployment Examined.** The control vehicle density (**A**) is created by calculating the density of westbound (inbound Nashville) vehicles in 528 ft (0.1 mile) and 5 minute windows. Counts of control vehicles (**B**) has three categories compared. A vehicle is considered "on" once the data starts recording. A vehicle is "on the testbed" if it is along the experimental corridor (not including turning around at exit/entrances to I-24). Inbound Nashville vehicles with control engaged are counted by the number of inbound (i.e. westbound I-24) vehicles within a 5 minute window that have control engaged. The partially overlapping looping routes driven by control vehicles, to support penetration rate and avoid overcrowding on/off ramps, are shown in (**C**).

is following at a standard ∼3s time gap is ∼300 feet away; that is 17.2 vehicles per mile. The local density of our control vehicles per mile reached a maximum above 26 vehicles per mile. Standing on the side of Interstate-24 observing this deployment, one would see an inbound Nashville control vehicle from our CAV fleet passing on average every 22 seconds. The precise density-effectiveness trade-offs of CAVs in mixed autonomy traffic control in real settings requires more research outside the scope of this work.

To understand our CAV deployment further, we can look at the status of all 100 vehicles in Figure 5.3B, and consider the maximums they reached. The total number of vehicles concurrently running our software reaches a maximum of 99, the number of vehicles on the experiment corridor concurrently reaches a maximum of 86, and the number of simultaneously active control vehicles in the inbound Nashville morning traffic reaches a maximum of 58, as shown in Figure 5.3B. The vehicles in the fleet operate in a loop, shown in Figure 5.3C; this tactic supports the control vehicle density, and therefore the effect in traffic, in being out-sized even compared to the large fleet.

5.2.3 A Scalable Connected Automated Vehicle.

Creating a scaled CAV fleet starts with creating a modification strategy for new connectivity and control in a single vehicle that is scalable. A series of significant pieces of hardware and software infrastructure are needed in order to extend a commodity vehicle into a connected automated vehicle. With regard to hardware: our custom wiring harness and printed circuit boards (PCB) create an electronic-to-digital bridge between a vehicle's physical sensing and actuation network and our embedded computer; the PCB, embedded computer, and battery uninterrupted power supply constitute the embedded computing complex (Figure 5.4 B-D). A custom wiring harness is necessary because unlike OBD-II interfaces, there is not a standardized connector and wiring layout for in-vehicle networks. A custom PCB was needed to overcome price constraints and scarcity in the embedded computer chip market, as well as provide the lean functionality we required. The need for lean principles in implementation cannot be overstated; simultaneously the specifications for a functional field test require a high level of robustness and agility in implementation.

To mediate the information sharing two-ways between the vehicle and experimental control, a suite of novel software tools were created or extended. Notably we extend earlier work: libpanda[97], a broad-scope package which houses data recording and vehicle control interfaces, and CAN-to-ROS[118], [128] a model-based code generative package providing a dynamic bridge between heterogeneous vehicle networks and the ROS framework for real-time sensor data and control. Other software components were created or extended to support and leverage information sharing from vehicle to infrastructure (critical for scientific understanding and execution of field testing) and vehicle to vehicle (enabling fleet collaboration, facilitated by central server proxy). These hardware and software pieces are discussed in greater detail in the materials and methods section of this work. The custom electronics, computing hardware, and software infrastructure coalesce to transform a commodity vehicle into a connected automated vehicle at less than 3% of the vehicle's retail price. This directly fills the technological gap to creating and deploying a large team of CAVs, capable of mixed autonomy traffic control.

5.3 Discussion

We have presented a new capability of automated vehicles: mixed autonomy traffic control. We introduce the tools and technologies for a scalable CAV fleet, validate them on a fleet of 100 vehicles, and use them on the first large scale deployment of mixed autonomy traffic control. The CAV fleet was deployed, reaching an average penetration rate of approximately 2.7%-3.6% per lane, during the week of November 14-18, 2022 on I-24 near Nashville, TN in heavy morning congestion. Our open scalable hardware and software platform enables experimental autonomy and connectivity in commercially available SAE level 1 and level 2 automated vehicles. Critically, our tools also enabled a rapid develop and deploy cycle. During the week, we tested multiple software versions on the fleet, enabling new controls to be run from one day to the next.

Our deployment of CAVs at scale differs from other field work in a fundamental way: scale. The scale of the robotic vehicle fleet changes the fundamental design of the physical platform, effecting everything upstream, including all software and control algorithms. Managing scale which is orders of magnitude larger than the baseline (one-off small-scale experimental automated vehicles of the past), is a challenge in itself. Scaling introduces a need for meta-automation, quality control, reproducibility, and monitoring software simply not necessary at small-scale. These features support the reliability of our fleet in adverse 'in the wild' field conditions. An issue previously handled in 1 minute becomes a 2 hour issue if approached with the same process. The constraints of scale met in this work forced costs (computational, financial, time) significantly

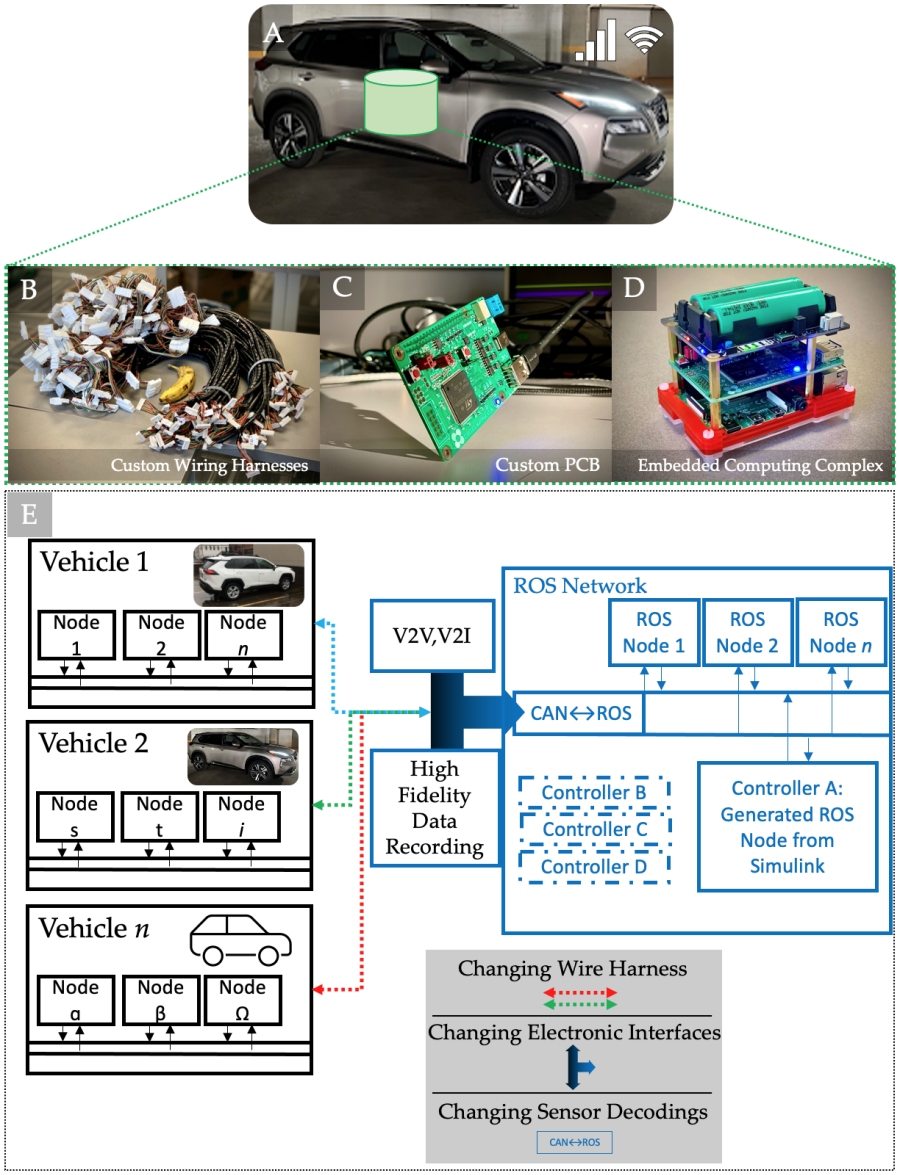


Figure 5.4: **Infrastructure of a Scalable Connected Automated Vehicle.** Overview of the hardware and software infrastructure within a vehicle (**A**). The key hardware components (**B-D**) are installed in each vehicle to bridge an extension of the stock vehicle electronic system to experimental control. (**E**) portrays the 2-way bridge between heterogeneous in-vehicle networks and the ROS framework. This allows the system to leverage existing infrastructure like Simulink ROS node generation for controller development, deployable immediately in the heterogeneous CAV team. Additionally, there is integration with inter-vehicle communication, and data recording, among other apps.

down on a per-vehicle basis. These efficiencies can be carried forward into small scale field research as well. The financial barrier to entry in fielding experimental control is scaled down two orders of magnitude, something appealing to researchers interested in developing translational technologies that make a societal impact.

This work opens up new frontiers in researching CAVs in the wild. There are several open questions about the potential effect of broad CAV adoption in real traffic flow conditions which can now be investigated, such as the effectiveness of varying controllers and their dependency of the road geometry, individual vehicle dynamics, and hyper-local traffic conditions, which all have non-trivial effects. This work allows us to shift from a somewhat myopic perspective of fielding automation in some vehicles, to a plenary perspective of changing the nature of traffic flow in the wild.

5.3.1 Possible Extensions

Future work will focus on extending deployment from days into months or years; the short deployment window is a limitation in this work. A long-term deployment could have expansive new capabilities, and would need augmentation of our current tools and technologies. An ongoing long-term fleet deployment could become a mixed autonomy traffic control version of the Robotarium [131], a remotely accessible multi-robot research testbed. A technical challenge in pursuing a long-term deployment is in privacy, security, and robustness. To address privacy in a long-term study, some preliminary work [132] shows how edge-based privacy can be implemented, where sensitive data is not recorded. Increased security is needed in a long-term deployments since there is less physical access to the vehicles by researchers, and is necessary to prevent tampering on distributed devices. The rigors of long-term deployments would demand greater reliability and robustness from our software systems. Consider when vehicles have limited networking connections: the current software assumes a regular wireless connection for an automated recording pipeline; without rectifying this assumption major failures could occur. Longitudinal studies in naturalistic driving vary in fleet size from around 10 [83], to 100 [84], and then 1000s [85]–[88] of deployed vehicles. They are not pointed at a specific research application or question *per se*. They aim to provide a valuable resource to the research community: volumes of naturalistic driving data from a broad scope of drivers, vehicles, and locations. Outside of the research community, vehicle OEMs collect data from their fleets of vehicles which are in the millions; this data is kept private for competitive advantage and regulatory reasons. A limitation of the large research data sets is that they cannot be used to evaluate new vehicle technologies that emerge. A long-term ongoing deployment of our tools opens up the time and capability to probe the numerous open questions on the effect of new and evolving CAV technologies on transportation systems.

Additional deployments using the tools in this work could introduce control algorithms to focus on objectives beyond traffic smoothing that are relevant to traffic engineers and society writ-large, including safety and throughput. Our technology backbone, created to field CAVs at scale, significantly lowers the barriers to fielding novel vehicle technologies in the wild in general. Already, distributed experimental sensing in the wild has been featured and adopted across domains: on roadways [75]–[78], in estuarial and riverine settings [79], pastures [80], and in production-scale fermentation processes [81]; distributed sensing combined with experimental control, shown here, is not yet pervasive. There is a history of experimental automated vehicle control at a small-scale [34], [61], [63], [65], [73], [74], [126], [133]. Instead of of pushing the ceiling of automation in vehicles higher, we look to test ideas involving distributed vehicle sensing with experimental control for improving safety and efficiency at a societal scale in the wild

Using this work as a basis, there are many research opportunities for fielding technologies in a full-scale

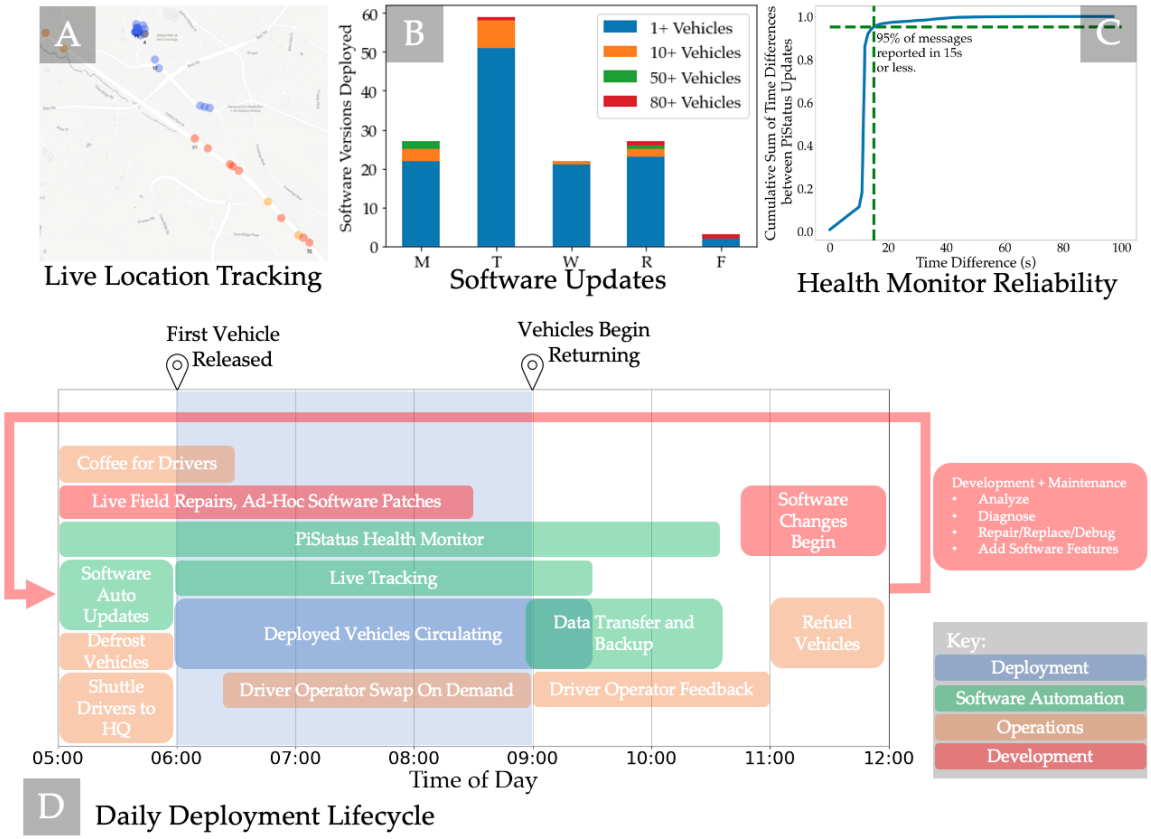


Figure 5.5: **Managing Scale in a 100 CAV Deployment.** Software features key to managing scale (**A-C**) are shown. Playback of (A) is featured in Movie 1. Many software versions (B) were deployed, with the most distributed ones being deployed in 10+, 50+, and 80+ vehicles. The daily lifecycle of a deployment is overviewed (**D**). Outside of the circulating vehicles, there is a collection of software automation, operations tasks, and iterative development which critically support the field deployment.

mixed autonomy setting. For example, investigating time-sensitive networking (TSN) between vehicles, where a guarantee on the delivery of information can be made. This guarantee could be used to augment safety assurances in car-following [89], and platooning [73], [74] settings. Versions of safe message passing have been derived, simulated, and tested at small-scale [45], [46], [134], but still need be fielded in a mixed autonomy full scale real world setting to address the sim-to-real gap.

5.4 Materials and Methods

5.4.1 System Architecture

This section provides an overview of the organization, design, and implementation of the individual vehicle platform. The system architecture is split into three categories: (i) vehicle layer, (ii) the 'bridge' layers (physical and software), and (iii) server layer. The *Vehicle Layer* includes the stock Advanced Driver Assistant System (ADAS) and original equipment manufacturer (OEM) sensor networks and electronic control. The vehicle layer meshes into the custom *Bridge Layers*. The Bridge involves several components of a *Hardware Bridge* and *Software Bridge* that extend the commodity vehicle into a connected automated vehicle (CAV) at

scale; the core function here is creating space for a fungible experimental controller to be plugged in. Finally the *Server Layer* is where individual vehicle information is aggregated and used in centralized processes, or passed directly to other vehicles; the Server Layer also serves an important role in managing scale.

Vehicle Layer

We needed to adapt a commodity vehicle in order to have a scalable CAV fleet. To push the costs of scale down, we aimed to maximally leverage existing sensors and actuation mechanisms to minimize the extent of our platform's footprint. In contrast, an automated driving installation of the past can cost multiples of the cost of the vehicle in sensors and compute [61], [63]. Recent AV field work in eco-driving[133], using a similar CAN injection approach to control, uses embedded computing equipment that retails on the order of tens of thousands in USD (unacceptably expensive for our constraints). A stock vehicle needs several added abilities in order to become part of our CAV fleet. The most important are an interface with vehicle sensor data, and interface with vehicle control system, and scientific quality data collection.

A given vehicle's underlying technological systems can vary widely. The networking on the vehicle layer may use CAN, CAN FD, Flexray, Automotive ethernet, or various other communication protocols depending on the implementation of the specific make, model, model year, and trim. Communication may vary based on subsystem in the vehicle. Regardless of the implementation details, it is critical to have real-time high frequency sensor information from the vehicle in order for it to be used as a live mobile sensor, for experimental control actuation, and for scientific data value. The vehicle layer is endowed with these enhanced features through the hardware and software bridge.

Hardware Bridge Layer

The Hardware Bridge creates interfaces between the three different software-driven entities (in-vehicle network, Software Bridge, Server Layer) of the system architecture. The hardware pieces marry natively independent systems to form our cohesive densely interconnected platform. Hardware interfaces are needed for reading on-board sensor data, for providing inputs to control algorithms, for sending control commands, for scientific data management, and system health monitoring. The most essential and cumbersome task for the Hardware Bridge is to provide a two-way physical translation between the electrical signals from vehicle sensors and digital signals used in the Software Bridge; this is achieved with our custom PCB and wire harnesses which in combination provide the two-way physical translation. The implemented Hardware Bridge consists of a custom printed circuit board (PCB) informally referred to as the MattHAT, a custom wiring harness cable for the vehicle CAN interface, a battery-powered uninterrupted power supply (UPS), and a 4G and wifi compatible radio for networking interfaces. The former two components are discussed next.

MattHAT The MattHAT has three keystone functions within the system architecture. It translates physical CAN electrical signals with transceivers into digital signals to pass off to libpanda [97] to handle; it uses an onboard digital potentiometer to provide an interface with the buttons on the Nissan Rogue cruise control system; and a relay switch to cut the physical CAN connection and inject commands into the CAN bus when needed. The device is compatible with CAN and CAN-FD, and OBD-II protocols. These features combined allowed for the MattHAT to be compatible with all vehicles in our heterogeneous fleet, and prospectively with many more vehicle platforms. The MattHAT is based on a 144-pin 32-bit STMH723ZGTb microcontroller, chosen due to its three on-board CAN FD hardware peripherals. While the MattHAT plugs into the Raspberry Pi's General Purpose Input Output (GPIO) header, USB is used as the primary connection for CAN

communication due the support of CAN over USB in libpanda. The MattHAT features a few jumper selections for power selection and for initiating the firmware boot of the microcontroller. The MattHAT's digital potentiometer is an entirely separate circuit designed into the PCB. This circuit connects the Raspberry Pi GPIO header over the Serial Peripheral Interface (SPI) supported by libpanda. The potentiometer connects to the vehicle's wire harness using a screw terminal and separate wire. A GPIO-controlled small relay is used to arm the digital potentiometer to prevent accidental voltage inputs during startup.

Custom CAN Cable To gain access to vehicles' onboard sensors and pass the physical signals to/from the MattHAT, you need an access point. The CAN cables were required to carry out the encoded system state as reported on CAN (e.g. velocity, steering wheel position) and send control commands into the vehicle. Local in-vehicle sensor networks carry all information for vehicle use including the radar and sonar sensors, odometer, and even window motor control. It is the closest piece of the Hardware Bridge to the stock vehicle in the two-way pipeline. Incoming information travels from vehicle sensors to control algorithms, and then outgoing from those algorithms through the CAN cable to the vehicle for commanding control.

Unlike the higher order layers of the system architecture, this piece of the system architecture was informed by vehicle layer detail – connector pieces, pin locations, pin type, crimping tools, wiring insulation, twisted pair combinations, and wire length. There are several points of access on a given vehicle, with several different protocols, a myriad of partially overlapping sensor and system-state data, which vary widely from vehicle to vehicle. Our implementation ultimately relied on developing new approaches to parameter estimation in cyber-physical systems [135]. This made it possible to discern the critical sensor data from the noise and impertinent information. With no off-the-shelf solution, it took several hundred hours to finalize the design of the cable, and several hundred more hours to manufacture the next 100 cables.

Software Bridge Layer

This layer of the system architecture sits between the Server Layer and the Hardware Bridge. The libpanda package [97] and the CAN to ROS tool [128], which bridges in-vehicle (nominally CAN) and ROS networks, serve as the pipeline between the Hardware Bridge and the rest of the Software Bridge. The cumbersome heterogeneity of different vehicles are abstracted away by the self-configuration and code generation features of CAN to ROS to shield the upper software layers from the lower level complexity and heterogeneity in the hardware. Our Linux systemd services contribute in several different categories, acting as managers and organizers. They operate the data pipeline between the embedded devices and the Server Layer, and also manage and organize the embedded software and hardware systems. The ROS Layer of the Software Bridge is a message passing network layer to process, record, and distribute system information, including housing fungible vehicle control algorithms.

Libpanda Our software package[97] was used to create vehicle interfaces. It governed the use of the custom PCB for the data interfaces between the vehicle layer and the Software Bridge, and it managed the control interface with state-based models to actuate based on requests from the controller node in the ROS layer of the Software Bridge.

Knowing that driver operators may have unexpected behaviors, and with safety as the first priority, libpanda leaves the stock safety systems intact and emulates the stock cruise control system to maximize safety and comfort despite being an experiment control system. We created supervisory control models in libpanda,

translating control requests from the ROS layer into vehicle actuation, accounting for inevitable issues arising from control design integration and adverse field conditions. A control heartbeat guaranteed either fresh experimental control liveness, or stock OEM control. A whitelist from the Server Layer required central permissions in order to execute experimental control on a given vehicle. These guards held in place by libpanda work to maximize safety and performance in the field.

CAN to ROS This software package [128] is a two-way interface between the in-vehicle network and the ROS network. In tandem with libpanda, it creates an interface from ROS to the vehicle directly. Its model-based code generation allows for automated self-configuration at runtime enabling real-time sensing in ROS on heterogeneous vehicle systems.

Systemd services Several Linux Systemd services were created to manage the embedded software systems. These software services were the mechanism by which we were able to effectuate automatic software updates, automated data management (recording and transfer), live tracking and system status monitoring, and communications with the server layer for control planning.

ROS Layer As a result of the CAN to ROS package, the ROS framework layer provides an approachable software space for non-domain experts to run experimental control algorithms. Vehicle interfaces for control and data are abstracted away, and what remains is a developer-friendly environment. We tutored control designers to integrate their control software in our preferred method: use a provided Simulink model of the Software Bridge, edit the controller node, and generate the formally verified ROS code. Alternatively control designers could write their controller manually in a ROS node. Either way, they were able to test their implementation with software-in-the-loop on-demand. We created a remotely accessible set of embedded hardware testing kits, each consisting of an embedded computer running the software bridge, substituting live data from the hardware bridge with pre-recorded sensor data played back through ROS. Controller designers were given remote SSH access to these kits to perform test runs with their new controllers, and compare the controller output on the pre-recorded data with their expected output.

Server Layer

Individual vehicles interface with the centralized server layer via the embedded compute. Each vehicle sends up local information and receives non-local information. Sometimes, local information sent up is for system health and monitoring such as the live tracking and health monitoring 'PiStatus' applications, which are critical for scalability. In applications such as latent networking and control heartbeat an exchange is made between the server and individual vehicle to inform live longitudinal control of the vehicle on the open road.

Latent Networking Latent networking created the opportunity to build control whitelisting, a control heartbeat, and live connectivity between vehicles. Control whitelisting created a centralized power to enable or disable a vehicle's experimental control live. In case of emergency, this feature assures some operational control from the field HQ. The control heartbeat ensures liveness of connectivity from the individual vehicle for experimental control to be enabled, otherwise stock systems would be in control. This way, if for any reason a vehicle does not have live connectivity the local safety is supported. Live connectivity between vehicles was leveraged to input empirical measurements from CAN data from some vehicles into the control algorithms

decisions for other vehicles. By sharing information through a central portal, traffic conditions in areas relevant to any given 'ego' vehicle were available agnostic to proximity or fixed infrastructure. Experimental control implementations were empowered to decide which sets of information are or are not pertinent.

Control Heartbeat Libpanda was redesigned for a heterogeneous fleet with vehicle interface abstraction in mind, extending the previous functionality of message rate checking needed for Toyota control. The control heartbeat served as a safety measure to disconnect control messages if no service is regularly sending commands, e.g. when a control command node process crashes on the embedded compute. This was initially built out for control in Toyota vehicles, because failing to send commands at a high enough rate will cause manufacturer modules to error for safety reasons. Libpanda had therefore been designed to check the command rate and disable control both gracefully and safely. This functionality was extended to work across all supported vehicles, which have varying liveness constraints before triggering errors in manufacturer modules. The heartbeat is also a guard against the liveness limitations of experimental algorithms. Libpanda connects the control heartbeat into ROS, so it functions as a time-dependent request. By exposing this information to the ROS layer, any higher level controllers or services can request controls to be enabled. If these higher-level functions stall or crash then libpanda will relinquish spoofing, resulting in the vehicles to operating like normal stock systems.

5.4.2 Managing Scale

At its core, this work deals with combatting issues that arose from efforts to scale. Societal scale problems need large scale studies to investigate their solutions, but implementation at unprecedented scale in this case requires new tools. We built tools to track live deployment of vehicle controllers, enable large-scale deployment of the electronics hardware bridge, monitor CAV system health in real-time, manage distributed software versioning, and manage data collection and transfer. These tools are necessary when fielding a large CAV fleet as a research instrument. Where automation could not be of use, we leveraged process, structure, and organization to manage complexity and performance.

The CAV fleet tools and technologies were developed with maximal modularity; this promotes co-development and adaptability. Co-development was very useful in development phases, to have many smaller components be pushed along by different team members and to maximize 'uptime' when waiting for components with supply-chain constraints. Agility and flexibility are critical in large-scale field tests, as the tools and technologies run into unpredictable and chaotic conditions, and adaptability makes or breaks the ultimate functionality. We were faced with a handful of technology failures which we did not encounter in development and testing, and were saved by the adaptability built into our systems.

Live Tracking A critical component of understanding the status of the 100 CAV team was a live tracking map, showing each vehicle and its status in real-time on a road map. We used this to understand and monitor the distribution of vehicle team within the traffic flow, the liveness of vehicle connectivity spatially throughout the testbed, which vehicles were running experimental control, their travel direction, and their velocity. This tool is made to be leveraged by users unfamiliar with the construction of the CAV scaling platform, and lower level implementations. Control designers could use this tool to rapidly identify potential problems with novel controllers, and the impact of traffic congestion on expected test vehicle performance. It also enabled personnel teams to see at a glance how many drivers were out on the road, their location to anticipate a driver swap, and which drivers may need help. The live tracker is implemented with two pieces of software: a

live-tracker ROS node running in the vehicle ROS layer, and a light-weight mapbox-based webapp. The ROS node communicates at 1 Hz with the server the following: (1) VIN (to identify the car and driver), (2) GPS coordinates and time, (3) Ego car velocity, acceleration, and control system status, (4) Whether the vehicle is moving westbound on the I-24 highway route or not. This data, when it hits the server, is promptly stored into a mySQL database for rapid retrieval. The Mapbox-based webapp contacts the server via a REST API call that returns the latest information from the database, and updates this every 15 seconds.

Hardware Bridge at Scale Hardware scale introduces problems in the field. A task which takes 1 minute for a single vehicle will take about 2 hours for 100 vehicles. A task which takes 1 minute for a domain expert will take much longer and be performed with less accuracy by a non-domain staff member. To create a fleet of 100 CAVs, we needed staff outside the domain to do critical work in order for the system to be operational. We worked to make each task lean, easy, and intuitive. These tasks could then be monitored at scale by domain experts to focus attention on which vehicles need more work, and which are proper nominal state.

Here we provide one detailed example to elucidate the broader thrust: We created a device to test CAN Cable quality. It consisted of the cut ends of a fully assembled CAN cable soldered to a microcontroller to systematically check the correctness of electrical connections on the handmade cables. A failure in a CAN cable has especially high cost because it requires disassembly of a vehicle to replace (see vehicle installations in Movie 1), and the production of each custom cable takes hours. When the quality control device was plugged into a computer, a display showed the correctness of the plugged in cable in a color coded wiring diagram. A spare 'Gold Standard' cable was made for reference to assure the cable tester was correct. This automated quality control saved countless errors in cable manufacture, and resulted in zero failures from the CAN cables in the field.

The preparation for hardware at scale affected several other processes for field testing, both through logistics and automation. Before the fleet of 100 stock vehicles was available, we sourced several tool sets for disassembly and reassembly, documented and tested our processes for dissemination to non-domain staff, and to close our field work we were able to disassemble and uninstall our system within 2 days to return vehicles in factory-state to the manufacturer. Flashing the firmware on a MattHAT PCB was semi-automated so it could be done for all devices in under 2 hours; another example of how just making a 10 minute process 2 minutes saves days of work at scale. Since battery UPS could be overdrawn in field conditions (and cause system malfunctions if too low) the PiStatus monitor added live reporting on the battery voltage *in situ*. This addition was seamless, enabled through automated software updates in the field. We could monitor the battery UPS voltage for each vehicle and by color-coded status proactively swap in batteries from a bank of fully charged ones.

PiStatus

In order to successfully process hardware installation, software installation and perpetual updates, system performance, and vehicle-by-vehicle status in the field, a live monitoring system is a necessity. This monitor also led to vehicle issue diagnoses with cables, software bugs, and more in the field. This software service, PiStatus, including its server-side database counterpart, allowed for real time monitoring of the health granular enough for each installed component, and broad enough for the entire CAV team functionality.

In implementation, PiStatus is a novel systemd service for self-reporting vehicles' status. Bash scripting and other interconnected systemd services provided the local information from each vehicle to be sent to the server-side database 5 times per minute; from this database, several php webpages were created to display

and organize this information. Small details, such as color coding the information displayed on the site, made it possible for a holistic understanding of the distributed system in the blink of an eye.

Software Updates

Automatically deploying software upgrades is an essential function of the fleet as a research instrument. Foremost, this is the critical difference between deploying an algorithm vs. any algorithm for mixed autonomy traffic control. Second, the inevitability of software failures must be planned for, since a complete fidelity simulator for the physical deployment of CAVs does not exist. A control designer without domain expertise in robotics, automation, or vehicle systems, must be able to create a new or updated control system and deploy it for testing immediately, otherwise there is no realistic chance for the fleet to be properly calibrated and controlled in the critical time window of field testing. To collapse the development/deployment feedback cycle we needed a mechanism to update controllers systematically and on-demand. Our design choice was for the embedded device to look for an update when powered on, and install updates immediately if there is an update to be made. Each update includes a prescribed set of software version hashes to all repositories' versions to be used. There were multiple of these sets of hashes, assigned by VIN, to allow for running multiple versions of control simultaneously in the fleet. By creating the ability to vary software version on a vehicle-by-vehicle basis, we created the opportunity for partitions of the fleet to be tested A/B and for unstable changes to be hardened. Software updates were managed through git, with some critical changes happening down to the minute with coordinated radio calls, and others on a scheduled basis. As a practical matter, software updates were managed by closing access to the libpanda repository in the organization to all but the core team. A mismatch between a new commit on the master branch and the local version triggered updates (pull and rebuild). To ensure beyond a doubt that systems were up to the minute with bug fixes, we often had to 'boot' all vehicles. Since we created a software version monitor in PiStatus, we could confirm a vehicle's software version before letting them drive. The update attempt would occur without this redundant check, but monitoring gave assurance that no runtime issues prevented the software installations. Updates occurred ad-hoc during testing, at night to prepare for the next day, the morning of, and in the afternoons to test new software in small-to-medium scale tests.

Data Management

Scientific Data Quality The vehicles are collecting high fidelity information about their state at 20+ Hz. This richness is critical for applications which need to understand dynamics such as controls, traffic engineering and micro-simulation, and artificial intelligence research [18], [89]. The archival data collection enables analyses on the effectiveness of each different controller fielded, and several other research questions of interest. These include: studying the effects of the vehicle operator as a human-in-the-loop; analyzing the effectiveness of software upgrades; or collected data used as ground truth to compare to camera tracked trajectories. There already exists an open sourced tool for analysis of CAN data[130] which makes CAN analysis approachable. This first deployment of mixed autonomy traffic control has generated 22752 miles of driving data over a combined 1022 hours, a novel and valuable data archive.

Automated Data Transfer Data collection is part of the automated startup of the operating system on boot. We collect CAN data to capture the in-vehicle network in high fidelity, GPS data to capture location and velocity redundantly, rosbag files to capture the ROS network traffic, system logs, and launch environment

JSON files for post-hoc analyses. A total of 13.7 Terabytes of data were collected over 5 days of driving. Over 1.7 Terabytes of which was vehicle CAN data, GPS, and ROS bagfiles. The remaining 12 Terabytes are dashcam video data. We used one private/public key pair embedded in a locally distributed OS copy to allow each RPi device access to a user account for transferring data to our local server. We set up a hardened wireless access point in the secure parking area for the fleet, spliced from optical fibers of our operational hub. We also created a tool to monitor the data transfer live at scale from the server-side. On the hour, these data were synced to the Cyverse[136] data store in the cloud. Having the data recorded and uploaded automatically saved countless hours of work to assure start/stop of records and manually transfer the data. Since the data are so critical for the controller developers to examine the results of their algorithms' deployment, it was useful to know that the data were in fact in hand, properly recorded and backed up automatically. Furthermore, because of the dynamic nature of our system, data were examined in the morning of the field deployments and new controllers were being tested hours later for the next days 100 CAV fleet deployment. Fresh data informed critical field decision-making. This capability was enabled in part through automated data transfer.

Repairs and Technology Failures

It was a surprise how vital our PiStatus tool, the live system monitor, was to the success of the field deployment in dealing with repairs and technology failures. We would have not been able to successfully install, maintain, and update all of the cables and embedded devices on all of the vehicles without a live and continuous monitoring tool detailing the state of the electronics and software. Some third-party hardware failed or failed to perform at times in the field. We observed inconsistent errors from SD cards, resulting in empty binaries or non-bootable operating systems, and some third-party electronics with subpar quality simply failed. Another hurdle was loose or unplugged cable connections, as there were several connection points per vehicle so several hundred opportunities for a loose or incomplete connection. Some missed cable connections were a result of the rapid disassembly/reassembly of the vehicle, causing temporary OEM system failures (e.g. parking brake non-functional). Each cable connection is a single-point failure for a vehicle's CAV functionality, so 100% compliance was necessary.

Some failures were triggered only once the large-scale field testing environment was introduced. For example, the UPS boards worked without issue in testing for auto-on functionality when external power supplied. We did not anticipate the high volume that the vehicles were going to be power cycling (from defrosting unseasonable ice, updating software at the last minute, field staff testing system functions). This revealed a quirk in the UPS boards where a capacitor in the 'Auto-On' circuit had not yet discharged when the vehicle was power cycled, rendering a conditional failure in auto-on functionality.

Part III

Expanding Testbed Interfaces

6 | SAILing CAVs: Speed-Adaptive Infrastructure-Linked Connected & Automated Vehicles

> If we could change ourselves, the tendencies in the world would also change. As a man changes his own nature, so does the attitude of the world change towards him. ... We need not wait to see what others do.
>
> Mahatma Gandhi

This section includes material from a publication:
M. Nice, M. Bunting, G. Gunter, *et al.*, "Sailing cavs: Speed-adaptive infrastructure-linked connected and automated vehicles," *arXiv preprint arXiv:2310.06931*, 2023

6.1 Introduction

This chapter introduces a novel capability for *connected automated vehicles* (CAVs) in freeway settings: dynamically adjusting a vehicle's *adaptive cruise control* (ACC) system to adhere to an infrastructure-based variable speed limit system using LTE communication. We deploy this capability on a vehicle in heavy traffic on Interstate 24 near Nashville, TN, which has a new active traffic management system including variable speed limits (Figure 6.1).

Infrastructure-based *variable speed limit* (VSL) systems have been deployed in several locations around the world over the past few decades [27], [137], [138]. These systems have the potential to increase public safety and mobility by adjusting the maximum speed limit to help smooth traffic flow.

One limitation of infrastructure-based VSL is that it requires drivers to comply with the posted speeds to maximize the benefits, which does not always occur in real deployments. Consequently, researchers have been interested in using a small fraction of automated vehicles to improve compliance of the overall traffic stream [94], and thereby increasing the benefits [28], [139]–[143]. Such works have largely been limited to simulation, due to the lack of widely available CAVs and communication gaps between the infrastructure operators the vehicle automaton systems.

The main contribution of this work is the development and field deployment of the first connected automated vehicle with the capability to follow publicly broadcast variable speed limits. We implement this capability with open source hardware and software that extends a stock vehicle's adaptive cruise control. We demonstrate the system on a vehicle (Figure 6.2) in heavy traffic on an open roadway, and compare the performance of the equipped vehicle to a human piloted vehicle driving in the same traffic.

Our work is enabled by the creation of a low-cost and scalable information pipeline from real public infrastructure into the automated driving of a vehicle. Our implemented system leverages the existing paradigm of the vehicle manufacturer's ACC. The desired speed is set by the driver on all roadways except the whitelisted area for the VSL system, wherein the desired speed is set by the VSL system. The nominal longitudinal control will match the vehicle's speed to the desired speed, unless intervention comes from the supervisory safety controller to facilitate safe car-following and stopping.

Figure 6.1: Movie 1: Variable speed limit (VSL) system activated and posting 30 mph speed limit during congested traffic. Photo taken from the ego vehicle which automatically adjusts its velocity to 30 mph in response to the VSL system. View a video associated with this work at: https://youtu.be/gFwJfEvnogI?si=BC_smYujNFJ2Adpj.

Our work builds on the work of [94], in which comparably high-cost CAVs are deployed on a congested freeway and adjust their desired speed based on information from 5 fixed roadside sensors. In contrast, our work deploys a low-cost CAV in tandem with active traffic management infrastructure posting a legally enforceable speed limit that is visible to all drivers. Beyond speed harmonization, connected vehicle field experimentation has occurred in application domains ranging from vehicle platooning[41], [73], to truck safety and efficiency[144], and demonstrating dedicated short-range communications[134], [145].

The remainder of this chapter is organized as follows. Section 6.2 describes the Interstate 24 (I-24) infrastructure and the software architectures which extend a stock vehicle into a VSL-following CAV; Section 6.3 characterizes the performance of our CAV deployment; and Section 6.4 discusses future directions extending from this work.

6.2 Methods

6.2.1 Active Traffic Management Infrastructure

The Interstate-24 SMART Corridor [146] is an *active traffic management* (ATM) system designed to improve safety and improve travel time variability, particularly during congestion caused by incidents that cause non-recurring congestion. The corridor consists of 28 miles of Interstate 24 (I-24) running in the east/west direction between Nashville and Murfeesboro, Tennessee, USA, along with a parallel section of State Route 1 (SR-1) between the two cities and connector roads between the Interstate and State Route.

The primary ATM strategies deployed to I-24 are variable speed limits and lane control systems, in the form of overhead gantries spanning the roadway and spaced every 0.5 miles. VSLs are posted at intervals

Figure 6.2: Control vehicle (right) preparing to deploy as a VSL-following CAV on I-24 behind pilot vehicle (left) for trajectory comparison.

of 1-minute in a range from 30 to 70 mph. Speed limits are chosen with the goal of reducing traffic speed variance, either through a tapered speed reduction algorithm, or through a human operator. The reduction of speed variance through the posted VSLs is intended to improve overall traffic safety. Speed limits are posted on LED message boards above each lane. Radar detection units measure the speed of traffic and are installed on average every 0.35 miles and report speed, occupancy, and volume measurements per lane aggregated over 30-second intervals. These measurements are used by the SMART Corridor traffic operations team in the calculation of posted VSLs.

Notably, not all vehicles need to exhibit strict compliance with the posted speed limit for the system to have a positive effect. A small number of vehicles complying with the speed limit has a greater *effective compliance rate* since non-complying vehicles have limited ability to maneuver around complying ones [10].

6.2.2 Vehicle System Architecture

Our control vehicle system is implemented on a 2020 Toyota Rav4 pictured in Figure 6.2. Figure 6.3 outlines the vehicle's software system implementation. Using low cost hardware, the vehicle system accesses the SMART Corridor data through a web-based pipeline over an LTE connection with a tethered mobile phone. Using this information, along with vehicle state propioception, vehicle control commands are created. We leverage the ROS message framework [98] for system design. Our system can be broken down into three categories: vehicle interfacing, VSL integration, and control design.

Vehicle Interfaces

Interfacing with the control vehicle was performed using the software libpanda[97]. Libpanda has the capability to firewall *controller area network* (CAN) messages between system modules designed by the *original equipment manufacturer* (OEM), allowing third party massages to replace OEM messages. Through libpanda, both on-board measurements can be read/recorded, and control commands can be sent to the vehicle. In tandem with with CAN interfacing, libpanda also interfaces with USB GPS modules to provide position

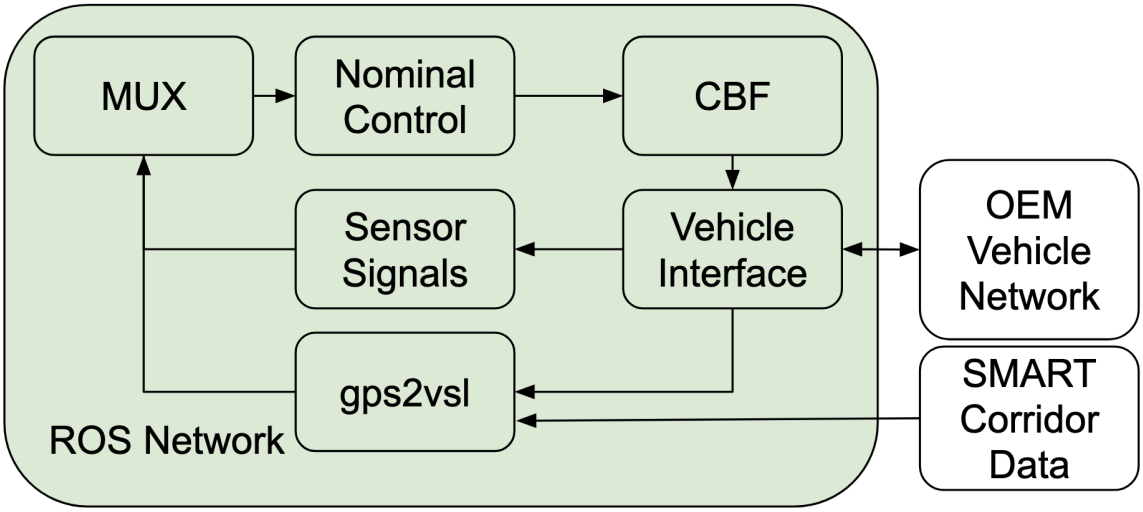


Figure 6.3: Three networks interact in the VSL-compliant CAV: SMART Corridor (fixed sensor network), OEM Vehicle Network (e.g. radar sensor, drive-by-wire), and the ROS message passing network. Vehicle proprioception, speed recommendations, and safety control actions are handled here.

information. Libpanda also keeps track of the OEM Advanced Driver Assistance System (ADAS) module state to prevent hardware-level errors when attempting to engage the system.

Code generation techniques as in [128] are used to convert manufacturer-specific vehicle CAN message into a homogenenous framework in ROS. In Figure 6.3, the vehicle interface represents a ROS node with an autogenerated CAN parser that produces sensor data like radar signals and cruise control setpoint. The vehicle interface node is a part of the can_to_ros project [118], exposing CAN-level vehicle systems to ROS.

Multiple setpoints, which are the desired maximum vehicle speeds, are expected to operate the vehicle based on the state of its pose with respect to the I-24 SMART Corridor. Orchestration of multiple speed setpoint sources is done with a combination of a multiplexer and a ramp function. This system has two sources of velocity setpoints. The first is the setting defined by the user as `/user_set_point`. The second is the posted speed limit provided by the VSL as `/vsl_set_point`. These can be selected based on the logic defined by the gps2vsl node for when the VSL setpoint is valid (see Section 6.2.3).

Whether setpoints are changed from multiplexing logic or from updates from the VSL, discrete jumps occur often on the order of $\approx 5\frac{m}{s}$ causing potential transient issues in velocity controllers. Such issues result in large commanded accelerations or decelerations leading to unsafe behaviors. To prevent this, a time-based ramp function is placed on the output of the multiplexer to smoothly transition the setpoint. The ramp effectively limits the rate of change the setpoint; in our implementation the setpoint was limited to $1.5\frac{m}{s}$ up and $2\frac{m}{s}$ down.

The ramp works well for when the system is engaged however the driver also has the ability to disengage the system, resulting in the vehicle greatly speed mismatching the setpoint. To prevent transients on system re-engagement, the vehicle's current measured speed from the CAN is provided as `/vel`, and is selected whenever libpanda reports that the system is disengaged. This lets the ramp closely follow the current speed, preventing memory issues that could lead to large setpoint jumps. Figure 6.4 shows the full structure of the multiplexer.

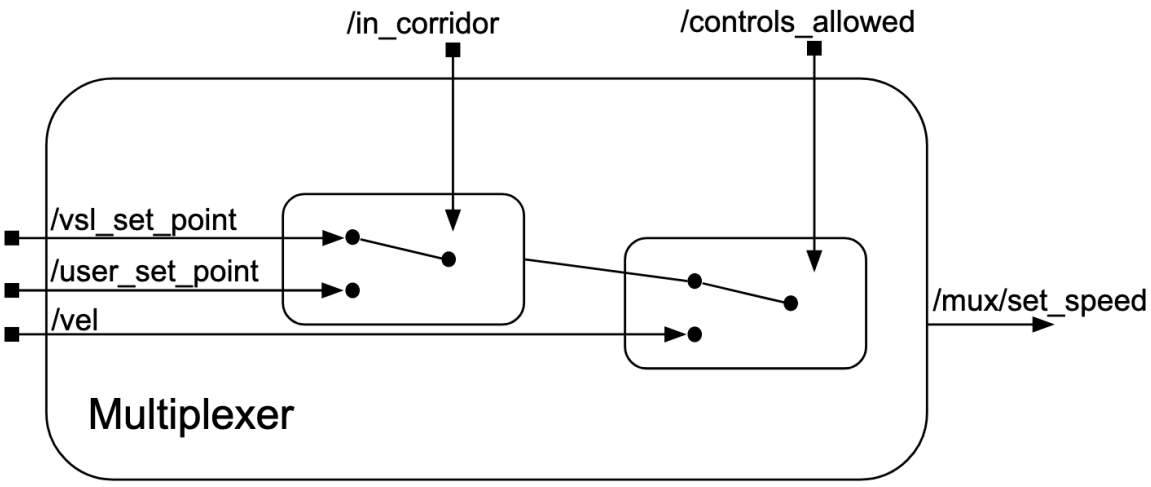


Figure 6.4: The multiplexer decides which signals are used for the desired velocity using two switches. If libpanda[97] is not allowing control, then the desired speed is output as the velocity signal; this feature avoids discontinuities in desired velocities in entry/exit states. If libpanda is allowing control, then the input from a second switch is passed through. In the second switch, if the vehicle is inside the SMART Corridor then the speed recommended by the VSL is the desired speed; otherwise (i.e. if you are anywhere else in the world), the desired speed is set to the set point on the driver dashboard controlled by steering wheel buttons.

6.2.3 Data Integration from Public Infrastructure

Messages are sent from the traffic operations center to each gantry whenever the speed limit should change. Our VSL data feed is obtained from a database mirror that records these messages from the traffic operations center to the gantries.

Web-based service to establish speed at each gantry

SMART Corridor data are made available through a three stage process: (1) mirroring of gantry updates into a database; (2) periodic server-side join of default data and gantry updates into a snapshot of VSL system data; and (3) on-demand web-based fetch of the most recent VSL system data.

A database mirror records all changes that are made by traffic operations to the VSL gantries. For efficiency at runtime, our system considers only changes made in the previous 24 hours when establishing the current speed limit. Thus it is necessary for the VSL system web pipeline to include both the default speed limit at this stretch of the roadway, and the most recent VSL posted speed if it has been updated within the past $T = 24$ hours.

The VSL system data are assembled every $15s$ and cached on the server for quick response when requested, rather than executing a query on request. Each row includes the default maximum speed limit for each gantry (in case there have been no traffic operation modifications in the previous window, T), and includes whether or not the VSL gantry is "triggered" meaning it is posting a speed limit lower than the maximum based on decisions by the SMART Corridor ATM system.

The on-demand web-based infrastructure is a single URL that returns the entire dataframe for use by the car. Although future work may determine that it is more feasible to return a subset of the dataframe, the current uncompressed size of $\approx 15kB$ means that the return sizes are negligible for a demonstration project.

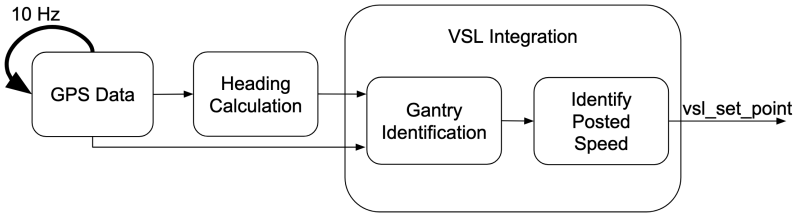


Figure 6.5: High level flowchart showing how our implementation starts with GPS data and produces a vehicle's desired velocity (also called a set point). See Figures 6.6 and 6.7 for more detail on the Gantry Identification and Identify Posted Speed components, respectively.

Models for gps2vsl

The models created for the automated VSL application take inputs from the GPS module, on-board vehicle CAN data, and the VSL system to output a useful desired velocity, or 'set speed', to the vehicle's experimental control algorithms. Broadly, this implementation results in a set-and-hold behavior by gantry as the CAV passes VSL gantries travelling through the SMART corridor. Desired speeds are fetched repeatedly to keep the vehicle's velocity tied closely to the posted speed limit as it changes in by location and time.

Figure 6.5 overviews the process to finding the relevant gantry (g_r) and the VSL posted speed (v_{g_r}) for the CAV desired speed, summarized as follows. With GPS location, we can identify the location of the vehicle (l) and it's heading (h). Using this information, we create a simple model and algorithm to identify the relevant VSL gantry g_r (if applicable) and identify the posted speed at the identified gantry v_{g_r}. To identify g_r, we define a polygon $p_{corridor}$ representing the set of locations where the SMART corridor lies, then proceed to calculating the state of the vehicle with respect to the VSL system (i.e. $l \subset p_{corridor}$? and $h == westbound$?), and send forward g_r when identified. Figure 6.6 features a model showing how this functions. As implemented, the CAV considers a gantry to be g_r when approaching it and crossing a 0.15 mile threshold. v_{g_r} is identified with a lookup triggered either with an event (like crossing the 0.15 mile threshold) when g_r is updated, or every 5 seconds. This allows the vehicle to react to changes in v_{g_r} from the same gantry over time, or changes to g_r while traveling down the freeway. Figure 6.7 summarizes this process of taking in the g_r when sent, and publishing v_{g_r} as the CAV desired speed. These components are tested with software-in-the-loop leveraging the default ROS[98] playback functionality with recorded drives containing trajectories in the SMART corridor.

6.2.4 Controllers

Here we describe the different control algorithms running on the experimental vehicle throughout testing.

Vehicular dynamics are controlled via a commanded acceleration value sent along the vehicle's CAN bus. A low-level control system implemented by the vehicular manufacturer converts this command to more specific vehicular dynamic commands (e.g. throttle,braking, engine). Let u_{cmd} refer to the acceleration command send to the low-level controller.

We create values of u_{cmd} through two higher level control algorithms. The first control law we refer to as the *nominal controller*, which calculates acceleration commands meant to track the desired speed as per the VSL system. Let u_{nom} refer to the acceleration coming from the nominal controller. Let v be the measured speed of the vehicle, and v_{g_r} be the desired speed from the relevant gantry in the VSL system. The acceleration from the nominal controller is calculated using a proportional control law of the following form:

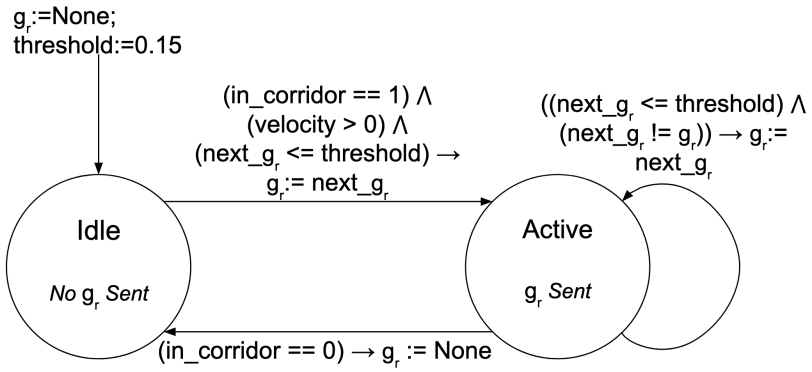


Figure 6.6: The Gantry Identification component has two states: Idle, where no gantry g_r is sent forward, and Active, where a VSL gantry g_r is sent on. When the system starts it enters as Idle. When conditions are met there is a transition to the Active state. The vehicle continues to set and hold g_r, sending it forward as well, until the vehicle leaves the SMART corridor (via freeway exit or out the ends of the instrumented freeway).

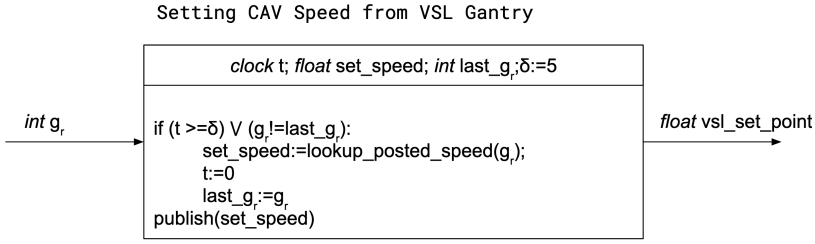


Figure 6.7: Identifying the v_{g_r} takes in a sent gantry value g_r and publishes a desired vehicle velocity in two ways. Either there has been an event where the g_r changes, which triggers a lookup request for the posted speed, or it has been 5 seconds since the last cached posted speed has been queried so a new lookup request is triggered. This allows the vehicle to react to changes in v_{g_r} from the same g_r, or changes in the g_r while traveling down the freeway.

$$u_{nom} = k_p \left(v_{g_r} - v \right) \tag{6.1}$$

where k_p is the proportional gain parameter. A value of 0.8 was used for k_p in experimentation.

In addition to the nominal controller, we use a *control barrier function* (CBF) as an *active safety filter* (ASF). Here we give a high-level description of the CBF used, for a more detailed description of developing CBFs as ASFs for vehicular control we direct the reader to [89], [147], [148]. Let u_{safe} be an acceleration command calculated using the CBF. Additionally, let s be the inter-vehicle spacing, and let v_l be speed of a preceding vehicle, both as measured by the vehicle's onboard radar system. Acceleration commands meant to supervise for safety are calculated as follows:

$$u_{safe} = \frac{k_{CBF}}{t_{min}} \left(s - \left(t_{min} v + s_{min} \right) \right) + \frac{1}{t_{min}} \left(v_l - v \right) \tag{6.2}$$

where $k_{CBF}, t_{min},$ and s_{min} are control parameters which we assign values of $0.1, 2.0,$ and 15.0 respectively. This CBF is designed to keep the vehicle's spacing-gap above a value of $t_{min} v + s_{min}$, where t_{min} is a minimum time-gap, while s_{min} is a minimum spacing-gap. k_{CBF} is a control gain parameter. This choice of safety is common [148], [149], but not the only possible choice.

To create a union between the nominal controller and the ASF we do the following:

$$u_{cmd} = \min \left(u_{nom}, u_{safe} \right) \tag{6.3}$$

which is interpretable as taking the smallest in value acceleration calculated by either the ASF or the nominal controller. This leads to overall system control which tracks velocity when safety is not a concern (not following another vehicle closely), but control that tracks for safety when needed.

6.2.5 Hardware Instrumentation

To extend a stock vehicle's hardware, we use a Raspberry Pi 4 board, a CAN interface board, a GPS module, connecting cables, and a battery-powered uninterrupted power supply. In total, the instrumentation cost comes under $500 USD. The embedded computer is situated underneath the passenger seat, and cables to connect to the ADAS module and GPS module are routed inconspicuously to allow for a unobstructed view for the driver operator of the vehicle. Mobile phones using LTE act as a mobile hotspot to provide internet connectivity for requesting the latest VSL setpoints.

6.2.6 Experimental Design

The experiment consisted of two vehicles which were released from a parking lot and directed to travel in the high occupancy vehicle lane (lane number 1 in the standard incident management lane numbering scheme). The speed of the first, or 'pilot', vehicle was regulated by the driver, who was instructed to travel at 70 mph (the maximum speed limit) when the posted VSL messages reported 70 mph. When the VSL gantries posted speeds less than 70 mph, the driver was instructed to drive at the prevailing speed of traffic. The speed of the second vehicle, or 'control vehicle', was regulated by the enhanced adaptive cruise control system developed in this work which dynamically adjusts the maximum speed of the vehicle based on the real-time variable speed limits. Both drivers were instructed to maintain a safe operating environment and to abandon the experiment if conditions on the roadway prevent a safe experiment from being executed. The experiments

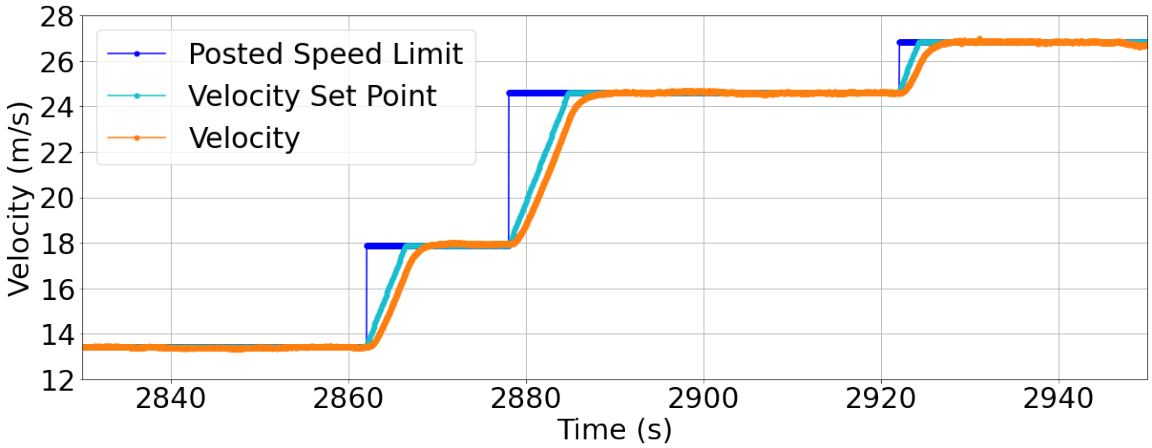


Figure 6.8: In the time domain, we can see the input response of the gantry, the ramping up, and then the response of the vehicle velocity.

were conducted in the 5:30-7:30 am window in which I-24 experiences the start of morning traffic conditions and regular traffic waves develop.

6.3 Results

We characterize the performance of our control vehicle with respect to its ability to comply with the posted variable speeds, and characterize the effect of this compliance compared to the downstream pilot vehicle in the traffic flow following the prevailing traffic speeds. This analysis is achieved in part by using the *strym* library[130].

6.3.1 Performance of CAV for VSL-Following

Figure 6.8 shows the performance of the CAV as the posted speed limits increase. The posted speed v_{g_r} increases (blue), the ramp filter (light blue) smooths the step up in velocity, and the measured velocity (orange) rises to remain pinned to the variable speed limit.

Figure 6.9 highlights the performance of the CAV while traveling down the freeway. This trajectory segment covers the three straightforward cases when transitioning to a new gantry g_r: an increased v_{g_r}, an equal v_{g_r}, or a decreased v_{g_r}. As the CAV crosses the threshold to transition to the gantry near mile marker 57.6 v_{g_r} is updated, and before reaching the gantry the desired velocity is reached. The posted speed limit v_{g_r} is the same at the second gantry met in the middle of Figure 6.9, so we see no velocity changes. When meeting the third gantry, v_{g_r} is read at the threshold approaching the gantry, and well before passing the sign the CAV is travelling at the new lower v_{g_r}.

The trajectory segment in Figure 6.10 is the same as in Figure 6.8 but new insights are revealed when plotting in the roadway coordinate domain. There are three steps up in velocity in this trajectory. Recall, we choose an implementation which sets-and-holds the gantry g_r, while continuing to listen to changes in the posted speed v_{g_r} which may occur until switching to the next gantry. The first step up past mile marker 64.2 reflects a change in the v_{g_r} from the g_r passed by earlier at mile marker 64.4. The second step up in velocity occurs in a straightforward manner – when crossing the threshold to the next gantry. Note that again the v_{g_r} is reached before passing under the gantry. The third step up shows that as we cross the threshold to the gantry

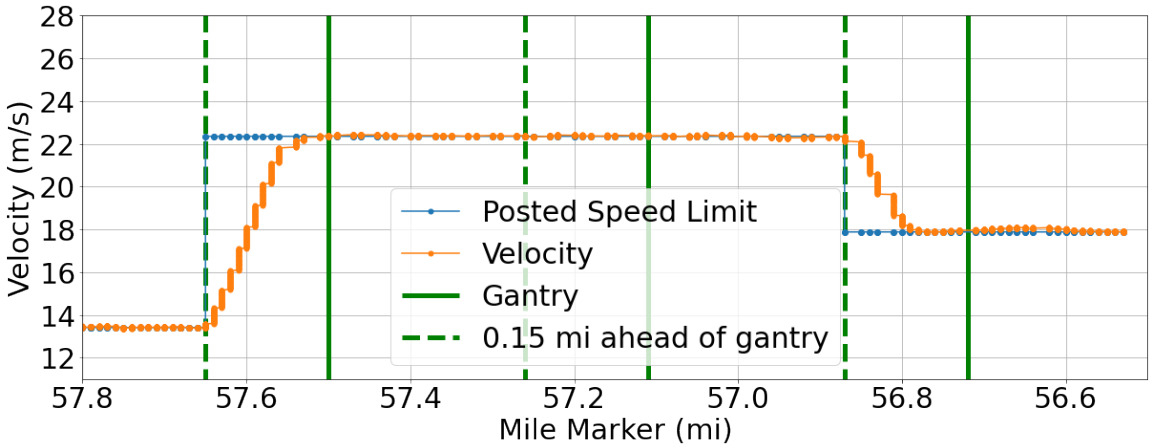


Figure 6.9: This is a figure showing how the vehicle is receiving new information starting at the pre-gantry dotted line, and the adjusting to the posted speed before reaching the gantry.

near mile marker 63.2, v_{g_r} has not changed. As the control vehicle continues to approach the g_r, the v_{g_r} and the CAV responds accordingly by increasing its velocity.

The control vehicle is not always able to cleanly follow velocity limits as posted by the VSL system. A heavily congested interstate highway is a complex environment in which safe car-following often takes priority over the ability to travel at the speed limit. Figure 6.11 features the mode switching in our multiplexed control system, which allows for smooth transitions between variable speed limit following and keeping track of safety.

The control vehicle should have a reasonable rise/fall time (interval from when a new v_{g_r} is ingested and the vehicle reaches v_{g_r}). Due to the small-scale of this deployment and the complex nature of the wild congested freeway environment, we report simple statistics on these events to claim generally reasonable behavior. There were 8 rising events in our experimental drives prompted from VSL system, ranging from a minimum rise time of 3.97 seconds to a maximum of 11.50 seconds, and with a mean of 6.21 seconds. The changes in v_{g_r} range from $2.24\frac{m}{s}$ to $8.9\frac{m}{s}$. There were 6 falling events, ranging from a minimum fall time of 5.21 seconds to a maximum of 8.08 seconds, and a mean of 6.79 seconds. The changes in v_{g_r} range from $-2.24\frac{m}{s}$ to $-4.47\frac{m}{s}$. This behavior is in line with the system design and implementation, and results in comfortable and reasonably prompt changes to the vehicle's velocity as dictated by the VSL system.

6.3.2 Potential Benefits

The lead (pilot) car and the control car (ego) left at nearly identical times, and drove in the same lane. Their speed plotted by the section of the roadway is shown in Figure 6.12, with overlay of the mean speed and standard deviation divided into three segments. As the standard deviation is normalized by the number of samples, it is roughly equivalent to the *normalized mean-squared error* (NMSE).

It is noteworthy that the average speed over this time is very similar in the first and last segment, with a larger average speed in the middle segment for the pilot car. However, the standard deviation of the ego vehicle is between 10%-25% lower than the standard deviation of the pilot vehicle, compared to its average speed in those areas. This is consistent with reductions found in [16], which indicates that there could be significant energy and safety benefits, and that there is motivation for additional study.

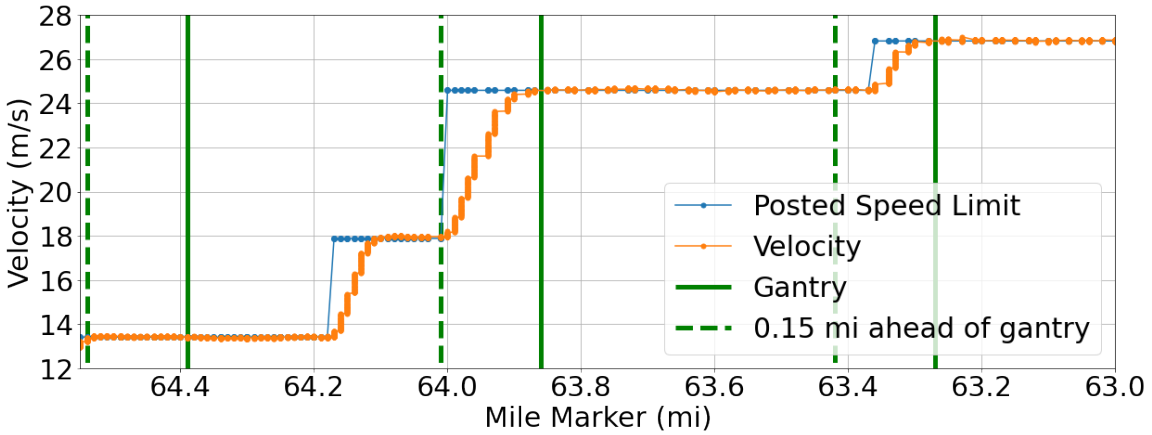


Figure 6.10: This figure shows further than Figure 6.9, that our implementation is a 'set and hold' at each passing gantry. The gantry near mm 64.4 increases the posted speed to 18 $\frac{m}{s}$ before we reach the next gantry, so we increase our speed. When reaching the next 'update zone' for gantry at mm 63.85, we update our velocity setting again.

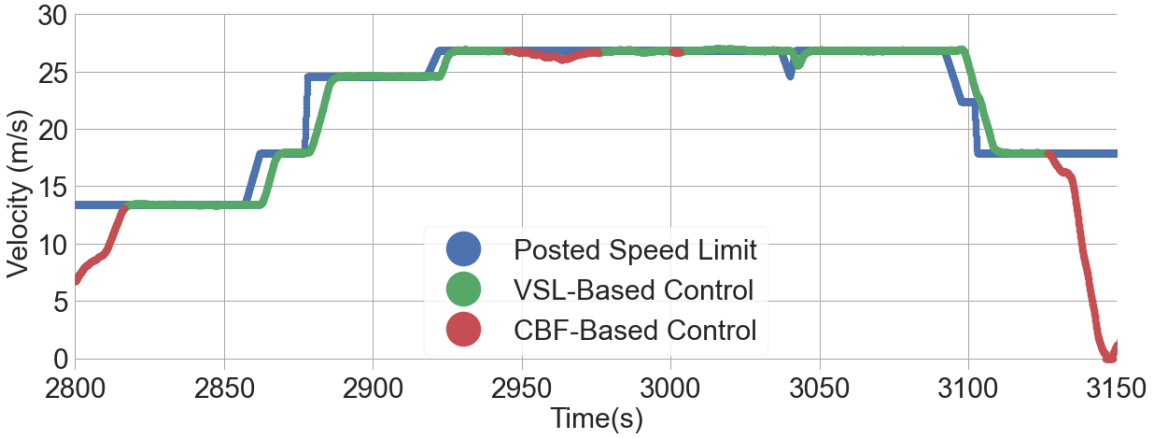


Figure 6.11: Multiplexed control states: while satisfying forward collision avoidance (control barrier), the vehicle software makes best efforts to match the posted speed limit and remain comfortable to ride in.

mm	Pilot (NMSE/Mean)	Ego (NMSE/Mean)
59.5	(6.361/7.938)	(5.041/7.955)
61.5	(7.203/10.410)	(6.002/7.822)
63.5	(5.582/7.315)	(5.212/7.302)

Table 6.1: Average speed and MSE for each car shown in Fig. 6.12.

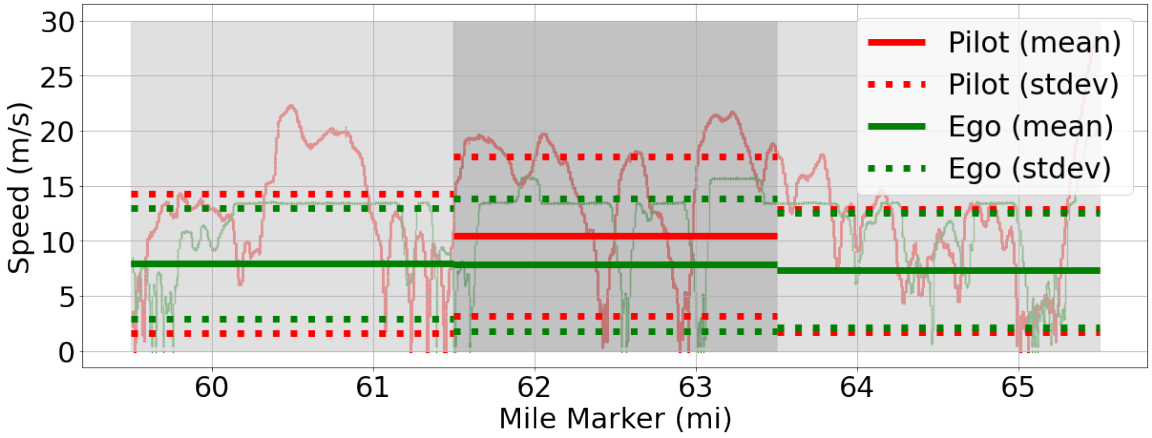


Figure 6.12: Less deviation is shown in the ego car than the non-controlled car, but average speed is very similar. Ego behavior is preferred when it comes to traffic smoothing, and increasing energy efficiency.

6.3.3 Practical Findings for Future Deployments

There are psychological and safety environment components which are tested by complying with the v_{g_r}. Nearby drivers are frustrated by going much slower than prevailing conditions, even though they are speeding into stopped traffic. It is more challenging to comply with v_{g_r} in an environment where prevailing speeds are greater than $10\frac{m}{s}$ faster, as we measured.

6.4 Conclusions and Future Work

Our work demonstrates it is possible to deploy CAVs with the capability to follow publicly broadcast variable speed limits at scale with LTE connectivity and affordable hardware. The field experiment shows strong performance of ADAS-equipped vehicles that react through connectivity to Variable Speed Limit systems in the field. The vehicle's reaction is correct with respect to specification within the geographical range (zone) of each VSL gantry, and responds as designed when a posted speed limit changes while moving within a gantry zone. The comparison between the pilot vehicle and control vehicles show nearly identical travel times and average speeds, but the variation in speed of the pilot vehicle is notably higher than the ADAS-equipped control vehicle.

The results go beyond simply driving at the posted speed limit and presenting the driving data. The impact includes an explicit cost that could enable this adoption at scale: less than $500 per vehicle, using connectivity over existing LTE networks to a tethered mobile phone without requiring 5G or high-speed radios.

This chapter demonstrated the feasibility and efficacy of the CAV platform. The traffic and safety analysis for this small-sized test show promising indicators, and motivate future work to scale the experiment to a larger number of vehicles in order to evaluate the efficacy of the system deployment on many vehicles simultaneously.

7 | *Via Media*: Fielding Connected Automated Vehicles to Follow Variable Speed Limits in Low-Compliance Regimes

It's all in the game.

Omar, The Wire (2002)

This section includes material from a publication:
M. Nice, G. Gunter, J. Ji, *et al.*, "A middle way to traffic enlightenment," *Proceedings of the ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPS)*, 2024

7.1 Introduction

Infrastructure-based freeway traffic control technologies are deployed on critical roadways to improve safety and mobility. Traditional systems include ramp metering to manage merging traffic onto the mainline, variable speed limit (VSL) systems that promote speed harmonization and reduce sudden slow-downs [27], [137], [138], and lane control systems that provide information about lane-closures ahead due to crashes. Recently, the widespread commercial deployment of level 1 and level 2 automated vehicles has opened new opportunities for freeway traffic control, for example to stabilize the overall flow when only a small fraction of vehicles are equipped [16], [94], [150]. Yet, today, the commercially available vehicle-based automation systems operate without coordination or cooperation with the infrastructure-based systems.

In this chapter, we consider the setting of cooperative variable speed limit control, in which connected and automated vehicles (CAVs) adjust their speed to follow the infrastructure based variable speed limit (VSL) system [93]. In fully automated traffic flows, the problem of collaborative control is purely technical, e.g., designing the sensing, communication, and control systems to enable vehicles to follow the posted speed limits. However, in mixed autonomy settings, a pressing safety challenge arises from the inherent disparity between vehicles programmed to strictly follow speed limits and human-driven vehicles that frequently exceed these limits. As a motivating example, we have recently observed prevailing traffic as much as 30 mph above the posted variable speed limit on a major US freeway shown in Figure 7.1. Large gaps between the speed of traffic and the posted speed limit occur regularly in daily traffic jams. Naïve automated control of the vehicle to follow the speed limit rather than synchronizing vehicle speeds with the prevailing traffic flow will create unsafe conditions to unexpecting vehicles under human control. Simply following the prevailing traffic flow ignores the opportunity with CAVs to increase safety and efficiency on roadways.

Here is the main problem addressed in this work: *How can we design a controller to follow variable speed limits when it can, while keeping up with the prevailing speeds when it needs to?*

We reason that automated vehicles must not drive substantially slower than human piloted vehicles if they are to be considered safe to operate in traffic and socially acceptable (and thus turned on, an obvious liveness constraint) by the owners of the equipped vehicles. This requirement to drive relative to the surrounding traffic creates new design challenges, given the timescales on which the traffic conditions change, and the inherent systematic latencies by many of today's commercial traffic information providers. These traffic state estimates provide updates on traffic conditions that are averaged in time and space, and have latencies in

Figure 7.1: **Recurring Dilemma**: Variable Speed Limit (VSL) gantry shows a 30 mph speed limit on Interstate-24. Prevailing traffic is shown to regularly exceed the VSL by a large margin. In this work we demonstrate an automated vehicle controller that follows the VSL on the gantry when nearby vehicles do, and adopts a higher speed when prevailing traffic is moving much faster than the posted speed.

excess of a minute or more.

The main contribution of this work is to design, implement, and field test a new cooperative automated vehicle control algorithm that complies with variable speed limits when other human drivers do, and blends in with human drivers when they violate the posted speeds. The specific contributions are:

- Introduction of a new notion of safety for cooperative automated vehicle applications to avoid causing controlled vehicles to drive substantially slower than surrounding traffic. Our approach recognizes the necessity for automated vehicles to adhere with the typical driving behavior observed on the roads, even if it requires a deviation from the posted speed limit. The control algorithm on the vehicle maintains collision avoidance through the use of a control barrier function-based safety filter, follows the posted speed limit when prevailing traffic is also operating near the speed limit, and exceeds the limit when prevailing traffic requires it to.
- Development of a vehicular-based method for measuring prevailing traffic. Specifically we decode Controller Area Network (CAN) messages on a commercially available level 2 vehicle corresponding to the onboard radar unit, and use the observed radar measurements to estimate the speed of nearby downstream vehicles. Since the measurement is done on the vehicle, we can maintain safety (accurate awareness of with surrounding traffic) locally, even if we lose communication to external data sources.
- Field experiments on two control vehicles operating in heavy morning rush hour traffic on the I-24 Freeway near Nashville, TN. We implement our controllers using low-cost hardware, to enable scalability of our approach. Our findings from the experiments show that we spend 16.6% of time following the variable speed limit, 24.0% of time above the speed limit due to prevailing traffic, and 59.4% of the time in a car-following mode to prevent forward collision.

The remainder of this article is organized as follows. In Section 7.2 we describe our control system. In Section 7.3 we review the experimental setup. Section 7.4 provides the findings from the field test of our controller operating in heavy traffic.

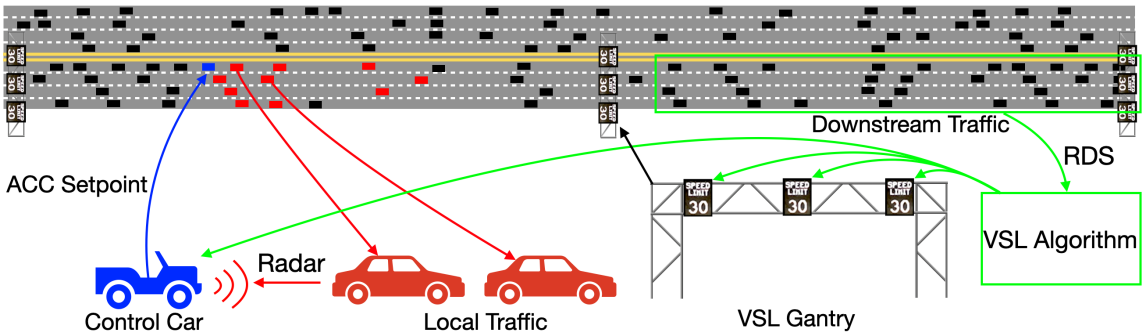


Figure 7.2: **Environment**: Overview of the control environment. The VSL system measures downstream traffic for aggregate traffic information. A control vehicle (blue) can measure timely information about highly local traffic in front and adjacent to the vehicle (red). Our controller changes the set speed based on information from both of these sources.

7.2 Methods

Our design challenge is to architect a vehicle speed controller that considers both the legally enforceable variable speed limit, and local traffic relative to the controlled vehicle. Figure 7.2 outlines the environment of the design involving a single control vehicle and two sources of traffic information. The controller acts as a replacement for the OEM Adaptive Cruise Control (ACC). The design integrates both downstream and local traffic conditions to switch into 5 different control modes:

- *Normal-Mode*: When not on a roadway with a VSL system and no vehicle is in front, then operate like a standard cruise controller.
- *VSL-Mode*: Drive at the speed setpoint provided by the VSL. This can occur if there is no traffic, or we do not meet the conditions to enter the other modes. This is a V2I interaction that is only be engaged while operating on a roadway with a VSL system.
- *Middleway-Mode*: If nearby traffic is driving much faster than the variable speed limit, then control the vehicle speed at a middle ground between the VSL speed and prevailing traffic. This is effectively driving in a reduced go-with-the-flow behavior.
- *CBF-Mode*: If a lead vehicle is in front and driving slower than the current speed setpoint then follow the leader in manner which will prevent collisions using a control barrier function (CBF). This mode overrides the other active modes at any time. This is similar to a stock ACC system.
- *Disengaged*: Control is inactive, driver has full control.

Normal-Mode is mutually exclusive to VSL-Mode and Middleway-Mode. When the vehicle is not on a roadway with a VSL system, the controller will only use the modes of the Normal-Mode or CBF-Mode to mimic the OEM ACC.

We will first describe the design of the controllers to acheive these modes, then describe the specific implementation including the location, vehicles, and vehicle hardware.

7.2.1 Controllers

Here we describe the different controllers and mode switching on the experimental vehicle throughout testing. Downstream traffic information is provided by the VSL system, which is primarily responsible for setting the variable speed limit on the gantries. Local traffic information is measured through onboard sensors on the

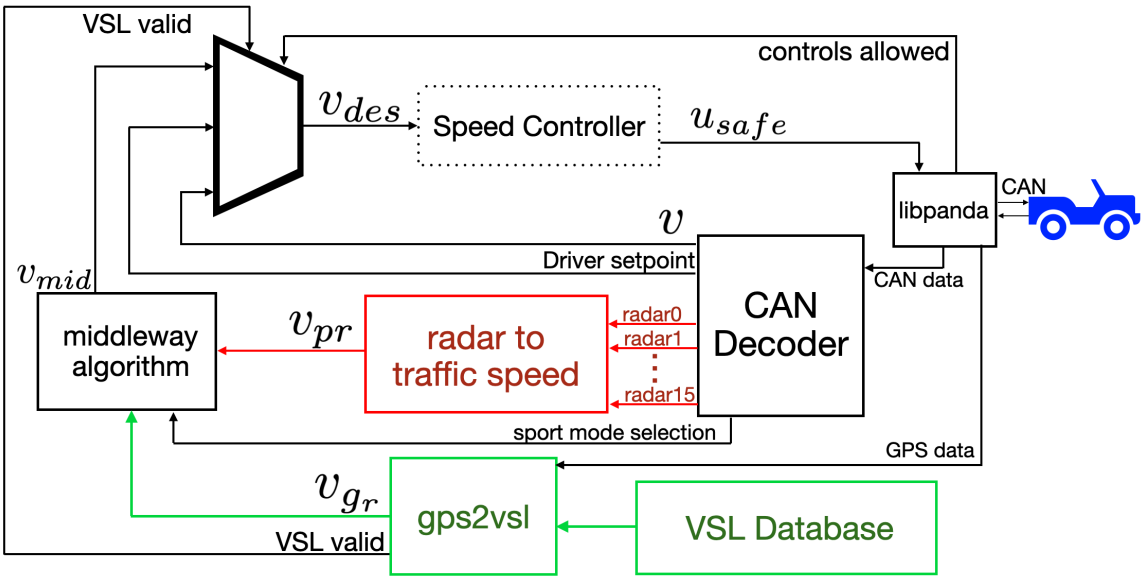


Figure 7.3: **Speed Selection**: The system architecture to determine the speed setting based on the VSL gantry, the state of the cruise controller, GPS information, and the radar data. The equation for the middleway algorithm is shown in equation 7.1.

car, such as radar. Information is fused from both sources to provide a speed setting for the controller vehicle. The design of the custom cruise controller is based on a hierarchy of a low-level speed and safety controller in tandem with higher-level speed setpoint selection algorithm. We start by looking at the design of the higher-level speed selection, then described the lower-level speed control.

Speed Selection

Figure 7.3 shows the design of selecting a speed setting for the speed controller. The speed setpoint is switched between three different sources through a multiplexer. The speed setting is either set to the current speed of the vehicle v (Disengaged), the driver's setpoint as set by the cruise controller interface, or the middleway algorithm that uses local and downstream traffic information v_{mid}. The setpoint is chosen based on the state of the vehicle and the location and direction of the vehicle.

- If the driver has not yet engaged the cruise controller then controls are not allowed (Disengaged), so the multiplexer sends the vehicle's current speed v as the controller setpoint. This is done to ensure that a smooth transition occurs when the driver engages the controller.
- If the controller is engaged but the vehicle is not in the VSL environment, then the multiplexer will switch the setpoint to the driver's setting from the gauge cluster (Normal-Mode/CBF-Mode).
- If the controller is engaged, vehicle is within the VSL region, and has a valid VSL reading, then the multiplexer switches to the setpoint provided by the middleway algorithm v_{mid} (VSL-Mode/Middleway-Mode/CBF-Mode).

The middleway algorithm shown in Figure 7.3 is defined as follows. Let v_{mid} be the output desired speed of the middleway algorithm, v_{g_r} be the recommended speed from the relevant gantry in the VSL system, v_{pr} be the average velocity of faster moving vehicles observed by the control vehicle's forward radar sensor, and $v_{des_{max}}$ be the highest allowable v_{mid}. v_{offset} is a runtime threshold parameter representing the how much slower than v_{pr} the vehicle operator is comfortable with, and is settable by the driver through the vehicle's

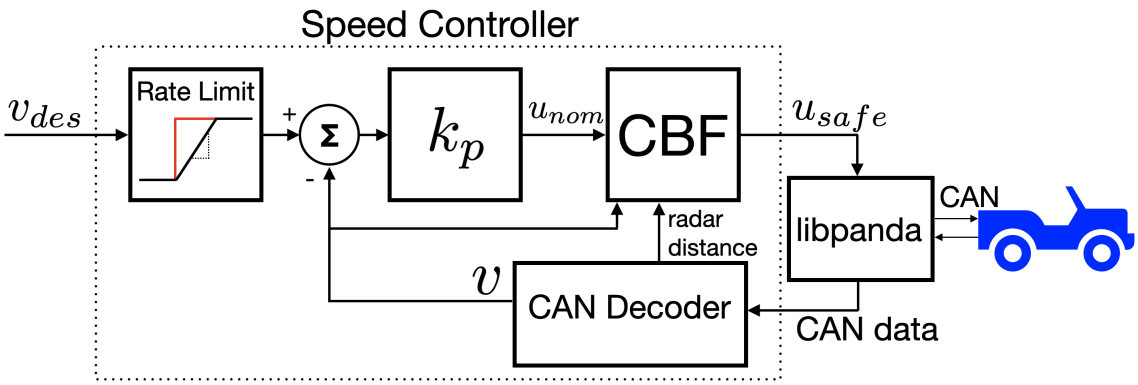


Figure 7.4: **Speed Controller**: this acceleration-based controller is a replacement for the OEM cruise controller. The controller takes an input speed setting, v_{des}, and sends acceleration commands to the vehicle through libpanda. The speed controller is based on rate limiting v_{des}, a nominal proportional controller, and a CBF to perform dynamic filtering to provide car following and prevent collisions.

Sport Mode and *Eco Mode* features. The desired speed is then calculated using a control law of the following form:

$$v_{mid} = \min(\max(v_{pr} - v_{offset}, v_{g_r}), v_{des_{max}}) \tag{7.1}$$

Note that the usage of the max function in (7.1) effectively encodes the mode switch between Middleway-Mode and VSL-Mode.

The local speed of traffic, v_{pr} is estimated when there are vehicles going faster than the ego vehicle. The estimate takes a point cloud of radar measurements from the last 5 seconds, and averages the observations from vehicles going faster. If there are not enough recent observations, the estimate is switched off (outputs 0) and the v_{mid} is consequently the posted VSL speed v_{g_r}.

Speed Controller

Figure 7.4 shows the low-level controller design. Vehicular dynamics are controlled via a commanded acceleration value sent along the vehicle's CAN bus using libpanda[97]. A low-level control system implemented by the vehicular manufacturer converts this command to more specific vehicular dynamic commands (e.g. throttle, braking, engine).

First, a time-based ramp function is applied to the v_{des} input of the nominal controller to produce v_{ramp}. This was designed for use cases when the setpoint may exhibit discrete jumps. Using a ramp function rate-limits the input and allows the setpoint to be changed without potentially unsafe transient effects feeding through to actuation. In our specific use case of dynamically changing the setpoint during the experiment the ramp function allows for switching between setpoints from different sources.

The vehicle acceleration request u is based on two control algorithms. The first is the control law we refer to as the *nominal controller*, which calculates acceleration commands meant to track the filtered desired speed v_{ramp}. Let u_{nom} refer to the acceleration coming from the nominal controller.

The acceleration from the nominal controller is calculated using a proportional control law of the following form:

$$u_{nom} = k_p \left(v_{ramp} - v \right) \tag{7.2}$$

where k_p is the proportional gain parameter, v_{ramp} is the rate-limited desired velocity, and v is the instantaneous velocity. in our specific implementation, a value of 0.8 was used for k_p in experimentation.

The CBF-Mode employs a low-level supervisory controller based on a control barrier function that overrides engaged controllers to avoid forward collisions. For a formal description of the design of this CBF, see [89], [147], [148]. The form of the controller is as follows:

$$u_{safe} = \frac{k_{CBF}}{t_{min}} \left(s - \left(t_{min} v + s_{min} \right) \right) + \frac{1}{t_{min}} \left(v_l - v \right) \tag{7.3}$$

where u_{safe} is the maximum allowable safe control acceleration, s the inter-vehicle spacing, and let v_l the speed of the lead vehicle immediately ahead. $k_{CBF}, t_{min},$ and s_{min} are control parameters which we assign values of $0.1, 2.0,$ and 15.0 respectively. This CBF is designed to filter control accelerations so that the vehicle's spacing-gap stays above a value of $t_{min} v + s_{min}$. This choice of safety is common [148], [149], but not unique. For example, this safety choice and design has been safely and effectively fielded in other open-road field tests, such as [20].

Controller behavior at scale

Even though our controller allows travelling above the posted VSL, if a series of vehicles run it, the traffic flow will approach VSL speeds or slower. Middleway-Mode is only needed as long as enough of the traffic flow continues to violate the speed limit, creating the scenario where following the law and maintaining safety by matching traffic flow conflict.

Consider a highway where all travelling vehicles are control vehicles where the penetration rate $p = 1$. There are two cases where traffic is not following the VSL speed v_{g_r}: either traffic is faster than v_{g_r}, or traffic is slower than v_{g_r}.

In the case where traffic is faster, consider a vehicle n that observes vehicles downstream of itself moving faster such that $v^n_{pr} - v^n_{offset} > v_{g_r}$. Vehicle n would travel tracking some speed $v^n_{pr} - v^n_{offset}$ slower than v^n_{pr}. Consequently, the vehicle upstream of n, vehicle $n+1$, would observe some $v^{n+1}_{pr+1} < v^n_{pr}$ and travel slightly slower than n. A series of control vehicles will then eventually approach v_{g_r}. For each vehicle m running our controller, when $v^m_{pr} - v^m_{offset} = v_{g_r}$, the controller's v^m_{des} will start to track v_{g_r} directly, i.e. VSL-Mode.

In the case where traffic is slower than v_{g_r}, the desired velocity v^m_{des} for all control cars m would still stay at v_{g_r}, however the CBF-Mode is empowered keep vehicle speeds slower than v_{g_r} to maintain forward safety.

In this work, the penetration rate $p \approx 0$, however this control scheme is suitable to be used as p increases in possible future deployments.

7.2.2 Hardware and Software Implementation

The prior section described the controller design, agnostic to specific implementation. This section discusses specific implementation in the environment, the vehicle control implementation, the vehicle-to-infrastructure interface, and the control vehicle computing hardware instrumentation.

Vehicle System Architecture

The control system was installed on two different Toyota Rav4s. In conjunction with the control cars, two additional cars were equipped with GPS recorders (Figure 7.5). Using low cost hardware, the vehicle system accesses the VSL data through a web-based pipeline over an LTE connection with a tethered mobile phone. Vehicle control commands are created by combining the vehicle-to-infrastructure connectivity with the vehicle's proprioception, and the vehicle's local traffic state exteroception. We leverage the Robotics Operating System (ROS) message framework [98] for system integration. The structure in Figures 7.3 and 7.4 are designed as ROS nodes and topics. Our system can be broken down into three categories: vehicle interfacing, VSL integration, and control design.

Hardware Instrumentation

Each of the 4 vehicles were equipped with the same set of hardware for both use cases of data collection and vehicle control. This includes a Raspberry Pi 4 running Raspbian and ROS. A USB GPS module based on the uBlox m8 provided location and time information. A board called the mattHat provided the CAN interface provided CAN reading in all 4 vehicles, and control in the 2 control vehicles. For live VSL database connectivity to get the latest setpoints, mobile phones provided a hotspot over USB cables using a utility called usbmux. Excluding the mobile phones, each hardware kit cost less than $500 USD.

Interfacing with the control vehicle was performed using the software libpanda[97]. Libpanda has the capability to firewall CAN messages between system modules designed by the *original equipment manufacturer* (OEM), allowing third party messages to replace OEM messages. Through libpanda, both on-board measurements can be read/recorded, and control commands can be sent to the vehicle. In tandem with CAN interfacing, libpanda also interfaces with USB GPS modules to provide position information. Libpanda also keeps track of the OEM Advanced Driver Assistance System (ADAS) module state to prevent hardware-level errors when attempting to engage the driving automation system.

Code generation techniques as in [128] are used to convert manufacturer-specific vehicle CAN message into a homogenenous framework in ROS. The vehicle interface is a ROS node with an autogenerated CAN parser that produces sensor data like radar signals and cruise control setpoint. The vehicle interface node is a part of the can_to_ros project [118], exposing CAN-level vehicle systems to ROS. The radar sensor is among the CAN-level sensors. In the case of both of the controlled Toyota Rav4s, the radar produces up to 16 tracks of point cloud data along with relative speed at each point.

Active Traffic Management Infrastructure

The experiment is held on a section of Interstate-24, specifically in the I-24 SMART Corridor [146] located near Nashville, Tennessee. This section is part of an *active traffic management system* (ATMS) to improve safety and reliability. VSL gantries are installed approximately every 0.5 miles to provide speed limits for all lanes, which can change at 30 second intervals. The posted speed limits can vary from 30 mph to 70 mph. A ROS node named gps2vsl can access the information posted to each VSL gantry from a basic URL request, discussed further in Section 7.2.2.

Data Integration from Public Infrastructure

Messages are sent from the traffic operations center to each gantry whenever the speed limit should change. Our VSL data feed is obtained from a database mirror that records these messages from the traffic operations

Figure 7.5: **Experimental Deployment**: Four vehicles, pictured here, are launched into early morning congestion on Interstate-24. From right to left, they enter into the traffic flow. Vehicles 2 and 4 are instrumented for experimental control, and vehicles 1 and 3 are operated under human-piloted control.

center to the gantries.

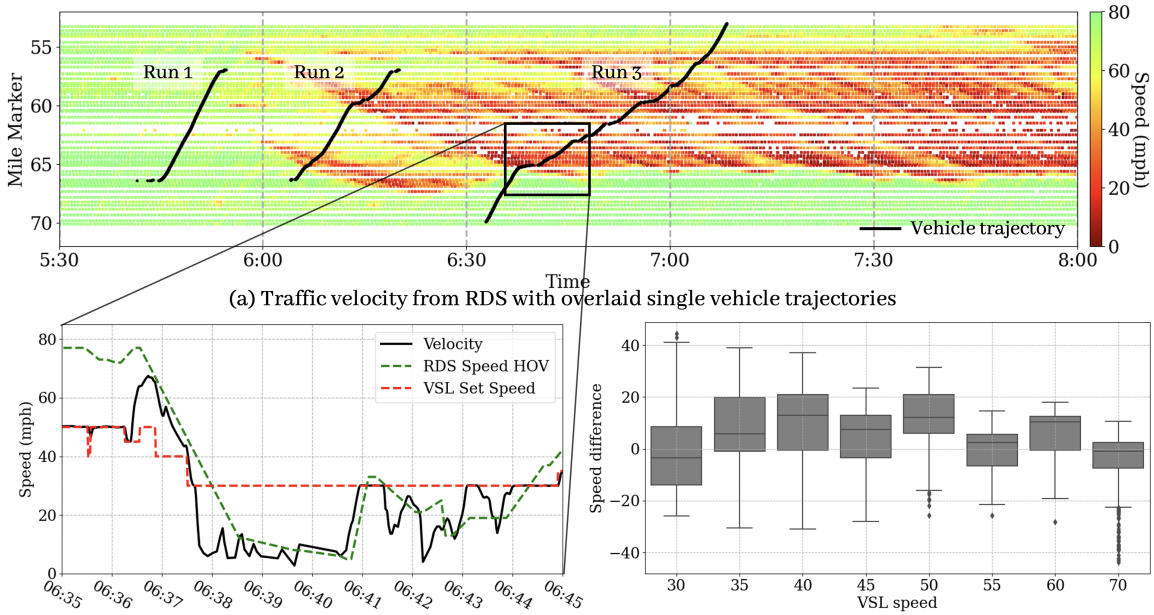
Models for gps2vsl

The gps2vsl node uses the latest position information provided by libpanda, and compares its location against the static set of VSL gantry positions. Geofencing is used to first check that the vehicle is within the I-24 SMART corridor, otherwise a ROS topic informs that the current VSL setpoint is invalid. Once inside the corridor, GPS is used to approximate the vehicle's heading to select either the east bound or west bound gantries. With direction known, the car location is compared against the locations of the gantries. If the vehicle enters within 0.15 miles of a gantry, then that specific gantry's speed limit will be published as a setpoint, along with informing other ROS nodes that the VSL setpoint is valid. Checking the pertinent variable speed limit for the vehicle occurs when entering the bounds of the downstream gantry and every 5 seconds, in order to capture VSL changes by location and over time.

7.3 Experimental Setup

The experimental control vehicle deployment consists of four vehicles. There are two pairs of vehicles; each pair has one control vehicle and a preceding 'probe' vehicle recording trajectory data. The speed of the first vehicle of each pair is regulated by the driver, who is instructed to maintain a safe driving speed at all times. Practically this results in drivers traveling close to the prevailing traffic speed. The speed of the second vehicle in each pair, or 'control vehicle', is regulated by the novel control system introduced in this work. All drivers were instructed to maintain a safe operating environment and to abandon the experiment if conditions on the roadway prevent a safe experiment from being executed. The experiments were conducted in the 5:30-8:30 am window in which I-24 experiences the start of morning traffic conditions and regular traffic waves develop.

The experiment is conducted on a segment of I-24W with four lanes, which starts from the mile marker 70 and ends at mile marker 53. The vehicles are instructed to operate in the left-most lane on the roadway. The



(a) Traffic velocity from RDS with overlaid single vehicle trajectories

(b) Speed profile: vehicle, RDS, VSL comparison

(c) RDS speed v.s. VSL control speed

Figure 7.6: **Measuring the Discrepancy Between Prevailing Speed and VSL**: In three parts, this figure shows the context of the main problem posed in this work. (a) shows estimates of the traffic state from fixed-infrastructure Radar Detection System (RDS) sensors along the SMART Corridor on 08/29/23, with overlaid trajectories from a single vehicle making three Westbound trips. X-axis is time, and y-axis is the roadway mile markers, with the direction of travel going upward. Note the consistent green area below the wall of red; this is where congestion starts. Also note the recurring changes between red/orange/yellow; these are 'stop-and-go' traffic waves. (b) shows the recurring dilemma an individual driver is faced with: when approaching a slowdown, either follow the posted speed limit, or keep up with traffic? In the minutes before a near stop, we observe a 25 mph+ discrepancy between the posted speed limit and the prevailing speed of traffic. This discrepancy resurfaces often, at the peak of traffic waves before the next stop. (c) expands the comparison of RDS (dotted green) and VSL (dotted red) in (b) to the the entire morning's traffic (05:00-09:59). The distribution of differences in speed show that the prevailing speeds regularly reach 10mph-20mph over the speed limit. 23.9% of RDS-measured traffic speeds exceed 10mph over the variable speed limit during morning traffic.

vehicles enter the roadway upstream of traffic waves, and then travel through the heavy congestion where the VSL activates and stop-and-go waves are observed. The current equipped VSL algorithm on SMART Corridor is designed to harmonize traffic speeds and is a modified version of the algorithm described in [10]. In particular, the VSL controller is activated when the observed traffic characteristics exceed predefined thresholds and the speed limits will be rounded to the nearest multiple of 5.

7.4 Results

This section summarizes the main findings of our implementation and experiments in real traffic. First, we will show that the prevailing traffic speed regularly exceeds the posted variable speed limit (VSL) by 10 mph or more on the freeway of interest. This quantifies and validates our anecdotal observations that motivated our design. Next, we establish that local traffic estimates need to be minimally latent to be accurate enough to understand the local traffic state in real time, which supports our decision to measure traffic locally on the vehicle rather than to rely on external traffic sources. Finally we highlight the behavior of our controller

in live traffic, showcasing that the various control modes are all regularly used when navigating complex freeway traffic.

7.4.1 Traffic Speed Far Exceeds Posted Speed Limits

The speed of traffic is regularly much faster than the posted speed limit (10 mph and higher). Figure 7.6 shows this in three parts: the macroscopic traffic patterns, the perspective of an individual vehicle, and the trends in comparison between the infrastructure-based average speed observations and the posted variable speed limit at the time and place of observation.

The time-space diagram in Figure 7.6 (a) shows the typical onset of congestion on I-24 Westbound. This plot is a pairing of fixed infrastructure Radar Detection System (RDS) and an instrumented vehicle recording its trajectories. With time in the x-axis, and the roadway direction going up the y-axis, vehicle trajectories (black) run up and to the right. Consequently, the slope of the trajectory is the velocity of the vehicle; a stopped vehicle creates a horizontal line. Around 6:00AM, traffic waves begin. Before 6:30, an approximately 10 mile region of congestion has formed and will continue for the next couple of hours. There is consistently a large sudden slowdown around MM 67, and a large number of traffic waves shown in alternating red and yellow regions. This overview conveys the typical congested traffic patterns on this roadway.

Figure 7.6(b) takes a closer look at this area of congestion from the perspective of the vehicle trajectories featured in Figure 7.6(a) in black. The variable speed limit is set between 50 mph and 40 mph in the region just upstream of the stopped traffic (06:35), giving an indication to all vehicles that they can anticipate a slow down. At the same time, the speed of traffic continues to travel at an average of of approximately 75 mph until just before 06:37. Before 06:38 the vehicle velocity (black) is below 10 mph and the RDS measuring aggregate speeds in this lane meets this observation at its next provided measurement (every 30s).

The traffic flow does not slow until a minute before meeting a wall of congestion and slowing to nearly a stop. This presents the operator of a control vehicle which only follows the posted speed limit two options: (1) follow the posted speed while the prevailing conditions are 25 mph+ higher, or (2) disengage the controller to support comfort and safety. This decision scenario repeats in the canonical traffic waves of the congestion region, every few minutes. The rest of the results section shows how we address this dilemma: a velocity controller which sees a middle way between the VSL and the high prevailing speeds. The control vehicle, being aware of the traffic speed in local surroundings, of the variable speed limit setting on the roadway, and of the forward collision safety, has a new way to ride the traffic waves in morning congestion. It does this by compromising between traveling fast enough to be ride comfortably in prevailing traffic, while also supporting the pro-safety and wave dampening goals of the VSL's active traffic management system.

7.4.2 Latency in Measuring Traffic Speed Induces Error

Here we describe the effect that latencies in fixed-infrastrucutre Radar Detection System (RDS) have on the accuracy of estimating prevailing traffic speeds. First, to create an estimate of prevailing speeds from RDS that we consider 'ideal', we compare the trajectory of the test vehicle back to the historical speed measurements. Every point along the test vehicle's freeway trajectory is mapped to the 4 RDS speed measurements in space-time that contain that point. The ideal measure of the prevailing traffic speed at that trajectory point is then calculated by taking the average of these 4 points, which would not be possible in real-time. Figure 7.7 shows the RDS speed measurements captured in the test lane, as well as the control vehicle's trajectory.

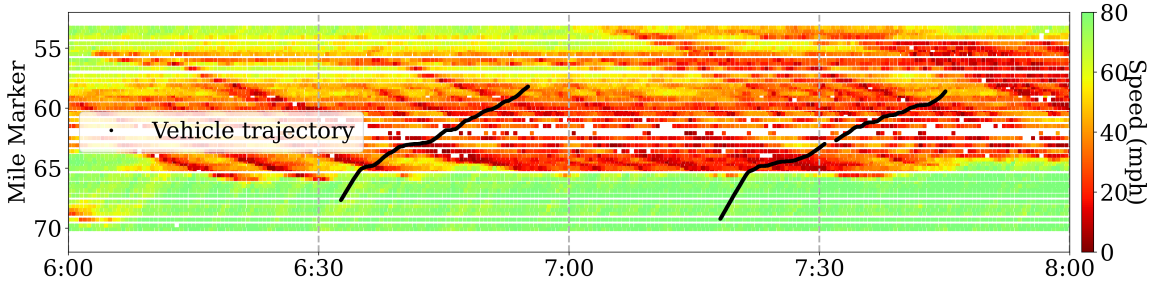


Figure 7.7: **Middle Way Control Deployed** Trajectories of a control vehicle are shown, laid over RDS speed measurements from the lane of travel. As in Figure 7.6, X-axis is time, and Y-axis is the roadway mile markers, with the direction of travel going upward.

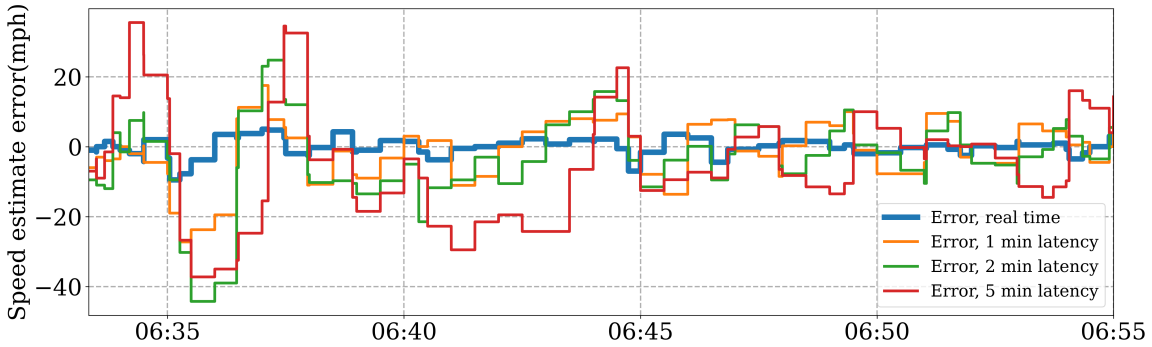


Figure 7.8: **Latency-Induced Errors**: Estimates of the speed of traffic are made, showing the effect of latency over time. Small errors in estimation are exacerbated with latency, because of how quickly the state of the traffic system changes in congested regions.

We subsequently compare the ideal speed measurement to the speed measurements that would have either been available in real-time, or with a certain amount of latency. Real-time speed estimates are calculated as the average only in space between the two RDS measurements most recently available at each trajectory point (but does not consider the 2 points ahead in time, as the ideal speed measurement does). Additionally, we account for possible latency by shifting the trajectory only in time by a certain added latency, and then performing this calculation again. The errors between the real-time, 1 minute latency, 2 minute latency, and 5 minute latency speed estimates and that of the ideal speed measurements are shown in Figure 7.8. In Figure 7.9 these errors are then shown as distributions. It is evident that real time RDS measurements have some error, and that latency in their measurement noticeably exacerbates the errors. The standard deviation of error increases 3.2 times from 2.35 mph away from 'ideal' to 7.45 mph with just one minute of delay. The standard deviations increase further, eclipsing over 10 mph of error, with standard deviation of 10.35 mph at two minutes of delay, and 12.91 mph at five minutes of delay. Commercial entities sell access to average traffic speeds with multi-minute delays, and the RDS sensors are limited to reporting over 30 second intervals. Considering the observation from Section 7.4.1 that in congested regions it is common to see 30 mph+ changes within 30 seconds, these fixed delay costs could be problematic. To avoid these issues, our control design opts to take estimates of the traffic speed from the nearby vehicles as measured by the on-board stock radar sensor.

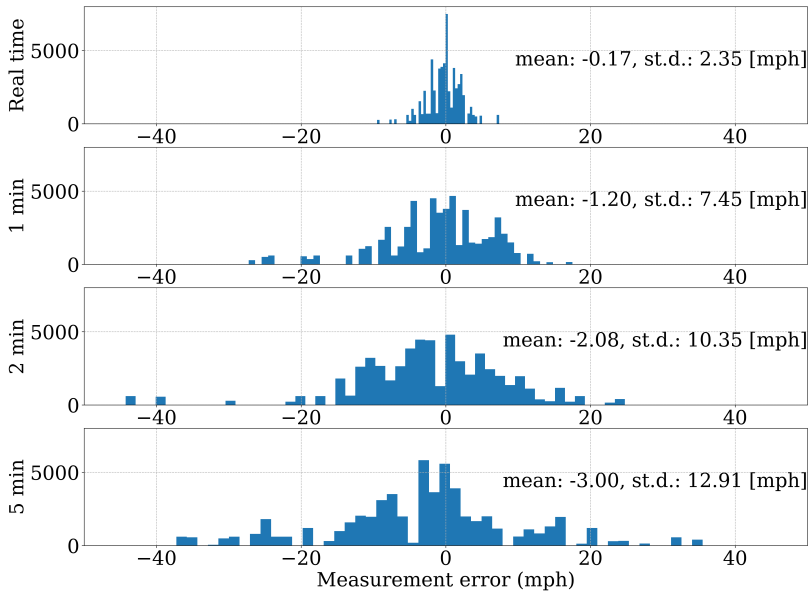


Figure 7.9: **Distribution of Latency-Induced Errors**: From Real time, to 1,2, and 5 minute latency. Notice a widening distribution of error in the measurement of local traffic speed.

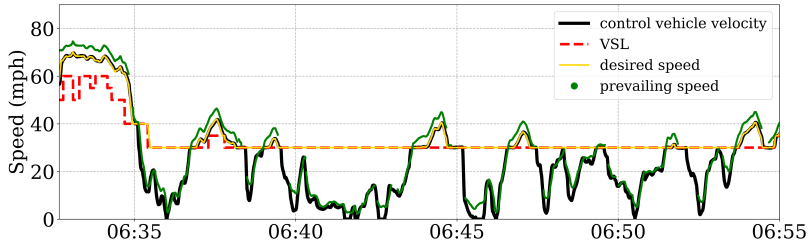


Figure 7.10: **A Single Complete Control Vehicle Trajectory**: An overview of a pass going through heavy morning congestion from the perspective of the control vehicle. 59.4% of the time is spent in CBF-Mode. Entering congestion and at the peak of recurring waves, the vehicle is in VSL-Mode (24.0% of time) and Middleway-Mode (16.6% of time).

7.4.3 Controller Performance

Earlier in the results we cover analyses informing design choices made to create a controller which is aware of the not just the variable speed limit, but also the local traffic speed. This subsection of the results showcases that controller's performance in deployment on the interstate during heavy morning congestion.

Figure 7.10 provides an overview of a driving trajectory with the novel controller. This plot emphasizes control decisions from the single vehicle perspective. In red, we show the posted VSL v_{g_r}. In green, we show the prevailing traffic v_{pr}. In gold, we show the desired speed v_{des} from the controller. Speeds are initially near 60 mph (v_{g_r}) and 75 mph (v_{pr}); at ∼06:33 we see the VSL drop gradually to 30 mph. Note that v_{g_r} is substantially slower than v_{pr} until the speed of traffic slows down to a stop at approximately 06:36. The control vehicle then speeds up and slows down over 20 more minutes in traffic waves. There are three states predominantly driving the vehicle's velocity (black). (1) any time the vehicle velocity is below the v_{g_r}, the CBF-Mode safety control is active; (2) matching v_{g_r} exactly (VSL-Mode); and (3) the speed of traffic v_{pr} is far enough above the v_{g_r} that the vehicle deviates from the posted speed limit. Over the time periods where control was active in the deployment, 59.4% of the time is spent in CBF-Mode, 24.0% of the time is spent in

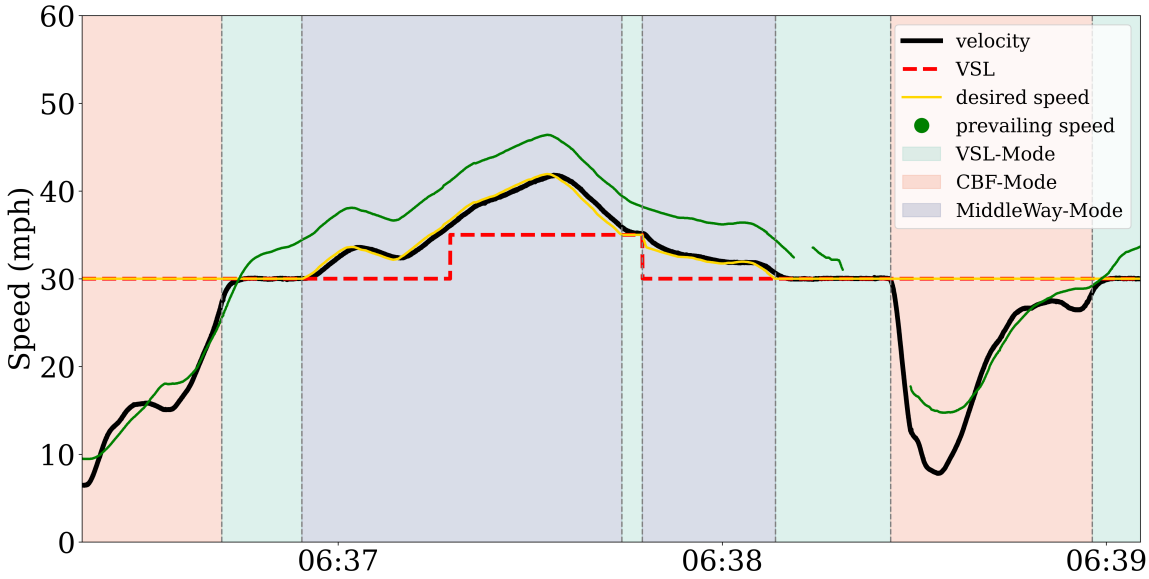


Figure 7.11: **Control in a Traffic Wave**: In a single traffic wave, we can understand the evolution of the state of the experimental control system in these recurring scenarios. Repeatedly, there traffic speeds up enough to allow the choice to follow the speed limit (VSL-Mode), then possibly speeds up faster than v_{offset} above the speed limit inducing MiddleWay-Mode, and slows again to find VSL-Mode again, then well below the speed limit inducing CBF-mode.

VSL-Mode and 16.6% of the time is spent in Middleway-Mode. In the entry to congestion before 06:35, the control vehicle is primarily keeping up with traffic, then pauses for a moment at v_{g_r}, before entering CBF-Mode. Throughout the rest of the congestion region on this westbound I-24 pass, the predominant driving automation is within the CBF-Mode; however, at the peak of the recurring traffic waves there are repeating opportunities for the novel controller to either travel in VSL-Mode, or speed up more to keep up with traffic speed in Middleway-Mode.

Figure 7.11 gives a closer look at the behavior of the controller in the recurring traffic wave scenario. There are several transitions between control modes, which are highlighted in different colors. Time regions in red have CBF-Mode active, time regions in green are when adhering directly to the posted VSL (VSL-Mode), and time regions in purple are speeding above VSL due to faster traffic speed (Middleway-Mode). This plot begins at the end of slowest part of a traffic wave, where the control vehicle speeds up and has the opportunity to match the VSL. The CBF intervenes with limits on acceleration to prevent forward collisions, so once the preceding local traffic speeds up the control vehicle reaches v_{des} at the posted VSL before 06:37. Approaching the velocity peak of the traffic wave, the traffic speed is faster than the runtime offset parameter v_{offset} (in this case $v_{pr} - 2\frac{m}{s}$), and the MiddleWay-Mode activates. Without this feature in place, a VSL-Mode following vehicle would have to weather a minute or so at each wave peak going 10-15 mph slower than traffic speed; this is an uncomfortable condition as a passenger.

In the wave shown in Figure 7.11 v_{g_r} increases and the control vehicle catches the VSL-mode again at just before 06:38. Within a couple seconds the posted VSL v_{g_r} drops down again, but instead of dropping down immediately the control vehicle stays within the offset below traffic speed and eases over the next 30 seconds or so to the posted VSL setting as the traffic slows. Around 06:38:30 the CBF-Mode is activated again as the control vehicle reaches the entrance to the bottom of the traffic wave, and soon proceeds to the beginning of

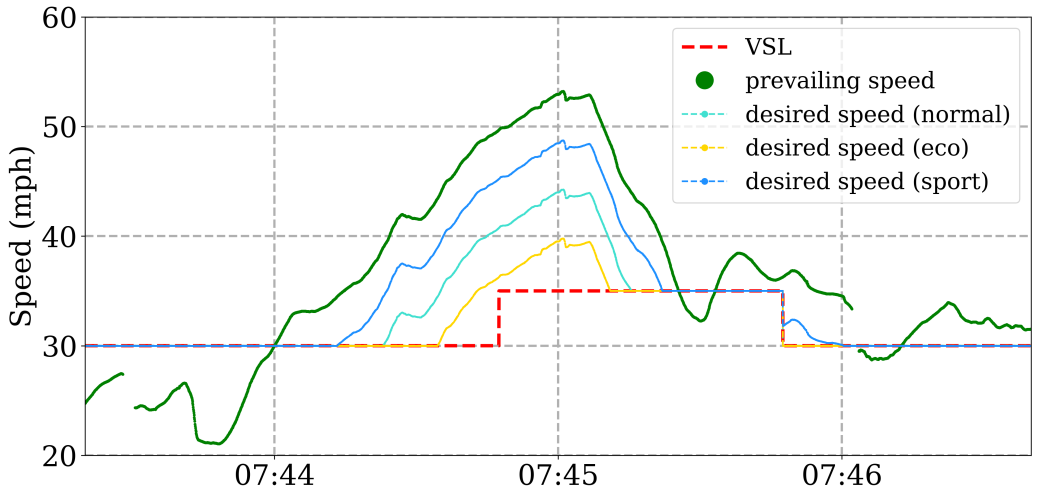


Figure 7.12: **Runtime Parameter v_{offset}:** The offset from v_{pr} can be set to 2/4/6 $\frac{m}{s}$ by the vehicle operator as they travel through congestion. This is achieved by listening to the vehicle drive mode (Sport/Normal/Eco, are 2/4/6 $\frac{m}{s}$ respectively). A smaller offset allows the novel controller to keep velocity closer to the local traffic speed, whereas a larger offset allows for more time travelling at the posted VSL.

the cycle again.

In the recurring traffic wave scenario, the offset parameter v_{offset} can be considered as the condition of how much faster than the posted speed limit does the vehicle need to observe the local traffic speed before deviating from the posted speed limit and speeding up. Figure 7.12 shows an open loop projection of how different offsets would effect the desired velocity, v_{des}, with the same observed traffic speed, v_{pr}, and posted VSL, v_{g_r}. At 07:44, when the plot for Figure 7.12 begins, the prevailing speed is at v_{g_r} so v_{des} is at $v_{g_r} = 30$ mph for all offset settings. When $v_{pr} \leq v_{g_r}$, $v_{des} = v_{g_r}$ by system definition. However, within a minute v_{pr} has increased by over 20 mph ($\sim 8\frac{m}{s}$). Now that $v_{pr} > v_{g_r}$ the offset parameter has a significant effect on where between v_{pr} and v_{g_r} the velocity of the vehicle will go. A smaller offset (i.e. 'Sport Mode') leads to a quicker switch to tracking v_{pr} instead of v_{g_r}. In the deployments made in this work, operators of the control vehicles chose primarily a 'Sport Mode' $2\frac{m}{s}$ setting; when the 'Default' $4\frac{m}{s}$ setting was in use, it was deemed too slow. This could vary in different traffic conditions and the difference between a larger population of operators; more investigation is needed to characterize v_{offset}.

7.5 Conclusions

This paper presented a new cooperative automated vehicle controller that adopts variable speed limits set by smart infrastructure, and adapts to the speed of traffic when prevailing speeds warrant doing so for safety reasons, and relies on a control barrier function when following a vehicle ahead. The result was a new real-time algorithm which was enabled by high-latency communication to infrastructure, low-latency on-board sensors, and real-time algorithms on board the car that are informed by the cyber-physical properties of the ego car and the vehicles around it.

The maximum vehicle speed never exceeds the value specified by the driver using the heads-up-display, mitigating this source for mode confusion. The updates to the variable speed limits can be made through mobile phone connectivity at high latency, without compromising the safety or efficacy of the solution. The implementation changes to deploy at scale are minimal, and do not require sensors or connectivity beyond

what is present on most vehicles sold today.

Field experiments validated the work on the open road during times of congestion when the VSL was active. The results show that the on-board ego car sensors were able to accurately estimate the speed of the flow of traffic (not just the speed of the ego car), as validated by roadside sensors. The field experiments demonstrate that each of the three modes of the presented controller are active during the drive in substantial portions, validating that the speed adaptation novelty has merit.

Additional validation in the field experiment showed that speed of traffic far exceeds the posted speed limits as the ego car approached stopped traffic. This mode, in particular, validates the middle way approach: driving slower to increase effective compliance of the VSL, but in a way that follows accepted safety guidelines. Further, traffic speed estimates from roadside sensors were shown to be unsuitable for real-time safety feedback, with latencies in which they are currently available.

Future work will explore large-scale simulations, high-resolution measurement of the influence on neighboring cars, and additional field deployments at scale. The large-scale simulation will explore how design choices in our prototype system would work at higher penetration rates. Further field deployments at scale can measure the influence on other vehicles in the flow, allowing us to infer an effective compliance rate based on our own measurements, to build advanced models for broader application in other system designs.

8 | Concluding Remarks

Arrakis teaches the attitude of the knife -
chopping off what's incomplete and saying:
'Now, it's complete because it's ended here.'
from "Collected Sayings of Maud'Dib" by the Princess Irulan

Frank Herbert, Dune (1965)

8.1 Recap of Contributions

The main contribution of this dissertation is a testbed for experimental connectivity and automation in cars. This testbed has been shown to be low-cost, scalable, and extensible. It has supported closing a car's control loop with a human-in-the-loop, and in an automated fashion; it was used to deploy the largest CAV traffic experiment to date, in order to control traffic in the mixed autonomy setting; and it has supported connectivity with infrastructure, and with other vehicles. The individual sets of contributions of this dissertation are summarized below:

Flexible Experimental Vehicle Control with a Human-in-the-loop. The CAN Coach is introduced, which gives feedback to a human-in-the-loop using on-board sensor data. We show that the CAN Coach can be effective as a *human-in-the-loop cyber-physical system* (HCPS) in improving the longitudinal control of an individual vehicle in a traffic flow. We conclude that (1) it is possible to coach drivers to improve performance on driving tasks using CAN data, and (2) it is a true HCPS, since removing human perception from the control loop reduces performance at the given control objective.

A Pipeline for Experimental Automated Vehicle Control. This pipeline avoids modeling challenges and produces a reinforcement learning (RL) controller that is then successfully deployed on four vehicles in dense highway traffic. This pipeline has three parts: (i) the data collected from human driving trajectories, (ii) the RL controller, and (iii) the deployment of the controller on physical vehicles. The dataset consists of 772.3 kilometers of recorded drives on the I-24. Using a simple simulator that integrates the recorded data, we learn a controller via policy-gradient methods with an asymmetric critic. We show that we are able to improve average MPG by 11% in simulation on congested trajectories, and then deploy the controller to a mixed platoon of 4 autonomous Toyota RAV-4's and 7 human drivers in a validation experiment and demonstrate that the expected time-gap of the controller is maintained in the real world test.

Enabling Mixed Autonomy Traffic Control. We deploy the first large-scale team of connected automated vehicles (CAVs) for mixed autonomy traffic control. We introduce a hardware and software platform to enable experimental autonomy and connectivity in commercially available SAE level 1 and level 2 automated vehicles. The platform supports experimental automated vehicle control, live mobile sensing with connected vehicles, high fidelity data collection, and scalability. This platform enables an agile develop/deploy cycle for at scale cyber-physical systems research, empowering its field deployment.

SAILing CAVs: Speed-Adaptive Infrastructure-Linked Connected Automated Vehicles. The development and field deployment of the first connected automated vehicle with the capability to follow publicly broadcast variable speed limits. We implement this capability with open source hardware and software that

extends a stock vehicle's adaptive cruise control. We demonstrate the system on a vehicle in heavy traffic on an open roadway, and compare the performance of the equipped vehicle to a human piloted vehicle driving in the same traffic.

***Via Media*: Fielding Connected Automated Vehicles to Follow Variable Speed Limits in Low–Compliance Regimes.** Introduces of a new notion of safety for cooperative automated vehicle applications to avoid causing controlled vehicles to drive substantially slower than surrounding traffic. The approach recognizes the necessity for automated vehicles to adhere with the typical driving behavior observed on the roads, even if it requires a deviation from the posted speed limit. Subsequent field experiments on two control vehicles operating *via media* in rush hour traffic found that the vehicle spends 16.6% of time following the variable speed limit, 24.0% of time above the speed limit due to prevailing traffic, and 59.4% of the time in a car-following mode to prevent forward collision.

This testbed flexible enough to allow for implementation and deployment of many connectivity and automation ideas in freeway-based ITS research projects. Future directions include making the testbed more accessible for third-party development, long-term and ongoing deployments, and the development and integration of security in communications.

8.2 Future Directions

The following outlines some future directions building on the contributions herein:

Third Party Applications This dissertation has featured the developments which enable competing and updated candidate controllers to be equipped dynamically and at scale in connected automated vehicles. Developing some form of a software development kit (SDK) would supercharge this capability, by empowering researchers and developers less familiar with the hardware and software stack of the testbed to develop their own third-party applications. The CAV SDK would feature APIs to access the control, sensing, and connectivity infrastructure that has been developed as part of the testbed at the core of this dissertation. Enabling third-party applications has broader impacts that stem from increased accessibility, and modularity. Making it more accessible to develop an application for CAVs will accelerate learning, development of novel ideas, and discoveries. Further, a modular app-based infrastructure could make software management cleaner and simpler; contracting the complexity in the software systems supports faster and error-free development.

Big Data Large data sets are at the heart of contemporary rapid technological developments. A well-known in example of this is in language models. Starting with 'Attention Is All You Need' [151] which introduced Transformer models, to the 175 billion parameter GPT-3 model (i.e. ChatGPT), and beyond to trillions of parameters.. these large language models (LLMs) achieve increasingly amazing performance in part because big data. By leveraging big data, transportation researchers can discover new patterns and develop data-driven approaches to making better operational decisions, and improving mobility at a societal scale. However, new tools need to be developed to generate, collect, collate, store, and analyze various kinds of transportation-related data sets. Transportation data, unlike language data, is siloed in closed traffic cameras, mobile phones, cars, infrastructure, and more. The testbed introduced in this dissertation can play two of important roles in moving toward big data and machine learning insights for traffic and mobility. (1) deploying the many vehicles in a long-term context like in the Living Lab [152] concept would allow for ongoing high fidelity data collection at-scale. Some existing data exists for naturalistic driving, but without the intent of collecting information relevant to experimental connectivity and automation the data is not useful in this context. (2) since the testbed is already integrated into the TDOT Smart Corridor, which is co-located

with the I-24 MOTION system[153], there is potential for a jump-change in big data for transportation by integrating all of these component data sources.

Dynamic Connectivity To maximize the utility of connectivity, dynamic connectivity with hybrid networking will be useful. Dedicated short-range connections, ad-hoc networking, on-board networks, telecommunication networks, and more, can be combined to get the best system performance and efficiency. Different types of connections and networks have differing strengths and shortcomings. Combining them, with a Bayesian approach to belief, will greatly increase the value of connectivity in cars; especially in low-penetration and non-uniform regimes where peers and smart infrastructure are often intermittent. In low-penetration connectivity regimes, cars and information streams may need to rely on telecommunications to pass messages across further distances since short-range radios will be ineffective or unrelaible. In other cases, such as with many secure open standards, it may be preferable to pass information from peer-to-peer if penetration rates locally are high enough. Ultimately, hybrid networking would strengthen the V2X capabilities of a car.

We can also consider the utility of establishing robust connectivity in mixed use settings. Modern urban mobility has unfederated mixed use with pedestrians, bicycles, cars, freight, and buses sharing the streetscape. These users increasingly have some forms of mobile sensing and networking communications, though they will not necessarily be uniform. Creating pathways for information sharing between mixed mode mobility as well as the smart infrastructure could open doors to new applications. We have seen the 'green wave' strategy for signalized intersections, and then the 'blue wave' corollary for reducing emergency response times; the next application may increase pedestrian safety through infrastructure connectivity, or increase bicyclist safety with integrated warnings to locally connected cars.

Security As soon as networking entered cars, they started being vulnerable to cyber-attacks. Some of the work described in this dissertation could be characterized as white hat man-in-the-middle attacks on OEM computer systems. Going forward, as connectivity increases, considering the secure networking and message passing protocols for cars will be critical. Self-organizing peer-to-peer networked communities are desirable for cars as mobile sensors, as it offers dynamacism and scalability to message passing in the complex transportation CPS. Without universal centralization, however, these networks are susceptible to malicious behavior such as lying and collusion. Development of decentralized trust management systems, perhaps using notions of reputation, neural network-based models, Bayesian thinking, or consensus protocols, will allow useful message passing to be reliable. With reliable information flowing, connected cars can have significantly improved safety, efficiency, and performance. Some versions of safe message passing have been derived, simulated, and tested at small-scale [45], [46], [134]; deployment at-scale in a real mixed autonomy remains to be seen.

Part IV

Appendix A

9 | Middleware for a Heterogeneous CAV Fleet

This section includes material from a publication:
M. W. Nice, M. Bunting, J. Sprinkle, *et al.*, "Middleware for a heterogeneous cav fleet," in *2023 5th Workshop on Design Automation for CPS and IoT (DESTION)*, 2023

9.1 Introduction

Since modern commodity vehicles house an increasing volume of sensors and automation, they are ripe to be transformed into research tools, or even experimental control vehicles. In order to extend commodity vehicles' use to be a research platform, a vehicle interface needs to be determined. ROS is an common standard for applications of robotic control [98]. In-vehicle networks have an obscured mix of protocols and implementation details, and it may be necessary to create bridges from different vehicle middlewares to ROS [154]. A large gap exists between the in-vehicle networks and ROS. Previously, an in-vehicle network to ROS bridge [118] was described. However, that work lacked the ability to adapt to new platforms, and did not support a heterogeneous fleet of vehicles. This work introduces an automated tool to support these needs.

A longstanding problem in robotic control was implementing and adjusting code for control in ROS. When developing, adapting, or fixing your implementation, there was a large exposure for errors to be introduced. To address this, Mathworks' ROS Toolbox has introduced ROS code generation from Simulink models with fantastic success. This tool integration, which includes a correct-by-construction code generation technique, has been adopted by robotics and cyber-physical systems software developers. Though in a smaller niche, our tool is inspired by this concept. Given the specifications of a vehicle model, this tool generates nodes to meet the ROS layer, and regenerate nodes driven by new developments, adaptations, or features introduced.

The contribution of this work is the design of a generative approach for CAV middleware, where the databases used to map signals during analysis can be redirected to topics and message types within ROS for runtime access to vehicle data. This goes beyond previous approaches, in that it by definition permits new or updated vehicle integrations simply by adding a new or updated database to the framework—without having to write software that maps CAN decoding information to ROS topics.

We introduce a flexible two-way interface between the in-vehicle network and the ROS network. In tandem with libpanda [97], it generates a bidirectional interface between ROS and the vehicle. This interface can regenerate at runtime in a heterogeneous fleet to match the vehicle detected at startup. Specifications can be quickly changed to aid in live sensor message decoding, agile vehicle control development, and other software application development user stories.

9.2 Background and Related Works

The current paradigm for in-vehicle networking is to have a series of electronic control units (ECUs) which act as computing nodes connected by several data busses which act as edges that connect the nodes. There is large variation in implementation, and much is obfuscated by manufacturers from the public. Some data busses are twisted difference pairs carrying Controller Area Network (CAN) data. This older protocol has been the target of some open sourced tools which have endeavored to publicly show how to make use of in-vehicle sensors

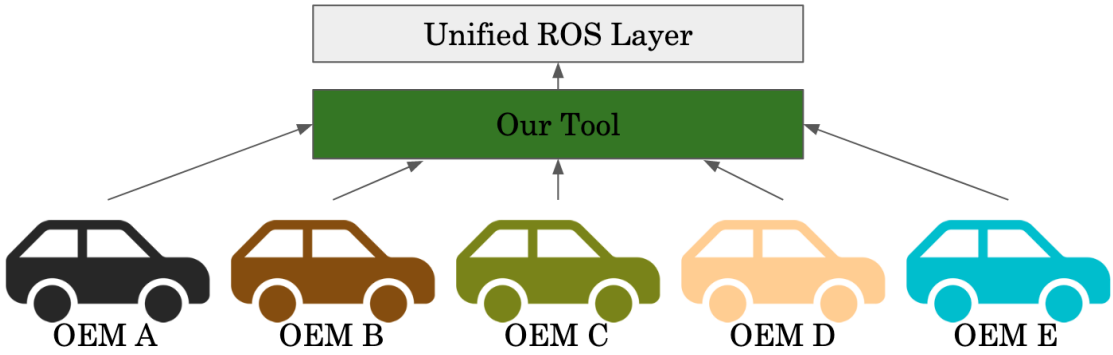


Figure 9.1: Our tool introduced in this work features automated ROS node generation to enable the development and deployment of a heterogeneous vehicle fleet from multiple Original Equipment Manufacturers (OEMs). In deployment it functionally makes lower-level vehicle-specific detail abstract to upper level software applications.

to extend the use of vehicles [130], [155]. These have even been used to drive experimental cyber-physical systems research [4]. However, CAN is phasing out of use in favor of CAN with flexible data rates (CAN FD), Automotive Ethernet, FlexRay, and other new vehicle networking protocols. CAN FD, for instance, is similar to CAN in many ways but has an order of magnitude increase in data rates. As a consequence, new security protocols and protections are being implemented, and ECUs must have heftier compute power. For system-critical information with high volumes of data and real-time constraints, like forward facing radar, CAN FD is not sufficient. Faster data rates like Automotive Ethernet (1000BASE-T1) are used, which is fast enough for potential use in time sensitive networking (TSN)[156]. Another characteristic of in-vehicle networks is the OEM baggage carried from decades of development. Under close examination and without insider explanation available, the implementation decisions observed often follow confusing patterns. Altogether, today's in-vehicle networking consists of corporate obfuscation, heterogeneous agglomerations of protocols, and cumbersome electronics; this work endeavors to digest a great deal of the difficulty in these details in order to provide a simple and easy to use interface for upper level software applications in commodity vehicles. Furthermore, we show it is apt for use across vehicle models by testing its use on vehicles from three different OEMs.

The underlying motivation for this work is to enable commodity vehicles to be capable of experimental robotic control. ROS [98] is the common interface where existing tools for control systems can meet with the vehicle interface. The challenge is to emerge from the detailed depths of vehicle implementation and meet control systems at the ROS layer with abstracted information on sensor and vehicle control data. Commercial or custom hardware can bridge electronics from the vehicle to the software. We leveraged libpanda [97] to manage the electrical to digital data interface. Our tool aims to fill the gap from raw digital data to ROS for agile use in a heterogeneous setting.

9.3 Middleware for Heterogeneous Fleet

The CAN to ROS tool, in tandem with libpanda [97], serves as the pipeline between the hardware (vehicle with embedded hardware instrumentation), and the ROS network. In ROS, data is processed, recorded, and distributed; this can include housing a vehicle control algorithm which sends control requests to our tool and through libpanda into the vehicle. The cumbersome heterogeneity of different vehicles are abstracted away

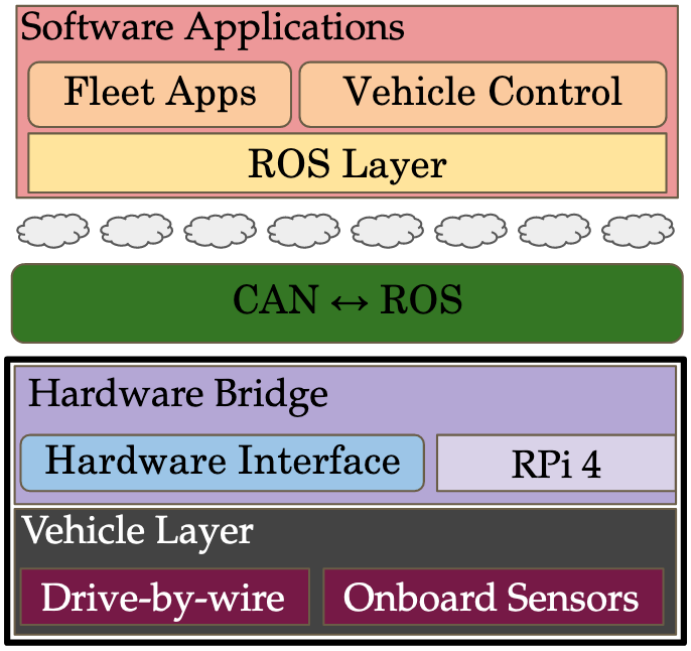


Figure 9.2: Our tool sits above the hardware of an in-vehicle network with access to live data. At runtime, our tool self-configures to provide a prescribed set of published sensor values in ROS. This shields upper level software applications from lower level complexity and heterogeneity.

by CAN to ROS to shield the upper level software applications from the lower level complexity and heterogeneity. Software applications which could require consumption of vehicle data include vehicle control, connected vehicle communications, or fleet monitoring systems. This middleware is useful online for field deployment and reverse engineering efforts, and useful offline for analysis and software-in-the-loop simulated use.

9.3.1 Live Use

Code generation for CAN to ROS was implemented initially to combat issues holding back new development and testing. If a CAN message needs to be added from, removed from, or modified in the ROS layer it can be done fast and with confidence because of the code generation framework. This happens often when adding a new vehicle to be supported by the software. Having the ability to monitor tenuous and established vehicle sensor signals drove developments in upper level software applications for field deployment, and new signal deciphering and correlation. Without this code generation automation, the development/testing cycle may have been significantly hindered by the system complexity. A welcome knock-on effect of the automated code generation was valuable flexibility in field deployment at-scale in a heterogeneous fleet. Software systems could automatically self-configure their sensor reading software, and control interface.

In situ CAN Signal Message Decoding

If you want to transform a commodity vehicle into a research instrument without adding custom sensing, then you need to be able to understand and leverage the on-board sensors. Utilization of message data on the CAN at runtime is the key to tight integration of software with car hardware. CAN encodings may be widely known, for example in the case of signals that are published by automakers for diagnosis of vehicle faults,

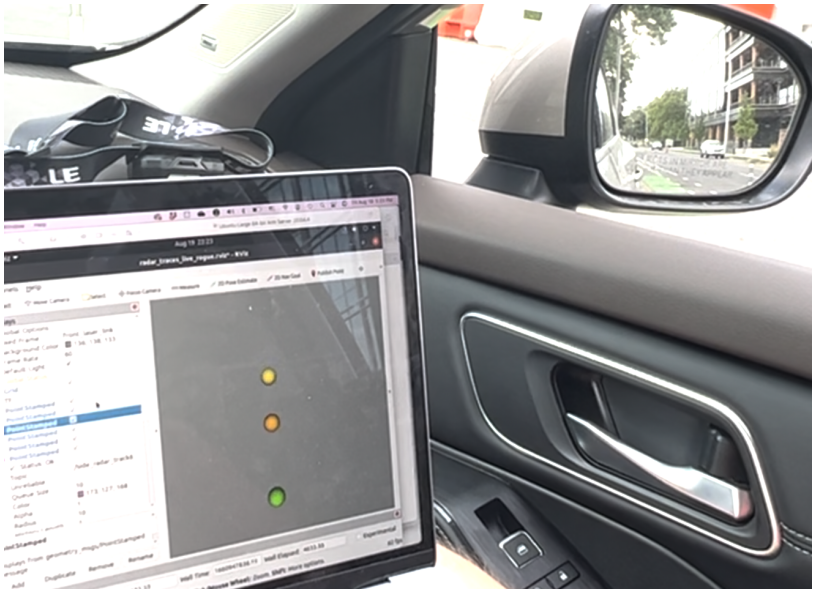


Figure 9.3: *In situ* CAN decoding is facilitated in ROS, thanks to the ability to project messages from multiple signals into visualization engines that can help in correlation.

or shared by many manufacturers for emissions checks. In many cases, the encodings used by automakers may not be published widely, but have been discovered by enthusiastic vehicle hobbyists. As new models of vehicles are embraced by researchers and hobbyists, there is a need to decode signals in order to inform runtime algorithms of current vehicle state—and to potentially modify information on the CAN by injecting control signals to influence the vehicle state.

Our research team goal is to decode only the messages we need for research tasks—not to perform a full-scale reverse engineering task and publish the results. Several works aim to create prescriptive solution to automate decoding and validation of in-vehicle sensor data, though each of these typically requires either some ground truth information [157], or utilizes an information theoretic [158] or machine learning approach [159] to classify the sensor type (not confirming any further details of the sensor).

An ideal approach is to allow for analysis and testing *in situ*, permitting an operator to correlate their own runtime perception with changes observed on the CAN while adjusting the ranges and widths of the CAN decoding database. Thus, our solution permits a design cycle that publishes an array of candidate decodings that could match a desired signal, generating the runtime framework that publishes those candidate decodings as ROS messages. Once these candidate signals are published, the operator can provide impulses to the system that should be visible using ROS tools such as rViz, as seen in Figure 9.3.

With live data streaming from the vehicle and subsequently being published in ROS, a researcher can test their postulated reverse engineered signal in real time. Pressing the pedals and perform behaviors which should trigger a suspected signal output in ROS, can be immediately informed as incorrect (if not observed) or reserved for continued investigation (if observed). Upon discrepancy, a researcher can adjust their signal decoding, or add more reference signals to be published in ROS, by simply editing one line in the JSON and DBC files and regenerating the decoding nodes.

Code Generation

To support a heterogeneous fleet of vehicles, new software components were created to automatically self-configure CAN to ROS. The workflow is summarized in Figure 9.6. This self-configuration feature allowed for abstraction of the specific vehicle sensor data signal details, which vary from vehicle to vehicle, into uniform signals in ROS (i.e. 'velocity' for all heterogeneous vehicles). CAN database (DBC) and JSON files specify signal decodings and a mapping from the DBC to the ROS message types and naming, respectively. Bash and python scripting uses this information to rewrite and build ROS nodes in C++ to publish can signals live for the supported vehicle. Automatic self-configuration relies on a function in libpanda[97] to read the vehicle identication number (VIN) of the vehicle, and a physical connection to the vehicle to read the VIN. Altogether this allowed for a previously unconfigured (or misconfigured) software system to be plugged into a supported vehicle and automatically self-configure, and subsequently read and record on-board vehicle sensors live. This is compatible with other automated features like experimental automated control systems and over-the-air software updates.

Structure for Vehicle JSON

Here we detail the information embedded in each vehicle-specific JSON for a supported vehicle. The data structure is a nested dictionary. The top-level keys are each single Message IDs for the vehicle sensor data. The Message ID is a static identifier for a grouping of broadcast messages which can contain several vehicle signals. The top-level values are a dictionary of ROS messages derived from the Message ID key. The keys in the nested second-level dictionary are the names of the ROS topic to be generated, and the values are the ROS message type for the ros topic and the signal(s) which compose the constituents of the ros topic which can be found in the DBC file. For example:

```
{"835": {"acc/accel_cmd": [["std_msgs::Float64"],
["ACCEL_CMD"]], "acc/acc_info":
[["geometry_msgs::Point"],
["MINI_CAR", "CAR_AHEAD", "CANCEL_REQ"]],
"acc/acc_cut_in": [["std_msgs::Int16"],
["ACC_CUT_IN"]]}}
```

A lot of key signals are continuous, but there are signals, like active gear, which are discrete states. For this case there are some extra options: if the ROS message type is a string, and a string lookup is defined in the DBC, then CAN to ROS will generate the node to decode the data which is natively an integer in-vehicle and publish the data as your string-defined state. If instead the integer-based state is preferred then simply define an integer for the ROS message type.

Field Deployment

The auto-configuration feature is an asset during field deployment. With a heterogeneous fleet of vehicles with embedded devices, you can be agnostic to which embedded hardware units are in the which vehicles or even swap them as needed. The simple runtime logic is explained in Figure 9.4; after the car starts power is supplied to the embedded Raspberry Pi 4 computer and it boots up; the VIN will be automatically checked on boot, and if it is different than on the previous power cycle its details will be looked up from public

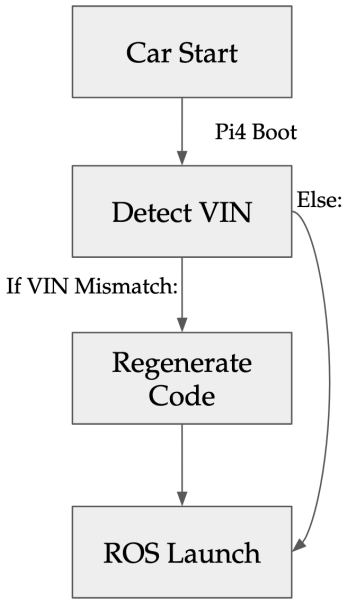


Figure 9.4: Flow diagram showing the runtime decision for code generation. After checking the VIN, either there is no mismatch and the start up processed proceed, or there is a mismatch and the in-vehicle network decoding needs to be changed. Once the translation node is rebuilt, the startup processes proceed as planned.

reference; a mismatch between the stored VIN and the currently read VIN triggers a code regeneration; after code regeneration, or if there is no mismatch in the first place, the ROS launch and the rest of system startup proceeds. This automation feature aligns with a cyber-physical systems software and hardware development philosophy which demands dynamic automated tools to sense their environment and adjust automatically. This is similar to common internet of things features, such as smart speakers able to sense localized acoustic patterns and reconfigure its tuning parameters to optimize performance.

There are two primary nodes that handle the decoding of in-vehicle network sensor messages. These nodes are written in C++ to provide fast performance even from lightweight embedded computers. If the use is online, the package parses live vehicle data with a parser from libpanda[97], and publishes the message information in the `/realtime_raw_data` node. Offline use leverages a recording of the `/realtime_raw_data` topic to recreate a live environment, or simulates live data to parse from a timestamped recorded CSV vehicle data file. The `/decode_subs` node subscribes to the data payload strings from the `/realtime_raw_data` node. The specified callback functions will parse, decode, and publish specified sensor information to its specified topic. These topics are shown as arrows in Figure 9.5, with nodes as ovals. Green color indicates which nodes code and topics decoding will be changed to conform with vehicle-specific configuration when switching between vehicles. Yellow nodes are optionally included, depending on specification. Gray nodes represent the arbitrary consumers of the in-vehicle sensor data in the ROS network layer.

9.3.2 Offline Use

Libpanda provides a feature to record and timestamp in-vehicle data in CSV files, which is useful for offline processing. Carried forward from previous work [118], this can be used for playback simulating a hardware-in-the-loop setup in the software-in-the-loop environment. This permits previously captured CSV files to be re-broadcast in ROS if a decoding is later discovered, which was not known at the time it was recorded. What

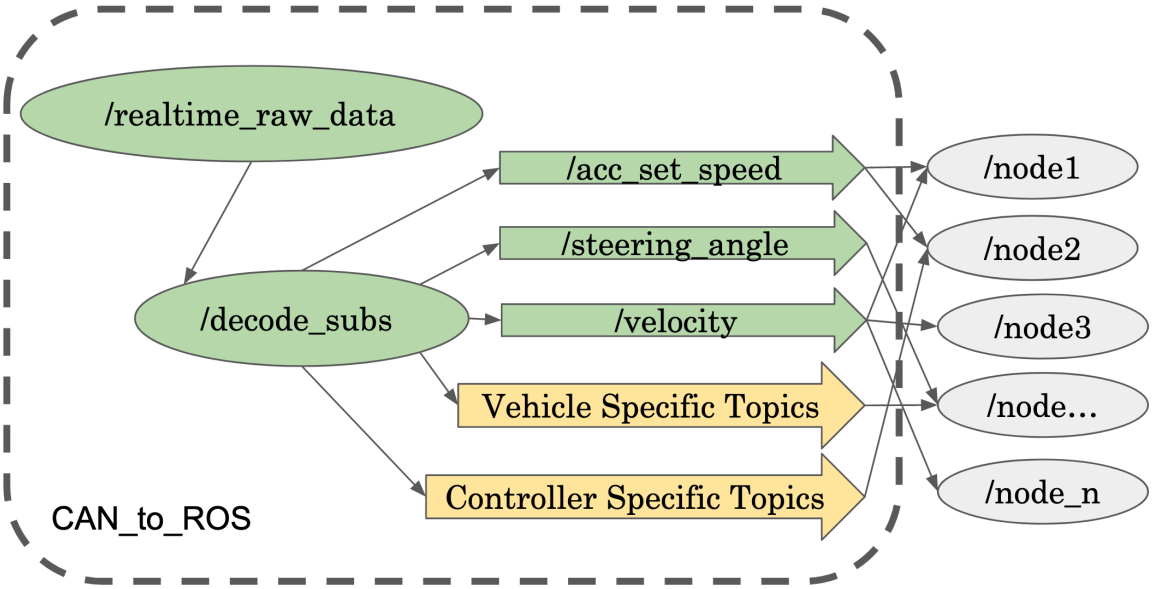


Figure 9.5: Nodes and some of the topics used in the ROS package. Green nodes will always be changed when underlying vehicle changes. Yellow nodes are dependent on other factors. Gray nodes represent the arbitrary consumers of the vehicle-sensor based data on the ROS network.

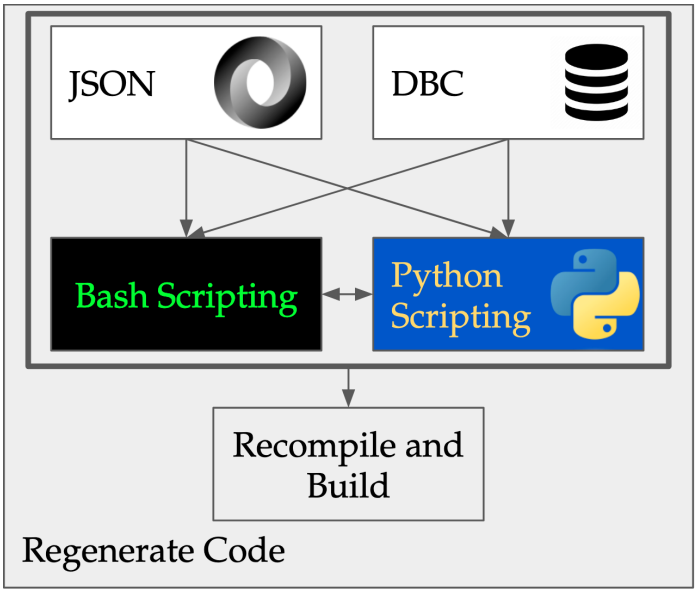


Figure 9.6: The rebuild dependency tree. An agglomeration of Bash and Python scripting, relying on prescribed JSON and DBC files, identifies and rewrites C++ ROS nodes. Once generated, the code is recompiled and built at runtime. Size and complexity is small, so runtime delays for rebuilding are in the order of seconds.

makes the offline use distinct in this work is that it can be used for finding new signal decodings and testing new vehicle software applications.

Similar to what is discussed in 9.3.1, offline playback can be used for a simulated live environment to test signal decodings. The feature is weaker than testing in the online in-vehicle environment, because in that setting you can change and interact with the ground truth as it evolves live. However, instrumented vehicles are not always available to sit in and test, so offline use from recordings is a suitable second-tier replacement.

This software-in-the-loop environment is also useful for properly integrating new software applications into the vehicle software stack. A control algorithm designer can test their implementation with software-in-the-loop to see if their algorithm outputs conform with their expectations; often a piece of the ROS implementation will not be hooked up properly, but it does not require being in-vehicle to solve this issue. Another software application, such as a vehicle monitor tool, can be tested with only software-in-the-loop for similar reasons. The connections between different components can be confirmed correct before taking the leap into field testing.

9.3.3 Limitations

A limitation of this work is its dependency on CAN database (DBC) information, which impacts the breadth of its support between vehicle platforms. These DBC files are tightly guarded proprietary secrets by OEMs, and the CAN database is difficult to recreate on an ad-hoc basis. There is significant but limited support in the open source community [130], [155]. To continue to support an evolving heterogeneous fleet, there is a treadmill which requires new CAN database information continually, and the open source community has not been able to keep up. Without the DBC, most users will realistically not have the time or motivation to create a new one independently. Though the implementation of the in-vehicle network is static for a given vehicle, they can change every model year, and are different between makes, models, and even trims.

9.4 Conclusion and Future Work

This work has shown a model-based code generation tool for developing, testing, and deploying a heterogeneous fleet of vehicles with robotic sensing in ROS. The tool has been demonstrated to successfully integrate across multiple OEM makes and models, streamlining the process involved to transform a traditional passenger vehicle into a research instrument. The primary advancements in the tool are its ability to perform live changes to the decoding files, where the agility of code generation can be leveraged to aid in identifying in-vehicle signals. This helps to build abstraction layers above lower level vehicle-specific details: resulting in a heterogeneous vehicle fleet that can have a common dataframe. Those capabilities build upon the previous features of the package, which provide runtime broadcast of CAN data streams as ROS topics, for consumption be networked computers to perform real-time data analysis.

Future papers will discuss architectures for software applications that can standardize access to data across OEM models. The use of the tool for vehicle control applications, connected vehicle applications, and fleet monitoring systems, is the purpose of ongoing research and analysis.

References

[1] "Transportation statistics annual report 2022," *Transportation Statistics Annual Report (TSAR)*, 2022. DOI: https://doi.org/10.21949/1528354.

[2] *Fast facts on transportation greenhouse gas emissions*, 2020. [Online]. Available: https://www.epa.gov/greenvehicles/fast-facts-transportation-greenhouse-gas-emissions.

[3] M. Barth and K. Boriboonsomsin, "Energy and emissions impacts of a freeway-based dynamic eco-driving system," *Transportation Research Part D: Transport and Environment*, vol. 14, no. 6, pp. 400–410, 2009.

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

[5] E. De Pauw, S. Daniels, L. Franckx, and I. Mayeres, "Safety effects of dynamic speed limits on motorways," *Accident Analysis & Prevention*, vol. 114, pp. 83–89, 2018.

[6] Z. Pu, Z. Li, Y. Jiang, and Y. Wang, "Full bayesian before-after analysis of safety effects of variable speed limit system," *IEEE transactions on intelligent transportation systems*, vol. 22, no. 2, pp. 964–976, 2020.

[7] M. Hadiuzzaman, J. Fang, M. A. Karim, Y. Luo, and T. Z. Qiu, "Modeling driver compliance to vsl and quantifying impacts of compliance levels and control strategy on mobility and safety," *Journal of transportation engineering*, vol. 141, no. 12, p. 04 015 028, 2015.

[8] B. Hellinga and M. Mandelzys, "Impact of driver compliance on the safety and operational impacts of freeway variable speed limit systems," *Journal of transportation engineering*, vol. 137, no. 4, pp. 260–268, 2011.

[9] F. G. Habtemichael and L. de Picado Santos, "Safety and operational benefits of variable speed limits under different traffic conditions and driver compliance levels," *Transportation research record*, vol. 2386, no. 1, pp. 7–15, 2013.

[10] Y. Zhang, M. Quinones-Grueiro, W. Barbour, C. Weston, G. Biswas, and D. Work, "Quantifying the impact of driver compliance on the effectiveness of variable speed limits and lane control systems," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 3638–3644. DOI: 10.1109/ITSC55140.2022.9922471.

[11] A. Nissan and H. N. Koutsopoulosb, "Evaluation of the impact of advisory variable speed limits on motorway capacity and level of service," *Procedia-Social and Behavioral Sciences*, vol. 16, pp. 100–109, 2011.

[12] G. D. Erhardt, S. Roy, D. Cooper, B. Sana, M. Chen, and J. Castiglione, "Do transportation network companies decrease or increase congestion?" *Science advances*, vol. 5, no. 5, eaau2670, 2019.

[13] T. Salem. "Why some cities have had enough of waze." (2018), [Online]. Available: https://www.usnews.com/news/national-news/articles/2018-05-07/why-some-cities-have-had-enough-of-waze (visited on 05/07/2018).

[14] G. Gunter, D. Gloudemans, R. E. Stern, *et al.*, "Are commercially implemented adaptive cruise control systems string stable?" *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[15] V. Milanés and S. E. Shladover, "Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data," *Transportation Research Part C: Emerging Technologies*, vol. 48, pp. 285–300, 2014.

[16] R. E. Stern, S. Cui, M. L. Delle Monache, *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 205–221, 2018.

[17] C. Wu, A. M. Bayen, and A. Mehta, "Stabilizing traffic with autonomous vehicles," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 6012–6018.

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

[19] M. Nice, M. Bunting, A. Richardon, *et al.*, "Enabling mixed autonomy traffic control," *arXiv preprint arXiv:2310.18776*, 2023.

[20] M. Nice, M. Bunting, G. Gunter, W. Barbour, J. Sprinkle, and D. Work, "Sailing cavs: Speed-adaptive infrastructure-linked connected and automated vehicles," *arXiv preprint arXiv:2310.06931*, 2023.

[21] M. Nice, G. Gunter, J. Ji, *et al.*, "A middle way to traffic enlightenment," *Proceedings of the ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPS)*, 2024.

[22] A. G. Sims and K. W. Dobinson, "The sydney coordinated adaptive traffic (scat) system philosophy and benefits," *IEEE Transactions on vehicular technology*, vol. 29, no. 2, pp. 130–137, 1980.

[23] S. Smulders, "Control of freeway traffic flow by variable speed signs," *Transportation Research Part B: Methodological*, vol. 24, no. 2, pp. 111–132, 1990.

[24] M. Papageorgiou, H. Hadj-Salem, J.-M. Blosseville, *et al.*, "Alinea: A local feedback control law for on-ramp metering," *Transportation research record*, vol. 1320, no. 1, pp. 58–67, 1991.

[25] P. Breton, A. Hegyi, B. De Schutter, and H. Hellendoorn, "Shock wave elimination/reduction by optimal coordination of variable speed limits," in *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, IEEE, Singapore: IEEE, 2002, pp. 225–230.

[26] R. L. Bertini, S. Boice, and K. Bogenberger, "Dynamics of variable speed limit system surrounding bottleneck on german autobahn," *Transportation Research Record*, vol. 1978, no. 1, pp. 149–159, 2006.

[27] X.-Y. Lu and S. E. Shladover, "Review of variable speed limits and advisories: Theory, algorithms, and practice," *Transportation Research Record*, vol. 2423, no. 1, pp. 15–23, 2014.

[28] B. Khondaker and L. Kattan, "Variable speed limit: A microscopic analysis in a connected vehicle environment," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 146–159, 2015.

[29] Y. Zhang, M. Quinones-Grueiro, Z. Zhang, *et al.*, "Marvel: Multi-agent reinforcement-learning for large-scale variable speed limits," *arXiv preprint arXiv:2310.12359*, 2023.

[30] S. Wang, M. Shang, M. W. Levin, and R. Stern, "A general approach to smoothing nonlinear mixed traffic via control of autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 146, p. 103 967, 2023.

[31] F. Wu, R. E. Stern, S. Cui, *et al.*, "Tracking vehicle trajectories and fuel rates in phantom traffic jams: Methodology and data," *Transportation Research Part C: Emerging Technologies*, vol. 99, pp. 82–109, 2019.

[32] Y. Sugiyama, M. Fukui, M. Kikuchi, *et al.*, "Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam," *New journal of physics*, vol. 10, no. 3, p. 033 001, 2008.

[33] S.-i. Tadaki, M. Kikuchi, M. Fukui, *et al.*, "Phase transition in traffic jam experiment on a circuit," *New Journal of Physics*, vol. 15, no. 10, p. 103 034, 2013.

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

[35] M. Forster, R. Frank, M. Gerla, and T. Engel, "A cooperative advanced driver assistance system to mitigate vehicular traffic shock waves," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, Toronto, Canada: IEEE, 2014, pp. 1968–1976.

[36] R. M. ALONSO, B. CIUFFO, D. ALVES, *et al.*, "The future of road transport," 2019.

[37] A. Talebpour and H. S. Mahmassani, "Influence of connected and autonomous vehicles on traffic flow stability and throughput," *Transportation research part C: emerging technologies*, vol. 71, pp. 143–163, 2016.

[38] M. Makridis, K. Mattas, B. Ciuffo, *et al.*, "Empirical study on the properties of adaptive cruise control systems and their impact on traffic flow and string stability," *Transportation research record*, vol. 2674, no. 4, pp. 471–484, 2020.

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

[40] B. Ciuffo, K. Mattas, M. Makridis, *et al.*, "Requiem on the positive effects of commercial adaptive cruise control on motorway traffic and recommendations for future automated driving systems," *Transportation research part C: emerging technologies*, vol. 130, p. 103 305, 2021.

[41] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *IEEE Transactions on intelligent transportation systems*, vol. 15, no. 1, pp. 296–305, 2013.

[42] G. J. Naus, R. P. Vugts, J. Ploeg, M. J. van De Molengraft, and M. Steinbuch, "String-stable cacc design and experimental validation: A frequency-domain approach," *IEEE Transactions on vehicular technology*, vol. 59, no. 9, pp. 4268–4279, 2010.

[43] M. Wang, W. Daamen, S. P. Hoogendoorn, and B. van Arem, "Cooperative car-following control: Distributed algorithm and impact on moving jam features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1459–1471, 2015.

[44] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[45] J. Yin, T. ElBatt, G. Yeung, *et al.*, "Performance evaluation of safety applications over dsrc vehicular ad hoc networks," in *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, 2004, pp. 1–9.

[46] Q. Xu, T. Mak, J. Ko, and R. Sengupta, "Vehicle-to-vehicle safety messaging in dsrc," in *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, 2004, pp. 19–28.

[47] H. A. Rakha, H. Chen, M. H. Almannaa, R. K. Kamalanathsharma, I. El-Shawarby, and A. Loulizi, "Field testing of eco-speed control using v2i communication," 2016.

[48] M. Chowdhury, M. Rahman, A. Rayamajhi, *et al.*, "Lessons learned from the real-world deployment of a connected vehicle testbed," *Transportation Research Record*, vol. 2672, no. 22, pp. 10–23, 2018.

[49] O. Tonguz, N. Wisitpongphan, F. Bai, P. Mudalige, and V. Sadekar, "Broadcasting in vanet," in *2007 mobile networking for vehicular environments*, IEEE, 2007, pp. 7–12.

[50] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, "Vanet security challenges and solutions: A survey," *Vehicular Communications*, vol. 7, pp. 7–20, 2017.

[51] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A comparative survey of vanet clustering techniques," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 657–681, 2016.

[52] Z. Xu, X. Li, X. Zhao, M. H. Zhang, and Z. Wang, "Dsrc versus 4g-lte for connected vehicle applications: A study on field experiments of vehicular communication performance," *Journal of advanced transportation*, vol. 2017, 2017.

[53] M. L. Delle Monache, J. Sprinkle, R. Vasudevan, and D. Work, "Autonomous vehicles: From vehicular control to traffic control," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, Nice, France: IEEE, 2019, pp. 4680–4696.

[54] M. Sullman, L. Dorn, and P. Niemi, "Eco-driving training of professional bus drivers–does it work?" *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 749–759, 2015.

[55] Z. Wang, Y. Hsu, A. Vu, *et al.*, "Early findings from field trials of heavy-duty truck connected eco-driving system," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand: IEEE, 2019, pp. 3037–3042.

[56] X. Liao, Z. Wang, X. Zhao, *et al.*, "Cooperative ramp merging design and field implementation: A digital twin approach based on vehicle-to-cloud communication," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 5, pp. 4490–4500, 2021.

[57] H.-P. Wu, W.-H. Shen, Y.-L. Wei, H.-M. Tsai, and Q. Xie, "Traffic shockwave mitigation with human-driven vehicles: Is it feasible?" In *Proceedings of the First ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services*, ser. CarSys '16, New York City, New York: ACM, 2016, pp. 38–43, ISBN: 9781450342506.

[58] U. Fugiglando, E. Massaro, P. Santi, *et al.*, "Driving behavior analysis through can bus data in an uncontrolled environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 737–748, 2018.

[59] T. Wakita, K. Ozawa, C. Miyajima, *et al.*, "Driver identification using driving behavior signals," *IEICE Transactions on Information and Systems*, vol. 89, no. 3, pp. 1188–1194, 2006.

[60] J. Carmona, F. García, D. Martín, A. d. l. Escalera, and J. M. Armingol, "Data fusion for driver behaviour analysis," *Sensors*, vol. 15, no. 10, pp. 25 968–25 991, 2015.

[61] S. Thrun, M. Montemerlo, H. Dahlkamp, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[62] R. E. Chandler, R. Herman, and E. W. Montroll, "Traffic dynamics: Studies in car following," *Operations research*, vol. 6, no. 2, pp. 165–184, 1958.

[63] R. K. Bhadani, J. Sprinkle, and M. Bunting, "The cat vehicle testbed: A simulator with hardware in the loop for autonomous vehicle applications," *arXiv preprint arXiv:1804.04347*, 2018.

[64] Y. Wang, G. Gunter, M. Nice, M. L. Delle Monache, and D. B. Work, "Online parameter estimation methods for adaptive cruise control systems," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 288–298, 2020.

[65] T. M. Jochem, D. A. Pomerleau, and C. E. Thorpe, "Maniac: A next generation neurally based autonomous road follower," in *Proceedings of the International Conference on Intelligent Autonomous Systems*, IOS Publishers, 1993, pp. 15–18.

[66] B. Krogh and C. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, IEEE, vol. 3, 1986, pp. 1664–1669.

[67] C. Thorpe and L. Matthies, "Path relaxation: Path planning for a mobile robot," in *OCEANS 1984*, IEEE, 1984, pp. 576–581.

[68] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[69] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[70] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[71] G. Seetharaman, A. Lakhotia, and E. P. Blasch, "Unmanned vehicles come of age: The darpa grand challenge," *Computer*, vol. 39, no. 12, pp. 26–29, 2006.

[72] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*. springer, 2009, vol. 56.

[73] K.-S. Chang, W. Li, P. Devlin, *et al.*, "Experimentation with a vehicle platoon control system," in *Vehicle Navigation and Information Systems Conference, 1991*, IEEE, vol. 2, 1991, pp. 1117–1124.

[74] A. Al Alam, A. Gattami, and K. H. Johansson, "An experimental study on the fuel reduction potential of heavy duty vehicle platooning," in *13th international IEEE conference on intelligent transportation systems*, IEEE, 2010, pp. 306–311.

[75] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, 2008, pp. 29–39.

[76] B. Hull, V. Bychkovsky, Y. Zhang, *et al.*, "Cartel: A distributed mobile sensor computing system," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, 2006, pp. 125–138.

[77] J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen, "Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 4, pp. 568–583, 2010.

[78] D. B. Work, O.-P. Tossavainen, Q. Jacobson, and A. M. Bayen, "Lagrangian sensing: Traffic estimation with mobile devices," in *2009 American Control Conference*, IEEE, 2009, pp. 1536–1543.

[79] A. Tinka, M. Rafiee, and A. M. Bayen, "Floating sensor networks for river studies," *IEEE Systems Journal*, vol. 7, no. 1, pp. 36–49, 2012.

[80] J. I. Huircán, C. Muñoz, H. Young, *et al.*, "Zigbee-based wireless sensor network localization for cattle monitoring in grazing fields," *Computers and Electronics in Agriculture*, vol. 74, no. 2, pp. 258–264, 2010.

[81] J. Bisgaard, M. Muldbak, S. Cornelissen, *et al.*, "Flow-following sensor devices: A tool for bridging data and model predictions in large-scale fermentations," *Computational and Structural Biotechnology Journal*, vol. 18, pp. 2908–2919, 2020.

[82] A. Wang, Y. Machida, P. deSouza, *et al.*, "Leveraging machine learning algorithms to advance low-cost air sensor calibration in stationary and mobile settings," *Atmospheric Environment*, p. 119 692, 2023.

[83] S. F. Varotto, C. Mons, J. H. Hogema, M. Christoph, N. van Nes, and M. H. Martens, "Do adaptive cruise control and lane keeping systems make the longitudinal vehicle control safer? insights into speeding and time gaps shorter than one second from a naturalistic driving study with sae level 2 automation," *Transportation research part C: emerging technologies*, vol. 141, p. 103 756, 2022.

[84] T. A. Dingus, S. G. Klauer, V. L. Neale, *et al.*, "The 100-car naturalistic driving study, phase ii-results of the 100-car field experiment," United States. Department of Transportation. National Highway Traffic Safety . . ., Tech. Rep., 2006.

[85] T. A. Dingus, J. M. Hankey, J. F. Antin, *et al.*, *Naturalistic driving study: Technical coordination and quality control*. Transportation Research Board, 2015.

[86] A. Blatt, J. Pierowicz, M. Flanigan, *et al.*, "Naturalistic driving study: Field data collection," Tech. Rep., 2015.

[87] M. Benmimoun, A. Pütz, A. Zlocki, and L. Eckstein, "Eurofot: Field operational test and impact assessment of advanced driver assistance systems: Final results," in *Proceedings of the FISITA 2012 World Automotive Congress: Volume 9: Automotive Safety Technology*, Springer, 2013, pp. 537–547.

[88] K. Takeda, J. H. Hansen, P. Boyraz, L. Malta, C. Miyajima, and H. Abut, "International large-scale vehicle corpora for research on driver behavior on the road," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1609–1623, 2011.

[89] G. Gunter, M. Nice, M. Bunting, J. Sprinkle, and D. B. Work, "Experimental testing of a control barrier function on an automated vehicle in live multi-lane traffic," in *2022 2nd Workshop on Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities Workshop (DI-CPS)*, IEEE, 2022, pp. 31–35.

[90] Y. Bian, Y. Zheng, W. Ren, S. E. Li, J. Wang, and K. Li, "Reducing time headway for platooning of connected vehicles via v2v communication," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 87–105, 2019.

[91] S. Aoki and R. Rajkumar, "Dynamic intersections and self-driving vehicles," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2018, pp. 320–330.

[92] Y. Li, C. Xu, L. Xing, and W. Wang, "Integrated cooperative adaptive cruise and variable speed limit controls for reducing rear-end collision risks near freeway bottlenecks based on micro-simulations," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 11, pp. 3157–3167, 2017.

[93] E. Grumert, X. Ma, and A. Tapani, "Analysis of a cooperative variable speed limit system using microscopic traffic simulation," *Transportation research part C: emerging technologies*, vol. 52, pp. 173–186, 2015.

[94] J. Ma, X. Li, S. Shladover, *et al.*, "Freeway speed harmonization," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 78–89, 2016.

[95] E. Vinitsky, K. Parvate, A. Kreidieh, C. Wu, and A. Bayen, "Lagrangian control through deep-rl: Applications to bottleneck decongestion," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, Hawaii, USA: IEEE, 2018, pp. 759–765.

[96] E. Santana and G. Hotz, *Learning a driving simulator*, 2016. arXiv: 1608.01230 [cs.LG].

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

[98] M. Quigley, K. Conley, B. Gerkey, *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, Kobe, Japan: ICRA, 2009, p. 5.

[99] P. Bazilinskyy, J. Stapel, C. de Koning, *et al.*, "Graded auditory feedback based on headway: An on-road pilot study," in *Proceedings of the Human Factors and Ergonomics Society Europe Chapter 2017 Annual Conference*, vol. 2018, HFES, 2018, pp. 115–126.

[100] J. Scott and R. Gray, "A comparison of tactile, visual, and auditory warnings for rear-end collision prevention in simulated driving," *Human factors*, vol. 50, no. 2, pp. 264–275, 2008.

[101] A. H. Jamson, F. C. Lai, and O. M. Carsten, "Potential benefits of an adaptive forward collision warning system," *Transportation research part C: emerging technologies*, vol. 16, no. 4, pp. 471–484, 2008.

[102] G. Abe and J. Richardson, "Alarm timing, trust and driver expectation for forward collision warning systems," *Applied ergonomics*, vol. 37, no. 5, pp. 577–586, 2006.

[103] B. Seppelt and J. Lee, "Making adaptive cruise control (acc) limits visible," *International journal of human-computer studies*, vol. 65, no. 3, pp. 192–205, 2007.

[104] S. H. Fairclough, A. J. May, and C. Carter, "The effect of time headway feedback on following behaviour," *Accident Analysis & Prevention*, vol. 29, no. 3, pp. 387–397, 1997.

[105] M. Risto and M. Martens, "Supporting driver headway choice: The effects of discrete headway feedback when following headway instructions," *Applied ergonomics*, vol. 45, no. 4, pp. 1167–1173, 2014.

[106] A. Loulizi, Y. Bichiou, and H. Rakha, "Steady-state car-following time gaps: An empirical study using naturalistic driving data," *Journal of advanced transportation*, vol. 2019, p. 8, 2019.

[107] Y. Wang, G. Gunter, M. Nice, and D. B. Work, "Estimating adaptive cruise control model parameters from on-board radar units," *arXiv preprint arXiv:1911.06454*, 2019.

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

[109] P. Ioannou, Z. Xu, S. Eckert, D. Clemons, and T. Sieja, "Intelligent cruise control: Theory and experiment," in *Proceedings of 32nd IEEE Conference on Decision and Control*, Dec. 1993, 1885–1890 vol.2. DOI: 10.1109/CDC.1993.325521.

[110] B. Besselink and K. H. Johansson, "String stability and a delay-based spacing policy for vehicle platoons subject to disturbances," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4376–4391, Sep. 2017, ISSN: 0018-9286. DOI: 10.1109/TAC.2017.2682421.

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

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

[113] M. R. Flynn, A. R. Kasimov, J.-C. Nave, R. R. Rosales, and B. Seibold, "Self-sustained nonlinear waves in traffic flow," *Physical Review E*, vol. 79, no. 5, p. 056 113, 2009.

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

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

[116] A. Kesting, M. Treiber, and D. Helbing, "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4585–4605, 2010.

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

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

[119] J. W. Lee, G. Gunter, R. Ramadan, *et al.*, "Integrated framework of dynamics, instabilities, energy models, and sparse flow controllers," in *Proceedings of the Workshop on CPS Data for Transportation and Smart cities with Human-in-the-loop*, 2021.

[120] F.-C. Chou, M. Gibson, A. Bhadani Rahul and Bayen, and J. Sprinkle, "Reachability analysis for followerstopper: Safety analysis and experimental results," in *Proceedings of 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

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

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

[123] L. Zhao and A. A. Malikopoulos, "Enhanced mobility with connectivity and automation: A review of shared autonomous vehicle systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 1, pp. 87–102, 2020.

[124] D. C. Gazis, R. Herman, and R. B. Potts, "Car-following theory of steady-state traffic flow," *Operations research*, vol. 7, no. 4, pp. 499–505, 1959.

[125] J. C. Herrera and A. M. Bayen, "Incorporation of lagrangian measurements in freeway traffic state estimation," *Transportation Research Part B: Methodological*, vol. 44, no. 4, pp. 460–481, 2010.

[126] S. Kato, S. Tsugawa, K. Tokuda, T. Matsui, and H. Fujii, "Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications," *IEEE Transactions on intelligent transportation systems*, vol. 3, no. 3, pp. 155–161, 2002.

[127] S. E. Shladover, C. A. Desoer, J. K. Hedrick, *et al.*, "Automated vehicle control developments in the path program," *IEEE Transactions on vehicular technology*, vol. 40, no. 1, pp. 114–130, 1991.

[128] M. W. Nice, M. Bunting, J. Sprinkle, and D. Work, "Middleware for a heterogeneous cav fleet," in *2023 5th Workshop on Design Automation for CPS and IoT (DESTION)*, 2023.

[129] A. Richardson, M. W. Nice, M. Bunting, *et al.*, "Analysis of a runtime data sharing architecture over lte for a heterogeneous cav fleet," in *The 3rd Workshop on Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities (DI-CPS)*, 2023.

[130] R. Bhadani, M. Bunting, M. Nice, *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.

[131] D. Pickem, P. Glotfelter, L. Wang, *et al.*, "The robotarium: A remotely accessible swarm robotics research testbed," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 1699–1706.

[132] M. Bunting, M. W. Nice, D. Work, J. Sprinkle, and R. Golata, "Edge-based privacy of naturalistic driving data collection," in *2023 ACM/IEEE 14th International Conference on Cyber-Physical Systems (ICCPS)*, 2023.

[133] J. Ma, J. Hu, E. Leslie, F. Zhou, P. Huang, and J. Bared, "An eco-drive experiment on rolling terrains for fuel consumption optimization with connected automated vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 100, pp. 125–141, 2019.

[134] F. Bai and H. Krishnan, "Reliability analysis of dsrc wireless communication for vehicle safety applications," in *2006 IEEE intelligent transportation systems conference*, IEEE, 2006, pp. 355–362.

[135] M. W. Nice, M. Bunting, G. Zachar, *et al.*, "Parameter estimation for decoding sensor signals," in *2023 ACM/IEEE 14th International Conference on Cyber-Physical Systems (ICCPS)*, 2023.

[136] T. L. Swetnam, P. B. Antin, R. Bartelme, *et al.*, "Cyverse: Cyberinfrastructure for open science," *bioRxiv*, pp. 2023–06, 2023.

[137] S. P. Hoogendoorn, W. Daamen, R. G. Hoogendoorn, and J. W. Goemans, "Assessment of dynamic speed limits on freeway a20 near rotterdam, netherlands," *Transportation Research Record*, vol. 2380, no. 1, pp. 61–71, 2013. DOI: 10.3141/2380-07. eprint: https://doi.org/10.3141/2380-07. [Online]. Available: https://doi.org/10.3141/2380-07.

[138] M. Papageorgiou, E. Kosmatopoulos, and I. Papamichail, "Effects of variable speed limits on motorway traffic flow," *Transportation Research Record*, vol. 2047, no. 1, pp. 37–48, 2008.

[139] R. Chen, T. Zhang, and M. W. Levin, "Effects of variable speed limit on energy consumption with autonomous vehicles on urban roads using modified cell-transmission model," *Journal of Transportation Engineering, Part A: Systems*, vol. 146, no. 7, p. 04 020 049, 2020.

[140] Y. Han, D. Chen, and S. Ahn, "Variable speed limit control at fixed freeway bottlenecks using connected vehicles," *Transportation Research Part B: Methodological*, vol. 98, pp. 113–134, 2017.

[141] S. E. Shladover, D. Su, and X.-Y. Lu, "Impacts of cooperative adaptive cruise control on freeway traffic flow," *Transportation Research Record*, vol. 2324, no. 1, pp. 63–70, 2012.

[142] M. Yu and W. D. Fan, "Optimal variable speed limit control in connected autonomous vehicle environment for relieving freeway congestion," *Journal of Transportation Engineering, Part A: Systems*, vol. 145, no. 4, p. 04 019 007, 2019.

[143] Y. Li, H. Wang, W. Wang, S. Liu, and Y. Xiang, "Reducing the risk of rear-end collisions with infrastructure-to-vehicle (i2v) integration of variable speed limit control and adaptive cruise control system," *Traffic injury prevention*, vol. 17, no. 6, pp. 597–603, 2016.

[144] A. Alan, C. R. He, T. G. Molnar, J. C. Mathew, A. H. Bell, and G. Orosz, "Integrating safety with performance in connected automated truck control: Experimental validation," *IEEE Transactions on Intelligent Vehicles*, 2023.

[145] M. Jerbi, P. Marlier, and S. M. Senouci, "Experimental assessment of v2v and i2v communications," in *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, IEEE, 2007, pp. 1–6.

[146] T. D. of Transportation, *Interstate 24 smart corridor*. [Online]. Available: https://www.tn.gov/tdot/projects/region-3/i-24-smart-corridor.html.

[147] G. Gunter and D. Work, "Safe driving with control barrier functions in mixed autonomy traffic when cut-ins occur," in *2022 European Control Conference (ECC)*, 2022, pp. 411–416. DOI: 10.23919/ECC55457.2022.9838155.

[148] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[149] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*, IEEE, 2019, pp. 3420–3431.

[150] M. Delle Monache, C. Pasquale, M. Barreau, and R. Stern, "New frontiers of freeway traffic control and estimation," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, IEEE, 2022, pp. 6910–6925.

[151] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL].

[152] T. W. Lochrane, H. Al-Deek, D. J. Dailey, and J. Bared, "Using a living laboratory to support transportation research for a freeway work zone," *Journal of Transportation Engineering*, vol. 140, no. 7, p. 04 014 024, 2014.

[153] D. Gloudemans, Y. Wang, J. Ji, G. Zachar, W. Barbour, and D. B. Work, "I-24 motion: An instrument for freeway traffic science," *arXiv preprint arXiv:2301.11198*, 2023.

[154] P. Morley, A. Warren, E. Rabb, S. Whitsitt, M. Bunting, and J. Sprinkle, "Generating a ROS/JAUS bridge for an autonomous ground vehicle," in *Proceedings of the 2013 ACM workshop on Domain-specific modeling*, Indianapolis,Indiana: ACM, 2013, pp. 13–18.

[155] Comma.ai, *Opendbc*, https://github.com/commaai/opendbc, 2022.

[156] L. L. Bello and W. Steiner, "A perspective on ieee time-sensitive networking for industrial communication and automation systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019.

[157] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "Librecan: Automated can message translator," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2283–2300.

[158] M. Marchetti and D. Stabili, "Read: Reverse engineering of automotive data frames," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, 2018.

[159] P. Ngo, J. Sprinkle, and R. Bhadani, "Canclassify: Automated decoding and labeling of can bus signals," 2022.