Experimental testing of a control barrier function on an automated vehicle in live multi-lane traffic

George Gunter[1], Matthew Nice[1], Matt Bunting[2], Jonathan Sprinkle[2], Daniel B. Work[1]

***Abstract*—This paper experimentally tests an implementation of a *control barrier function* (CBF) designed to guarantee a minimum time-gap in car following on an *automated vehicle* (AV) in live traffic, with a majority occurring on freeways. The CBF supervises a nominal unsafe PID controller on the AV's velocity. The experimental testing spans two months of driving, of which 1.9 hours of data is collected in which the CBF and nominal controller are active. We find that violations of the guaranteed minimum time-gap are observed, as measured by the vehicle's on-board radar unit. There are two distinct causes of the violations. First, in multi-lane traffic, *Cut-ins* from other vehicles represent external disturbances that can immediately violate the minimum guaranteed time gap provided by the CBF. When cut-ins occur, the CBF does eventually return the vehicle to a safe time gap. Second, even when cut-ins do not occur, system model inaccuracies (e.g., sensor error and delay, actuator error and delay) can lead to violations of the minimum time-gap. These violations are small relative to the violations that would have occurred using only the unsafe nominal control law.**

I. Introduction

Control barrier functions (CBFs) are used to design control schemes which are guaranteed to satisfy safety properties [1]–[3]. CBFs are employed as supervisory controllers to add safety guarantees to nominal control laws that are designed for performance, but do not provide safety guarantees.

A number of works have explored CBFs for controlling *automated vehicles* (AVs). In the works [1], [4], a minimum time-gap safety property is proposed and the corresponding CBF is derived. The resulting CBF safely supervises an unsafe quadratic program both in simulation and experimentally on scale-model cars [5]. The work [6] demonstrates robustness properties when *zeroing* CBFs (zCBFs) are used. The works [7], [8] explore how CBFs can be paired with reinforcement-learning based controllers to achieve safety on otherwise unsafe controllers.

This paper explores the problem of applying a zCBF to supervise an unsafe velocity controller on a full scale commercial vehicle. We implement an unsafe nominal PID velocity controller to maintain a constant velocity, and supervise it with a zCBF designed to guarantee a minimum time-gap is always maintained. The control system is implemented on a 2021 Toyota Rav4 and driven in multi-lane traffic over a two month span. We log the performance of the zCBF using data from the vehicle's CAN bus including space gap data from the forward-facing radar unit. We process this data to understand the extent to which the minimum time-gap safety property of the zCBF is maintained.

The main contribution of this work is to describe the performance of a zCBF designed to maintain a minimum time-gap on a full scale vehicle platform operating in live traffic. We find that the minimum time-gap safety property is violated due to two different causes. First, lane changes from other vehicles cause abrupt changes in the time gap, which can immediately violate the minimum time-gap by a wide margin. When these violations occur, the zCBF returns the vehicle back to a safe state over time. Second, violations can occur even when following a single vehicle (no cut-in), due to a combination of system modeling errors. The smallest time-gap (most violated) observed without a cut-in is 1.17 seconds. Although the zCBF in implementation does not satisfy the designed safety property, it significantly improves the safety of the nominal PID velocity control law, which would otherwise cause collisions.

The remainder of this article is organized as follows. In Section II, we provide background on CBFs, define the time-gap safety property, and provide the system model. Section III describes the experimental setup including the hardware and software implementation. Section IV presents the experimental results of the zCBF operating in live traffic, and in Section V we discuss the findings and highlight future research directions.

II. Preliminaries and background

Here we review preliminary concepts related to the design of CBFs. First, we present background on zeroing control barrier functions (zCBFs) (see [1], [4] for more detail). Next, we provide the system model, specify the minimum time gap safety property, and provide the resulting zCBF.

A. Zeroing control barrier functions

Here we describe the use of zeroing control barrier functions for safe control. Consider the following *control affine* system:

$$\dot{x}(t) = f(x) + g(x)u, \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the system state, $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ are the system dynamics which cannot be directly controlled, $g(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ are the controlled dynamics, and $u \in \mathbb{R}^1$ is a control input.

[1]G. Gunter, M. Nice, and D. Work are with the Department of Civil and Environmental Engineering and Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN.

[2]M. Bunting and J. Sprinkle are with the Department of Computer Science and Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN.

*Corresponding author is G. Gunter: george.l.gunter@vanderbilt.edu

Let a *safety property* by defined as follows:

$$h(x) \geq 0, \tag{2}$$

where $h(x) : \mathbb{R}^n \rightarrow \mathbb{R}^1$ specifies a safe state when non-negative and an unsafe state when negative. Let $\mathcal{C}_h$ be the set of all system states x which satisfy (2).

Control barrier functions are used to derive control laws which render the system state *forward-invariant* with respect to a specified safety property. This means:

$$h(x(0)) \geq 0 \implies h(x(t)) \geq 0, \forall t > 0. \tag{3}$$

If the following condition is met, then forward invariance of the safety property will be achieved [9], [10]:

$$\dot{h}(x) \geq -\alpha(h(x)), \forall x \in \mathcal{C}_h, \tag{4}$$

where $\dot{h}(x)$ is the time-derivative of h at a state x, and $\alpha(\cdot)$ is a class-κ function [1].

More condensed:

$$\dot{h}(x) \geq -\alpha(h(x)), \forall x \implies h(x(t)) \geq 0, \forall t > 0 \tag{5}$$

The term $\dot{h}(x)$ can be calculated using (1):

$$\begin{aligned} &\dot{h}(x) = L_f h(x) + L_g h(x) = \\ &\langle \nabla h(x), f(x) \rangle + \langle \nabla h(x), g(x)u \rangle, \end{aligned} \tag{6}$$

where $L_f h(x)$ and $L_g h(x)$ are the Lie derivatives of $h(x)$ with respect to $f(x)$ and $g(x)$ respectively, $\nabla h(x)$ is the gradient of $h(x)$ with respect to x, and $\langle \cdot, \cdot \rangle$ is an inner product.

By combining (6) and (3), the *safety-kernel* $\mathcal{K}_{\text{safe}}(x)$, which consists of all control inputs that will achieve forward-invariance of the safety property, is derived:

$$\begin{aligned} &\mathcal{K}_{\text{safe}}(x) = \\ &\{u : \langle \nabla h(x), f(x) \rangle + \langle \nabla h(x), g(x)u \rangle \geq -\alpha(h(x))\}. \end{aligned} \tag{7}$$

If the control input at each state x is chosen from $\mathcal{K}_{\text{safe}}(x)$, and $h(x(0)) \geq 0$ then safety is maintained for all following time.

Condensed:

$$u(t) \in \mathcal{K}_{\text{safe}}(x(t)) \forall t, h(x(0)) \geq 0 \implies h(x(t)) \geq 0, \forall t > 0 \tag{8}$$

B. Using zCBFs for safe control of AVs

System model. For automated vehicle velocity control, we assume the following dynamics [11]–[13]:

$$\dot{x} = f(x) + g(x)u = \begin{bmatrix} \dot{v}_f \\ \dot{v}_l \\ \dot{s} \end{bmatrix} = \begin{bmatrix} 0 \\ a_l \\ v_l - v_f \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u, \tag{9}$$

Misc:

$$x = \begin{bmatrix} v_f \\ v_l \\ s \end{bmatrix}, \tag{10}$$

where $x = [v_f, v_l, s] \in \mathbb{R}^3$ is the system state, v_f is the speed of the AV, v_l is the speed of the lead vehicle, s is the space gap between the AV and the lead vehicle, a_l is the acceleration of the lead vehicle, and $u \in \mathbb{R}^1$ is the commanded acceleration of the AV.

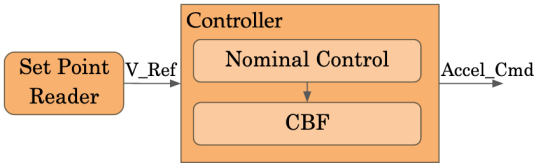

Fig. 1. Flow chart for the acceleration-based controller used to test the zCBF functionality.

Safety specification. We specify safety as a minimum time-gap that the vehicle must maintain, following the works [4], [4], [6]. Given the system model (10), the minimum time-gap safety property is specified as:

$$h_{TG}(x) = s - t_{\text{min}} v_f \geq 0, \tag{11}$$

where $h_{TG}(x) \geq 0$ is the safety property, and t_{min} is the minimum time-gap.

We use a linear class-κ function of the following form:

$$\alpha(h(x)) = kh(x), \tag{12}$$

where k is a non-negative scalar. This choice leads to the following safety kernel:

$$\begin{aligned} &\mathcal{K}_{\text{safe},TG}(x) = \\ &\{u : v_l - v_f - t_{\text{min}} u \geq -k\left(s - t_{\text{min}} v_f\right)\}, \end{aligned} \tag{13}$$

which can subsequently be rewritten as a direct inequality between the control input u and the system state x as:

$$u \leq \left(\frac{1}{t_{\text{min}}}\right)(v_l - v_f) + \left(\frac{k}{t_{\text{min}}}\right)\left(s - t_{\text{min}} v_f\right). \tag{14}$$

In our experimental work, we use parameter values of $k = 0.1$ and $t_{min} = 2.0s$. The time-gap was chosen to correspond with the common "two-second rule", while k was chosen from a combination of values examined in other works [4], [6] and driver comfort levels. Any $k \geq 0$ satisfies the forward-invariance condition in Equation (12), but differing values lead to differing responses to changes in the leading vehicle's driving, such as how quickly the vehicle recovers back to the minimum time-gap after a cut-in. It is likely that other values of k are reasonable.

The nominal velocity controller is a PID cruise control that maintains a constant speed set by the driver. It is described in the hardware and software architecture in the next Section.

III. Experimental Methods and Approach

This section explains how the hardware and software architecture is set up to send commands to the vehicle, and how the experiments are structured.

Hardware/Software Architecture. A model-based approach is used to directly specify the controller designs for both the nominal controller, and the zCBF. Through the use of open-source tools [14], [15] and MATLAB's software synthesis tools, we deploy ROS implementations in C++.

The nominal controller is a user-set cruise controller that uses the same interface as the stock *adaptive cruise control*

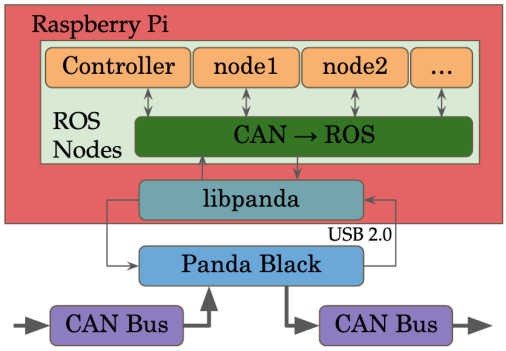

Fig. 2. Summary of the computational architecture of vehicle control on our platform.

(ACC) (i.e., we use the stock interfaces, but not the stock control system). The nominal controller records the reference velocity v_r from the stock ACC unit, and outputs the reference acceleration u through a PID controller, in the form of $u_{nom} = \Gamma(v_f, \dot{v}_f, v_r)$ where v_f, $\dot{v}_f$ are as in (10), and v_r is taken from the stock vehicle ACC unit. The model for $\Gamma(\cdot)$ is obtained through system identification for purposes of rapid velocity tracking, with no regard to passenger comfort. As this controller has no knowledge of v_l or s, it is an unsafe controller that can result in collision if allowed to operate behind a slower vehicle. This setup helps structurally to separate the behavior of the nominal controller and the intervention from the safety supervisor, especially in anticipating what should happen in common multi-vehicle scenarios on the roadway.

Figure 1 shows the signal flow and node architecture from the set point reference to the acceleration command sent to the vehicle. The set point ROS node reads in the set point from the cruise control system, then passes that value to the nominal controller. The nominal controller outputs an acceleration value, which is passed to the zCBF supervisory controller that bounds the acceleration command sent to the adaptive cruise control ECU on the vehicle. In this model, each node operates at a set frequency: 20Hz for the nominal controller, and 20Hz for the zCBF. Thus there is a worst case delay of up to 0.1s for data throughput if these models, during execution, are not perfectly synchronized in their dataflow.

Sending Acceleration Commands. Acceleration commands are sent to the vehicle using a custom software stack outlined in Figure 2 including the Comma.ai Black Panda device, libpanda [14], can2ros [15], and various ROS nodes, all running on a raspberry pi. To send acceleration commands to the vehicle, the acceleration commands are intercepted from the CAN bus by the Panda Black and replaced by commands from our controller. The vehicle's stock engine control unit transforms acceleration commands into brake and throttle inputs, with dynamics that are outside the scope of this paper.

Experimental setup and data processing. We implement the zCBF onto a commercially available Toyota Rav4 and drive in traffic over the course of two months from December 2021 through January 2022. The drives occur at different times of the day and on multiple roadways in Nashville, TN. The majority of the dives occur on freeways, with some additional drives occurring on other lower speed roadways. In all cases a trained driver supervised the vehicle and was prepared to assume control in the event of difficult driving conditions.

Data is recorded from the vehicle's on-board sensors using libpanda [14], and analyzed through strym [16]. The inter-vehicle spacing and speed-difference are measured using the vehicle's radar unit. Sample frequency differences between measurements are corrected by resampling all data at 10 hz using a univariate spline technique.

Given recorded data, we extract a collection of distinct car-following timeseries. Each car following timeseries begins when a vehicle enters in front of the AV, either through a lane change into the lane of the AV, or by becoming in range of the forward facing radar (250 m). The timeseries ends when the lead vehicle leaves the lane, leaves the range of the radar, or a new vehicle cuts into the lane with a smaller spacing. We categorize each car-following timeseries based on if the timeseries begins with a large violation of the minimum time gap for example due a cut-in (referred to as *initially unsafe* car following), or if the timeseries begins with the minimum time gap safety property satisfied (referred to as *initially safe* car following). For an initially unsafe timeseries, we distinguish safety violations occurring while recovering from the initial violation, vs those that occur after the CBF returns the system to a safe state but then a later violation occurs (still tracking the same vehicle). This allows us to distinguish between safety violations due to cut-ins and those due to errors in controlling the AV to maintain safety in response to the vehicle ahead.

$$u_{\text{cmd}} = \min(u_{nom}, \left(\frac{1}{t_{\min}}\right)(v_l - v_f) + \left(\frac{k}{t_{\min}}\right)(s - t_{\min} v_f)) \tag{15}$$

IV. Results

In this section we describe the results of experimental testing of the zCBF implementation on a live AV.

Over the collection period 1.9 hours of active control were measured. 106 total distinct car-following timeseries were recorded. 55 of these timeseries begin in an initially unsafe state (e.g., due to a cut-in), while the other 51 timeseries begin in a safe state.

Figure 3 illustrates a single continuous drive in which a collection of both initially unsafe and initially safe car following timeseries are extracted. Car following that begins in an initially unsafe state (due to cut-ins) are colored red, while car following timeseries that begin in a safe state are shown in blue. In freeway driving, lane changes create a mix of the two types of car following timeseries. The top subplot of Figure 3 shows measured time-gap (space-gap divided by speed), the middle subplot shows space-gap, and the bottom subplot shows relative speed difference (lead vehicle speed minus AV speed). Gaps in the timeseries correspond to when no lead vehicle is within following distance, and abrupt changes occur when a leading vehicle merges in or out of the AV's lane.

Initially unsafe car following In live traffic, cut-ins that violate the minimum time-gap safety property are observed.

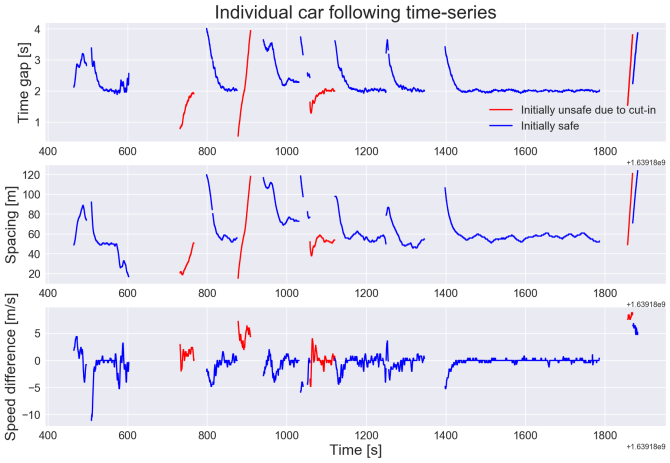

Fig. 3. Extracted instances of car following timeseries, split between initially unsafe (red) and initially safe (blue) car following.

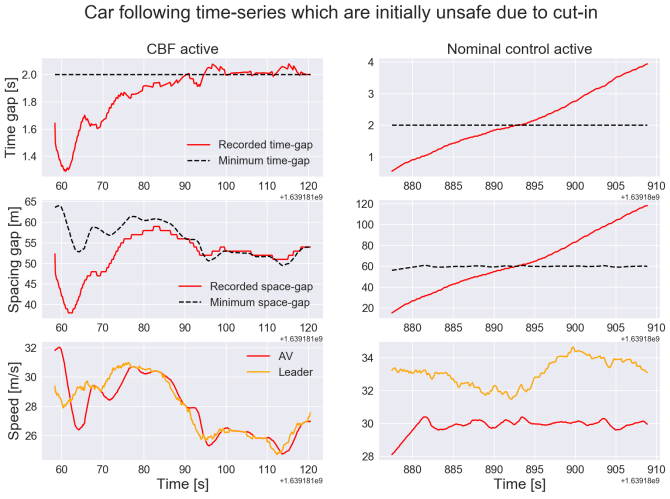

Fig. 4. Two different responses when an initially unsafe state appears. Left: the CBF is active and recovers the system to a safe state; Right: the nominal control is active and the system recovers to a safe state.

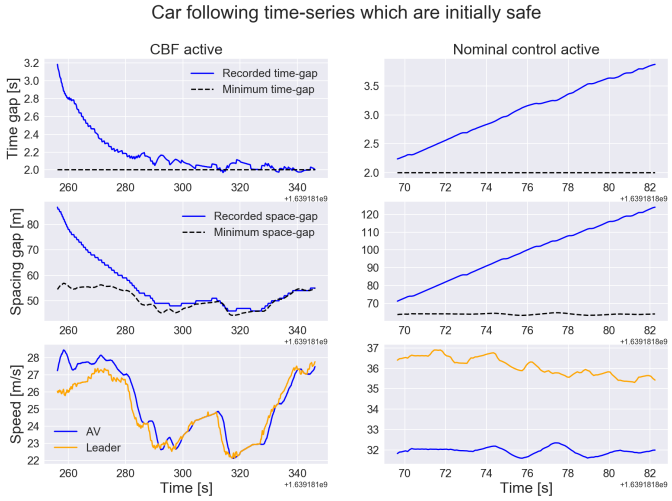

Fig. 5. Two different responses when an initially state appears. Left: the CBF is active and prevents large violations of the safety property; Right: the nominal control remains active and the system does not violate the safety property.

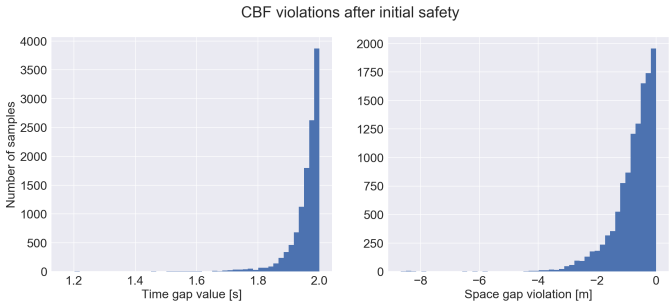

Fig. 6. A distribution of measurements which violate the minimum time-gap property during approach events.

These violations are unavoidable by controlling only the AV, but we are interested in how the AV recovers from these events.

Figure 4 shows a comparison between two different initially unsafe car following timeseries with respect to time gap, spacing, and speed. The left portion of the plot shows an event in which a slower moving vehicle cuts-in, causing the zCBF to intervene and adjust the AV speed. On the right, the cut-in is from a faster moving vehicle, leading the zCBF to remain inactive. In both scenarios the supervisory nature of the zCBF *recovers* the system state back towards the minimum time-gap.

Initially safe car following. For car-following in which the timeseries begins in a safe state, violations of the minimum time-gap safety property are observed. This is likely due to a combination of modeling inaccuracies (sensor error & delay, actuator error & delay), and other external disturbances such as changes in the road grade.

Not all initially safe car following timeseries exhibit violations of the safety property. In Figure 5 two different approach events are shown, with time-gap, spacing, and speed all plotted. On the left, the AV approaches a slower moving vehicle and must activate the zCBF to adjust its speed. The time-gap approaches 2.0 seconds as the AV slows to match the leader and at times dips below. On the right the AV follows a faster moving vehicle. Since the lead vehicle never nears the minimum time-gap boundary the nominal speed controller is allowed to stay active.

We consider measurements across all initially safe car-following timeseries. In initially unsafe timeseries, if the system recovers the system back to a time-gap greater than 2.0 seconds, we add all following measurements from that timeseries to the collection of initially safe timeseries. Figure 6 shows statistics across all such initially safe timeseries.

On the left subplot time-gap measurements which violate the safety property (less than 2.0 seconds) are shown. The minimum such observed time-gap is 1.17 seconds, meaning that the safety property was violated by at most 0.83 seconds across all initially safe timeseries. On the right subplot the distances by which the safety property are violated are shown.

The largest such violation was 8.7 meters.

Due to cut-ins from vehicles in adjacent lanes and modeling inaccuracies/external disturbances violations to the minimum time gap safety property are observed. Despite this, the implemented zCBF was able successfully stop the nominal control law, which focused only on speed, from colliding with other vehicles.

V. Conclusions and future work

This work experimentally tests an implementation of a zeroing control barrier function designed to maintain a minimum time gap on a full scale automated vehicle operating in traffic on freeways and other roads. We observe 55 cut-in events that immediately violate the minimum time headway. The zeroing CBF returns the vehicle to a safe state in all instances. We observe other violations of the safety property that are not possible if the assumed system model is error free. These violations are likely due to a combination of sensor, actuator errors, and external disturbances.

In our future work, we are interested to explore if more accurate system models that account for, e.g., actuator delay, are implementable on real vehicle platforms. A current challenge is that such models tend to require higher order information from the lead vehicle that may be difficult to estimate from radar data. Understanding the practical limits of safety guarantees on experimental platforms remains an important future direction.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) award number CID DE-EE0008872. The views expressed herein do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Additional support is provided by the DOT Eisenhower Fellowship, and NSF awards 2135579 and 1837652. George Gunter is supported by an NSF Graduate Fellowship.

References

[1] 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.

[2] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[3] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2004, pp. 477–492.

[4] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 6271–6278.

[5] A. Mehra, W.-L. Ma, F. Berg, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Adaptive cruise control: Experimental validation of advanced controllers on scale-model cars," in *2015 American control conference (ACC)*. IEEE, 2015, pp. 1411–1418.

[6] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[7] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[8] J. Choi, F. Castaneda, C. J. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions," *arXiv preprint arXiv:2004.07584*, 2020.

[9] M. Nagumo, "Über die lage der integralkurven gewöhnlicher differentialgleichungen," *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, vol. 24, pp. 551–559, 1942.

[10] F. Blanchini and S. Miani, *Set-theoretic methods in control*. Springer, 2008.

[11] J. Monteil, M. Bouroche, and D. J. Leith, "$\mathcal{L}_2$ and $\mathcal{L}_\infty$ stability analysis of heterogeneous traffic with application to parameter optimization for the control of automated vehicles," *IEEE Transactions on Control Systems Technology*, 2018.

[12] S. Cui, B. Seibold, R. Stern, and D. B. Work, "Stabilizing traffic flow via a single autonomous vehicle: Possibilities and limitations," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1336–1341.

[13] M. L. Delle Monache, T. Liard, A. Rat, R. Stern, R. Bhadani, B. Seibold, J. Sprinkle, D. B. Work, and B. Piccoli, "Feedback control algorithms for the dissipation of traffic waves with autonomous vehicles," in *Computational Intelligence and Optimization Methods for Control Engineering*. Springer, 2019, pp. 275–299.

[14] 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.

[15] 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.

[16] R. Bhadani, J. Sprinkle, G. Lee, M. Nice, G. Gunter, and S. Elmadani, "Strym: A python package for real-time can data logging, analysis and visualization to work with usb-can interface," 2020. [Online]. Available: https://github.com/jmscslgroup/strym