Enhancing the Performance of a Safe Controller Via Supervised Learning for Truck Lateral Control

Yuxiao Chen[1]
Department of Mechanical and Civil Engineering,
California Institute of Technology,
Pasadena, CA 91106
e-mail: chenyx@caltech.edu

Ayonga Hereid
Department of Mechanical and
Aerospace Engineering,
Ohio State University,
Columbus, OH 43210
e-mail: hereid.1@osu.edu

Huei Peng
Department of Mechanical Engineering,
University of Michigan,
Ann Arbor, MI 48109
e-mail: hpeng@umich.edu

Jessy Grizzle
Department of Electrical Engineering and
Computer Science,
University of Michigan,
Ann Arbor, MI 48109
e-mail: grizzle@umich.edu

Correct-by-construction techniques, such as control barrier functions (CBFs), can be used to guarantee closed-loop safety by acting as a supervisor of an existing legacy controller. However, supervisory-control intervention typically compromises the performance of the closed-loop system. On the other hand, machine learning has been used to synthesize controllers that inherit good properties from a training dataset, though safety is typically not guaranteed due to the difficulty of analyzing the associated learning structure. In this paper, supervised learning is combined with CBFs to synthesize controllers that enjoy good performance with provable safety. A training set is generated by trajectory optimization that incorporates the CBF constraint for an interesting range of initial conditions of the truck model. A control policy is obtained via supervised learning that maps a feature representing the initial conditions to a parameterized desired trajectory. The learning-based controller is used as the performance controller and a CBF-based supervisory controller guarantees safety. A case study of lane keeping (LK) for articulated trucks shows that the controller trained by supervised learning inherits the good performance of the training set and rarely requires intervention by the CBF supervisor.

[DOI: 10.1115/1.4043487]

Keywords: control barrier function, supervised learning, trajectory optimization

1 Introduction

Correct-by-construction control synthesis has been a promising direction of research that brings formal safety guarantees to controller design. In particular, control barrier functions (CBFs) can be overlaid on the existing controllers so as to impose closed-loop safety in a plug-and-play fashion [1,2]. The key idea in the design of a CBF is to compute a forward invariant set that contains the safe set and excludes the danger set. The CBF can then be implemented in a supervisory control structure to guarantee safety without redesigning the performance controller, hereafter called the "student" controller because it is being "supervised" by the CBF.

As shown in Fig. 1, u_0 denotes the control input from the student controller, which can be designed with any existing method, and u denotes the input signal after the intervention of the supervisory controller. If u_0 respects the safety constraint, then $u = u_0$; otherwise, a "minimal intervention" is applied. Depending on the form of the barrier function, the intervention may be solved through quadratic programming [1,3], mixed integer programming [4], or in other forms.

While safety is assured independently of the choice of student controller, if the student controller is not properly designed, or is designed in a way that is not compatible with the CBF, the CBF may be triggered frequently, leading to undesirable closed-loop performance. In Refs. [4] and [5], when the student controller for adaptive cruise control and obstacle avoidance were not properly designed, the CBF has to intervene frequently and severely, which is not desirable for other performance considerations such as riding comfort.

These examples, on one hand, demonstrate the power of a CBF to provide safety guarantees, but they also show there is room for improvement. The problem that we try to solve in this paper is to design student controller that has safety built in and rarely or never triggers the intervention of the CBF, therefore achieves good performance.

On the other hand, machine learning has been used extensively in dynamic control. Supervised learning has been used to learn a control policy with structure [6,7], deep learning recently was used to generate end-to-end lane keeping (LK) policy, i.e., a mapping directly from the camera pixels to the steering input [8], and reinforcement learning can be used to generate a control policy in an "explore and evaluate" manner [9–11]. However, one major deficit of machine learning is its extreme difficulty for analysis. The number of parameters contained in a neural network can easily reach several thousands, even millions, which makes it practically impossible to analyze. Therefore, the safety of a learning-based controller should rely on other tools, such as reachable sets and barrier functions. In this sense, machine learning and CBFs complement one other.

The existing methods that combine learning with safety guarantee include reachable-set-based learning scheme that can guarantee safety for online learning of a control policy [12,13], a barrier-function-based online learning scheme [14], and a Gaussian process learning [15]. Unlike approaches that aim at guaranteeing safety with learning, such as Refs. [12–15], the method

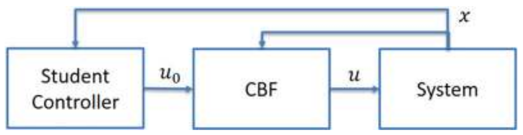

Fig. 1 Block diagram of CBF supervising a "student" controller

[1]Corresponding author.

Contributed by the Dynamic Systems Division of ASME for publication in the Journal of Dynamic Systems, Measurement, and Control. Manuscript received August 23, 2018; final manuscript received April 3, 2019; published online June 3, 2019. Assoc. Editor: Xuebo Zhang.

proposed in this paper separates safety from the performance. The safety guarantee is provided by a CBF, and the supervised learning is used to improve the performance considering the influence of the CBF as a supervisor.

The method we propose is to perform trajectory optimization offline, generate a library consisting of trajectories with good properties, namely, stabilizing an equilibrium, attenuating disturbances, and satisfying a CBF condition. We then use supervised learning to design a student controller that inherits the properties of the trajectory library. With supervised learning, the design of a safe student controller is transformed into the design of safe trajectories, which is much easier, as conceptually shown in Fig. 2. On top of the learning-based controller, the CBF is implemented as a supervisor to formally guarantee safety, as shown in Fig. 3. Since the CBF condition is enforced in the training set, an intervention by the supervisor is rarely triggered. It should be emphasized that the safety is still guaranteed by the CBF and the supervised learning only aims to improve performance under the presence of the CBF.

The main contributions of this paper are the following three points. First, a control barrier function is designed that guarantees safety for any truck lane keeping controller as a supervisory controller. Second, we propose a supervised learning based method to design a student controller that takes the CBF condition into account, is applicable to a large region of initial conditions, and rarely triggers an intervention from the supervisory controller. Third, we provide the stability and set invariance analysis of the learning-based controller under the framework of continuous hold (CH) feedback control. Applying the proposed method, we can provide a safety guarantee for LK control of an articulated truck, while achieving good ride comfort.

The remainder of the paper is structured as follows. We first introduce the truck model and the feedback linearization structure in Sec. 2. Then, we present the sum of squares (SOS) approach for the synthesis of a CBF in Sec. 3. Then, we show the trajectory optimization process with direct collocation that incorporates the CBF condition in Sec. 4. The obtained trajectory library is then used to train a neural network that acts as a trajectory generator, as presented in Sec. 4. The trajectory generator is implemented in a continuous hold control structure with CBF as the supervisor on top of it, which is presented in Sec. 5. Finally, we present the LK problem as an example in Sec. 6 and conclude in Sec. 7.

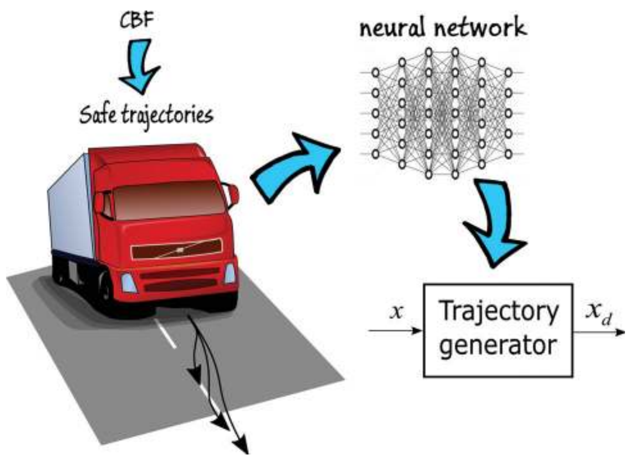

Fig. 2 Learning based trajectory generator

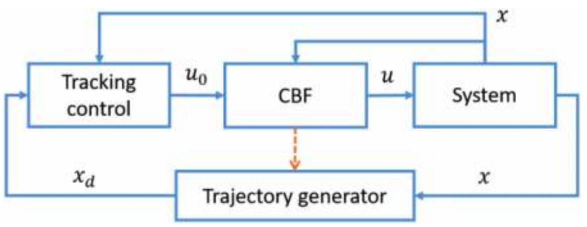

Fig. 3 Structure of the proposed supervisory control

2 Dynamic Model and Virtual Constraint

In this paper, we consider a control affine nonlinear model

$$\begin{aligned}&\dot{x} = f(x) + g(x)u + g_{d1}(x)d_1 + g_{d2}(x)d_2,\\ &x \in \mathbb{R}^n, \quad u \in \mathcal{U} \subseteq \mathbb{R}, \quad d_1 \in \mathcal{D}_1 \subseteq \mathbb{R}^{l1}, \quad d_2 \in \mathcal{D}_2 \subseteq \mathbb{R}^{l2}\end{aligned} \tag{1}$$

where x, u, d_1, and d_2 represent the state, input, measured disturbance, and unmeasured disturbances, respectively.

Remark 1. The unmeasured disturbance d_2 will be countered with the feedback control. In Sec. 3, the CBF constructed is robust for all possible d_2 within the assumed bound. Since unmeasured, d_2 does not appear in feedback linearization.

2.1 Model Assumptions.

The results in this paper are developed under three key assumptions:

Assumption 1. *There exists an output $z = h(x)$ for x within an open subset $\mathcal{S} \in \mathbb{R}^n$, such that for all $d_1 \in \mathcal{D}_1$, z has relative degree ρ, where the relative degree is defined as the integer such that $\forall x \in \mathcal{S}$*

$$\begin{aligned}&\mathcal{L}_g\mathcal{L}_{\bar{f}}^{i-1}h(x) = 0, \quad i = 1, 2, \ldots, \rho - 1;\\ &\mathcal{L}_g\mathcal{L}_{\bar{f}}^{\rho-1}h(x) \neq 0\end{aligned} \tag{2}$$

where

$$\bar{f}(x) = f(x) + g_{d1}(x)d_1 \tag{3}$$

Assumption 2. *It is assumed that when $d_2 = 0$, for all $d_1 \in \mathcal{D}_1$, there exists a unique $\eta_u(d_1) \in \mathcal{U}$ that maintains a unique equilibrium point $x_e \in \mathbb{R}^n$ with $h(x_e) = 0$, denoted as $x_e = \eta_x(d_1)$*

$$\begin{aligned}&f(x_e) + g(x_e)\eta_u(d_1) + g_{d1}(x_e)d_1 = 0,\\ &h(x_e) = 0\end{aligned} \tag{4}$$

Then from feedback linearization, there exists a state transformation

$$\begin{aligned}&\left[\frac{\sigma}{\xi}\right] = \left[\frac{T_1(x)}{T_2(x)}\right] = T(x),\\ &\sigma \in \mathbb{R}^{n-\rho}, \quad \xi = \begin{bmatrix} h(x) \\ \vdots \\ \mathcal{L}_{\bar{f}}^{\rho-1}h(x)\end{bmatrix} = \begin{bmatrix} z \\ \vdots \\ z^{(\rho-1)}\end{bmatrix} \in \mathbb{R}^{\rho}\end{aligned} \tag{5}$$

where T is a bijective diffeomorphism over $\mathcal{S}$, and the transformation satisfies $(\partial T_1/\partial x)g(x) = 0$. Therefore, the dynamics of the "hidden" states σ is represented as

$$\dot{\sigma} = \Gamma(\sigma, \xi) \tag{6}$$

In particular, $\dot{\sigma} = \Gamma(\sigma, 0)$ is the zero dynamics of the system with output z, and there exists a smooth surface $\mathcal{Z} \subset \mathcal{S}$ defined by $\mathcal{Z} := \{x \in \mathcal{S} \mid \xi = 0\}$, which is the zero dynamics manifold.

Assumption 3. *We assume that the zero dynamics of the system under output z is exponentially stable within $\mathcal{S}$.*

Then by Theorem 11.2.3 in Ref. [16], the following feedback linearization controller constructed from z and its derivatives stabilizes the equilibrium x_e:

$$u = -\frac{1}{\mathcal{L}_g\mathcal{L}_{\bar{f}}^{\rho-1}h(x)}\left[k_0\xi_1 + \cdots + k_{\rho-1}\xi_\rho + \mathcal{L}_{\bar{f}}^{\rho}h(x)\right] \tag{7}$$

where $\{k_i\}$ is a set of exponentially stabilizing gains in the sense that the following characteristic equation

$$\lambda^\rho + k_{\rho-1}\lambda^{\rho-1} + \cdots + k_0 = 0 \tag{8}$$

has all of its roots in the open left half plane. See, e.g., Ref. [17] for reference on feedback linearization and zero dynamics.

2.2 Virtual Constraint and Tracking Control. To let the system track a desired trajectory of z, we use the virtual constraint method, originally developed in the robotics literature [18–20], and now appearing more widely. Suppose we want the system to track a desired trajectory $z = h_{\rm des}(t)$, where $h_{\rm des}$ is a ρ times continuously differentiable function. The error states can be obtained by differentiation

$$\begin{aligned} e_1 &= z - h_{\rm des} \\ e_2 &= \dot{z} - \dot{h}_{\rm des} \\ &\vdots \\ e_\rho &= z^{(\rho-1)} - h_{\rm des}^{\rho-1} \end{aligned} \tag{9}$$

Pick $\{k_i\}$ to be a set of stabilizing gains as described in Eq. (8), and let

$$u = -\frac{1}{\mathcal{L}_g\mathcal{L}_f^{\rho-1}h(x)}\left[k_0e_1 + \cdots + k_{\rho-1}e_\rho + \mathcal{L}_{\bar{f}}^{\rho}h(x)\right] \tag{10}$$

When $h_{\rm des}$ is ρ times continuously differentiable and its derivatives are bounded, the feedback linearization control can locally track $h_{\rm des}$ imposed as a virtual constraint of z [21].

The benefit of the virtual constraint approach is that it gives a simple means of parameterizing the desired evolution of the vehicle. Instead of all the states, the desired trajectory is parameterized only by an output z satisfying Assumptions 1 and 3. Later, we will use trajectory optimization to determine the existence of a set of interesting trajectories that can be tracked by considering the full dynamics and the feedback structure.

2.3 Tractor-Semitrailer Models. In this work, we use two models: a design model and a validation model. For validation, we use TRUCKSIM with its impressive 312 states. The literature contains a range of less detailed models that could be considered for control design, ranging from the nonlinear 37-state, physics-based model in Ref. [22], to linear models. To demonstrate the fundamental robustness of the approach followed in this paper, we base the control design on a low-complexity model for an articulated truck adapted from Refs. [22] to [23], namely a four degrees-of-freedom linear model with eight states

$$x = [y \quad v_y \quad \psi \quad r \quad \psi_a \quad r_s \quad \phi \quad p]^{\rm T} \tag{11}$$

where y is the lateral deviation from the lane center to the tractor center of gravity, v_y is the lateral sideslip velocity of the tractor, ψ is the heading angle of the tractor, r is the yaw rate of the tractor, ψ_a is the articulation angle on the fifth wheel (the joint between the tractor and the semitrailer), r_s is the yaw rate of the semitrailer, ϕ is the roll angle, and p is the roll rate, as shown in Fig. 4.

The linear model is expressed in the form of Eq. (1) for consistency

$$\begin{aligned} \dot{x} &= f(x) + g(x)\delta_f + g_{d1}(x)r_d + g_{d2}(x)F_y \\ &= Ax + B\delta_f + E_1r_d + E_2F_y \end{aligned} \tag{12}$$

The input to the system is the steering angle δ_f of the tractor front axle and the disturbances are road curvature r_d and side wind F_y,

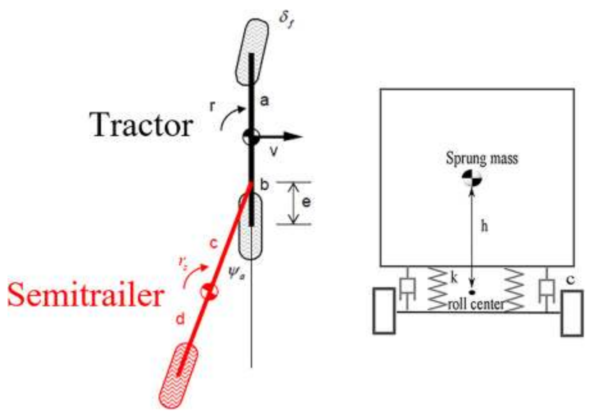

Fig. 4 Lateral-yaw-roll model of articulated truck

where r_d is the measured disturbance, namely, d_1 in Eq. (1) and F_y is the unmeasured disturbance, namely, d_2 in Eq. (1).

A priori, the above linear model is only valid under the following assumptions:

- The longitudinal speed v_x of the truck has small variation.
- Due to the stiff connection on the roll dimension, the roll angle of the tractor and the semitrailer is the same.
- The pitch and the vertical motion are weakly coupled with the lateral, yaw, and roll motion, and are ignored in the model.
- The angles are small, and therefore, the dynamics can be approximated by a linear model.

The simulations performed later in TRUCKSIM support that these assumptions are satisfied in a highway lane keeping scenario.

Remark 2. The methods developed in this paper, including the CBF synthesis, the trajectory optimization, and the continuous-hold controller, all apply to nonlinear models. Hence, for the remainder of the paper, we denote the model as in Eq. (1).

2.4 The Virtual Constraint for the Truck Model. We select the lateral displacement with preview as the output for feedback linearization

$$z = h(x) := y + T_0v_x\psi \tag{13}$$

with T_0 being the preview time, as shown in Fig. 5.

ASSUMPTION 4. *It is assumed that r_d changes slowly comparing to the system dynamics, therefore treated as constant in the trajectory optimization.*

Remark 3. The change of r_d typically takes several seconds, while the time constant of the lateral dynamics is below half a second, which justifies Assumption 4.

The output z so-defined has relative degree 2 for any r_d, i.e.,

$$\mathcal{L}_gh = 0, \quad \mathcal{L}_g\mathcal{L}_{f+g_{d1}r_d}h \neq 0 \tag{14}$$

To be more specific, the output dynamics is

$$\begin{aligned} z &= h(x), \\ \dot{z} &= \mathcal{L}_fh + \mathcal{L}_{g_{d1}}h\cdot r_d, \\ \ddot{z} &= \mathcal{L}_f^2h + \mathcal{L}_g\mathcal{L}_fh\cdot u + \mathcal{L}_f\mathcal{L}_{g_{d1}}h\cdot r_d \end{aligned} \tag{15}$$

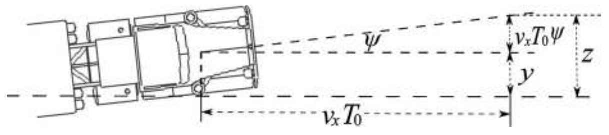

Fig. 5 Preview deviation as output

By Assumption 4, r_d changes slowly compared to the dynamics, therefore, $\dot{r}_d$ is omitted. Since there are eight states but only z and $\dot{z}$ are used in the feedback linearization, six dimensions of the state space are hidden. It is shown that the zero dynamics of the system is exponentially stable, see Appendix A for details. Since $\rho = 2$, the feedback structure in Eq. (10) is essentially a proportional-derivative (PD) controller

$$u = -\frac{1}{\mathcal{L}_g \mathcal{L}_f h(x)} \begin{bmatrix} K_p(z - h_{\text{des}}) + K_d(\dot{z} - \dot{h}_{\text{des}}) \\ +\mathcal{L}_f^2 h(x) + \mathcal{L}_f \mathcal{L}_{g_{d1}} h(x) r_d \end{bmatrix} \tag{16}$$

where K_p and K_d are the PD gains.

At this point, specifying the desired performance of the truck is simplified to designing h_{des}, the desired trajectory of the output z, which is discussed in Sec. 6.

Remark 4. If smooth steering angles are desired, the control design model can be augmented with an integrator appended to u. In this case, the system has relative degree three and the control design is nearly the same.

3 Synthesis of Control Barrier Function

In this section, we review some existing results for CBFs and present our CBF synthesis process for truck lane keeping based on bilinear alternation and perturbation of the CBF level set.

3.1 Overview of Control Barrier Function. Control barrier functions were first proposed in Ref. [1] in a reciprocal form and a zeroing CBF was subsequently introduced in Ref. [24], which is more robust than the reciprocal form. A zeroing CBF is a scalar function $b(x)$ of the state x that is positive in the safe set and negative in the danger set. The algebraic set $\{x|b(x) = 0\}$ is called the boundary of the CBF. For a zeroing CBF, the barrier condition can be written as

$$\dot{b} + \kappa\alpha(b) \geq 0 \tag{17}$$

where $\kappa > 0$ is a positive constant, and α is an extended class $\mathcal{K}$ function, that is, a function $f : \mathbb{R} \to \mathbb{R}$ satisfying

- f is strictly increasing and
- $f(0) = 0$.

When $b(x) > 0$, $\dot{b}$ can be negative, but is lower bounded by $-\kappa\alpha(b)$; at the boundary, $\dot{b}$ should be non-negative, which makes the set $\{x \,|\, b(x) \geq 0\}$ controlled invariant. When $b(x) < 0$, the condition in Eq. (17) enforces convergence to the set $\{x \,|\, b(x) \geq 0\}$ by setting a lower bound $\dot{b} \geq -\kappa\alpha(b) > 0$.

3.2 Synthesis of Control Barrier Function Using Sum of Squares Programming. The synthesis of a CBF is nontrivial. We use the SOS technique to synthesize a CBF for the truck LK problem.

Sum of squares has been widely used in the computation of invariant sets and barrier certificates for continuous dynamic systems, and it can be efficiently solved with semidefinite programming. In addition, with the help of Putinar's PositivStallensatz, SOS condition is enforced on semi-algebraic sets via multipliers [25]. For more information, see Refs. [2] and [25–30].

We focus on a dynamic system with the control affine structure in Eq. (1), where the dynamics is assumed to be polynomial, $\mathcal{U}$ and $\mathcal{D}$ are known as semi-algebraic sets defined by polynomials h_u and h_d

$$\mathcal{U} = \{u \,|\, h_u(u) \geq 0\},\ \mathcal{D} = \{d \,|\, h_d(d) \geq 0\} \tag{18}$$

To make the notation compact, let $g_d(x) = [g_{d1}(x), g_{d2}(x)]$, $d = [d_1, d_2]^{\mathrm{T}}$. In CBF synthesis, we set $\alpha(b) = b$ and seek a polynomial CBF $b(x)$ that satisfies the following:

$$\{x \,|\, b(x) \geq 0\} \cap X_d = \varnothing \tag{19}$$

$$\begin{aligned} &\forall x \in \{x \,|\, b(x) \geq 0\}, \quad \forall d \in \mathcal{D}, \quad \exists u \in \mathcal{U},\ s.t. \\ &\frac{db}{dx}(f(x) + g(x)u + g_d(x)d) + \kappa b \geq 0 \end{aligned} \tag{20}$$

where X_d is the danger set, a semi-algebraic set of x determined by polynomials h_{xd}

$$X_d = \{x \,|\, h_{xd}(x) \geq 0\} \tag{21}$$

$\kappa > 0$ is a positive constant, and condition (20) is referred to as the *CBF condition*.

The difference between a barrier certificate and a CBF shows up in condition (20), which depends on the control input, u. The existential quantifier of u renders Eq. (20) not directly solvable by current SOS solvers, and thus, we seek a conservative approximation in which we assume that the control input u comes from a polynomial controller of x and d, namely

$$\begin{aligned} &\{x \,|\, b(x) \geq 0\} \cap X_d = \varnothing; \\ &\forall x \in \{x \,|\, b(x) \geq 0\}, \quad u(x,d) \in \mathcal{U}; \\ &\forall x \in \{x \,|\, b(x) \geq 0\}, \quad \forall d \in \mathcal{D}, \\ &\frac{db}{dx}\big(f(x) + g(x)u(x,d) + g_d(x)d\big) + \kappa b(x) \geq 0 \end{aligned} \tag{22}$$

d includes both measured and unmeasured disturbances, but the input may only depend on measured disturbance.

Even with the simplification, there are two bilinear terms that must be addressed to make the problem solvable by SOS. The first bilinear term is between $b(x)$ and $u(x,d)$. We use bilinear alternation [2,31], which iterates the following two steps [2]:

- Fix the barrier candidate, search for a controller.
- Fix the controller, search for a better barrier candidate.

The following SOS program solves for a controller with a fixed $b(x)$:

$$\begin{aligned} &\min e \text{ subject to} \\ &h_u(u(x,d)) - s_1(x,d)b(x) - \sum_i s_2^i(x,d)h_d^i(d) \in \Sigma[x,d] \\ &\frac{db}{dx}\big(f(x) + g(x)u(x,d) + g_d(x)d\big) + \kappa b(x) + s_3(x,d)b(x) \\ &-\sum_i s_4^i(x,d)h_d^i(d) + eQ(x,d) \in \Sigma[x,d] \\ &s_1, s_2, s_3, s_4 \in \Sigma[x,d] \end{aligned} \tag{23}$$

where s_1, s_2, s_3, and s_4 are the SOS multipliers up to a certain order, 4 in this work. Q is a fixed SOS polynomial of x and d; e is a relaxation scalar variable that makes the SOS program feasible. When $e \leq 0$, Eq. (23) is a sufficient condition of Eq. (22). s_3 is used to enforce the CBF condition only when $b(x) \geq 0$. The first SOS constraint restricts the input within $\mathcal{U}$; the second SOS constraint enforces the CBF condition.

Remark 5. The choice of Q depends on the order of the polynomial required to be SOS. In many cases, Q can simply be $x^{\mathrm{T}}x$ *or* $[x;d]^{\mathrm{T}}[x;d]$.

The other step of the bilinear alternation searches for a better CBF candidate with the controller held fixed. In this step, the second bilinear term emerges. Because the CBF condition is enforced only when $b(x) \geq 0$, an SOS multiplier is used to enforce this condition, which creates a bilinear term between b and the multiplier. We use perturbation to solve this bilinear term. The idea is to enforce the CBF condition inside the zero-level set of the current CBF candidate b_0, and search for a small perturbation Δb, as shown in Eq. (24). A similar idea can be seen in Ref. [32]

$$\min e \text{ subject to}$$

$$-b_0(x) - \Delta b(x) - \sum_i s_1^i(x) h_{xd}^i(x) \in \Sigma[x]$$

$$\frac{d(b+\Delta b)}{dx}\left(f(x) + g(x)u(x,d) + g_d(x)d\right) + \kappa(b_0(x) + \Delta b(x)) - \sum_i s_2^i(x,d) h_d^i(x) + s_3(x,d) b_0(x) + eQ(x,d) \in \Sigma[x,d]$$

$$s_1 \in \Sigma[x],\ s_2, s_3 \in \Sigma[x,d],\ \|\Delta b\| \le \epsilon \|b_0\| \tag{24}$$

The norm is taken on the coefficient of Δb and b_0 for some selected monomial bases. Note that the CBF condition is enforced on the zero level set of b_0 rather than b which makes the bilinear term disappear (since b_0 is fixed and not part of the SOS variables). Because of this, we need the zero level set of $b_0 + \Delta b$ to be similar to that of b_0, which is enforced by the last constraint, with $1 \gg \epsilon > 0$, a constant that keeps Δb small compared to b_0.

The algorithm iteratively updates b_0 by $b_0 + \Delta b$ until no further progress can be made. Upon convergence, that is, $\Delta b \to 0$, the original CBF condition is enforced.

In summary, there are two loops in the algorithm. The inner loop iterates the perturbation process, updating b_0 with $b_0 + \Delta b$, while the outer loop iterates between updating b and updating $u(\cdot)$. Denote the optimization in Eq. (23) as $[u(\cdot), e] = \mathbf{OPT_u}(b)$, with b as input, $u(\cdot)$ and e as output; and denote the optimization in Eq. (24) as $[\Delta b, e] = \mathbf{OPT_b}(b, u(\cdot))$, with $b(x)$ and $u(\cdot)$ as input, Δb and e as output. The iteration terminates when a valid CBF is found or no improvement can be made, as shown in Fig. 6. Some

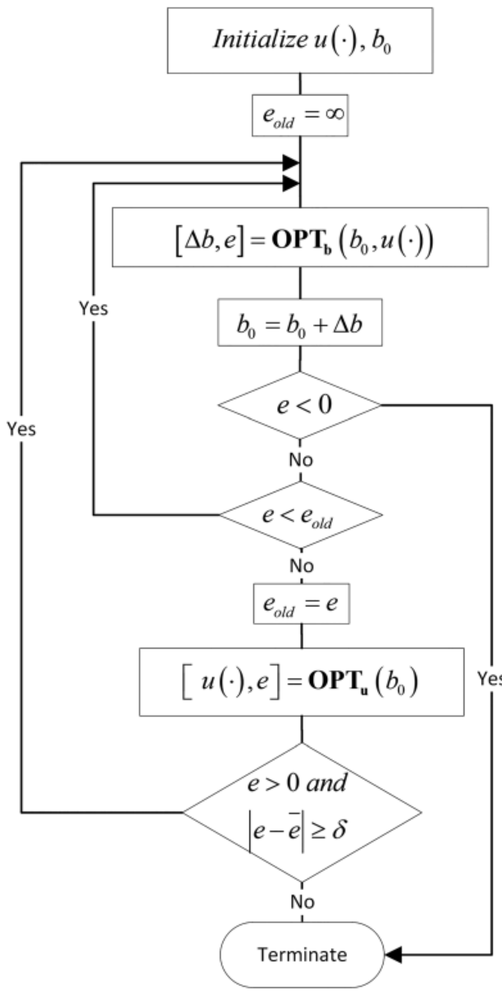
Initialize $u(\cdot), b_0$
$e_{old} = \infty$
$[\Delta b, e] = \mathbf{OPT_b}(b_0, u(\cdot))$
$b_0 = b_0 + \Delta b$
$e < 0$
No
$e < e_{old}$
No
$e_{old} = e$
$[u(\cdot), e] = \mathbf{OPT_u}(b_0)$
$e > 0$ and $|e - \bar{e}| \ge \delta$
No
Terminate
Yes
Yes
Yes

Fig. 6 Synthesis of a CBF via SOS

Table 1 List of parameters

v_x	$20\,\mathrm{m/s}$
Bound on y	$\pm 0.3\,\mathrm{m}$
Bound on ϕ	$\pm 0.1\,\mathrm{rad}$
Bound on r_d	$\pm 0.02\,\mathrm{rad/s}$(turning radius of 1000 m)
Bound on F_y	± 2000 N
Bound on δ_f	$\pm 0.2\,\mathrm{rad}$

key parameters for the CBF of lane keeping are listed in Table 1. The obtained CBF for the articulated truck model is a quadratic function of the eight states in Eq. (11).

4 Trajectory Optimization

Although a CBF guarantees safety of the system's trajectories, the closed-loop performance could be compromised if the student controller is not properly designed. For example, in Fig. 15, we show a student controller designed with linear quadratic regulator (LQR) requiring frequent interventions from the CBF and thus leading to bad ride comfort. In this section, we present an optimization procedure that incorporates the CBF condition, which is then used to train a student controller that is compatible with the CBF. In addition to the CBF condition, other constraints are needed to ensure the stability of the continuous hold controller, as introduced later in Sec. 5.1.

4.1 Direct Collocation. As discussed in Sec. 2, the trajectory optimization is boiled down to the optimization of h_{des}, the desired trajectory of the output z. Direct collocation is used to generate the trajectory of the states and h_{des}, while h_{des} is imposed as the virtual constraint.

Direct collocation is widely employed in trajectory optimization problems due to its effectiveness and robustness and is capable of enforcing nonlinear and nonconvex constraints. It is thus chosen to optimize the trajectory while enforcing the virtual constraint. It works by replacing the explicit forward integration of the dynamical systems with a series of defect constraints via implicit Runge–Kutta methods, which provides better convergence and stability properties particularly for highly underactuated dynamical systems. The result is a nonlinear programming problem [33].

In this paper, we utilize a modified Hermite–Simpson scheme based direct collocation trajectory optimization method [34]. Particularly, the flow (a.k.a. trajectory), $x(t)$, of the continuous dynamical system in Eq. (12) is approximated by discrete value x^i at uniformly distributed discrete time instant $0 = t_0 < t_1 < t_2 < \cdots < t_N = T$ with $N > 0$ being the number of discrete intervals. Let x^i and $\dot{x}^i$ be the approximated states and first-order derivatives at node i, respectively, they must satisfy the system dynamic equation given in Eq. (12). Further, if these discrete states satisfy the following defect constraints at all interior points $i \in [1, 3, \ldots, N-1]$:

$$\begin{aligned} \zeta^i &:= \dot{x}^i - \frac{3N}{2T}(x^{i+1} - x^{i-1}) + \frac{1}{4}(\dot{x}^{i-1} + \dot{x}^{i+1}) = 0, \\ \delta^i &:= x^i - \frac{1}{2}(x^{i+1} + x^{i-1}) - \frac{T}{8N}(\dot{x}^{i-1} - \dot{x}^{i+1}) = 0 \end{aligned} \tag{25}$$

then they are accurate approximations of the given continuous dynamics. Equation (25) defines the modified Hermite–Simpson conditions for the direct collocation trajectory optimization [34].

Based on the above formulation, now we can construct a constrained nonlinear programming problem to solve the trajectory optimization with the virtual constraint for the articulated truck model. To incorporate the virtual constraints based feedback control with the trajectory optimization, we enforce the output dynamics equation given in Eq. (15) at each node. Then, the control input u^i will be implicitly determined via this constraint without explicitly enforcing it as in Eq. (16). Further, the output z and its derivative $\dot{z}$ should equal to the desired trajectory $h_{\rm des}(t)$ at $t = 0$ to ensure that the system lies on the zero dynamics manifold $\forall t \in [0, T]$. The desired trajectory $h_{\rm des}$ is parameterized as a Bezier curve, which is widely used in computer graphics and related fields. A Bezier curve of order m is an mth-order polynomial defined on $[0, 1]$

$$\mathbf{B}(s) = \sum_{i=0}^{m} \alpha_i \binom{m}{i} s^i (1-s)^{m-i} \tag{26}$$

where α_i are the Bezier coefficients.[2] The Bezier order is chosen to be 8.

The initial condition of the trajectory optimization includes the initial state of the truck and the road curvature.

Let $\mathcal{J}(\cdot)$ be the cost function to be minimized, the trajectory optimization problem can be stated as

$$\begin{aligned}
&\text{argmin } \mathcal{J}(\cdot) \text{ subject to}\\
&\zeta^i = 0, \quad \delta^i = 0,\\
&\dot{x}^i = f(x^i)x^i + g(x^i)u^i + g_{d1}(x^i)d_1,\\
&\ddot{z}i - \ddot{h}_{\rm des}(t_i) + K_p\big(z^i - h_{\rm des}(t_i)\big) + K_d\big(\dot{z} - \dot{h}_{\rm des}(t_i)\big) = 0,\\
&x(t_0) = x^0,\\
&z^0 - h_{\rm des}(t_0) = 0,\\
&\dot{z}^0 - \dot{h}_{\rm des}(t_0) = 0,\\
&-u_{\max} \le u^i \le u_{\max},\\
&\dot{b}(x^i, \dot{x}^i) + \kappa \frac{e^{b(x^i)} - 1}{e^{b(x^i)} + 1} \ge 0,\\
&V\big(x(T) - \eta_x(d_1)\big) \le c_1 V\big(x^0 - \eta_x(d_1)\big),\\
&\|x_2(T) - \gamma(x)(T)\| \le c_3
\end{aligned} \tag{27}$$

where $z^i = z(x^i)$, $\dot{z}^i = \dot{z}(x^i)$, and $\ddot{z}^i = \ddot{z}(x^i, \dot{x}^i)$, respectively. The first three lines of constraints correspond to the collocation constraint; K_p and K_d are a set of stabilizing gains as discussed in Eq. (8), which are also used to tune the performance of the trajectory tracking. The fourth line specifies the initial states; the fifth and sixth lines correspond to the virtual constraint; the seventh line is the input constraint; the eighth line is the CBF constraint, the last two constraints are needed to guarantee stability of the continuous hold controller, which is explained in Appendix B.

Remark 6. The CBF condition is modified based on Eq. (17). Since $(e^b - 1/e^b + 1)$ is bounded within $[-1, 1]$, when $b(x)$ is small, the lower bound for $\dot{b}$ saturates at 1, instead of growing linearly as $-\gamma b$, which may be too difficult to satisfy. Besides, when $b(x) = 0$, $(e^b - 1/e^b + 1) = 0$, which resembles the original CBF condition in Eq. (19). Since $(e^b - 1/e^b + 1)$ is still an extended class $\mathcal{K}$ function, by Proposition 1 in Ref. [35], $\{x \,|\, b(x) \ge 0\}$ is still invariant under the modified constraint.

The cost function in Eq. (27) is a weighted sum of multiple cost functions, consisting of the following terms:

- Final value cost $V(x^{\rm T} - \eta_x(r_d))$, where $\eta_x(r_d)$ is the steady-state under a given r_d, and $V(\cdot)$ is a Lyapunov function around the origin;
- $\int z^2 dt$, the square integral of z;
- $\int \dddot{z}^2 dt$, the square integral of jerk;
- $\|y\|_\infty$, the maximum deviation from road center;
- $\|r\|_\infty$, the maximum yaw rate;
- $\int u^2 dt$, the square integral of the input; and
- $|\alpha_m|$, the penalty on the last Bezier coefficient (facilitate convergence of the Bezier curve).

[2]Bezier curve can parameterize trajectories of any finite length by scaling the input. Suppose the horizon of $h_{\rm des}$ is T, then the input is defined as $s = t/T$.

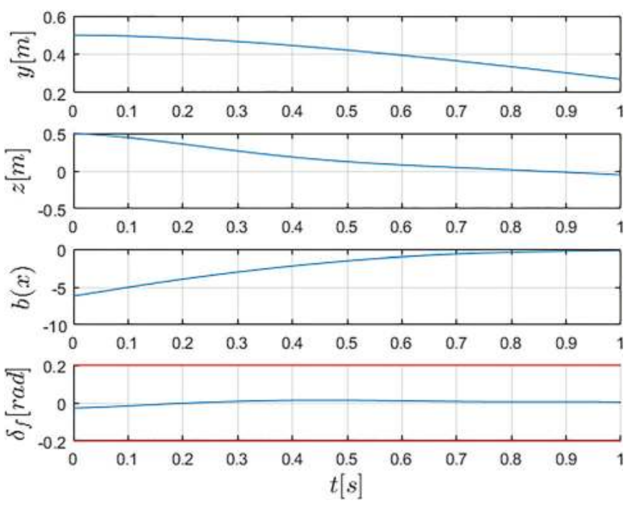

Fig. 7 Example of trajectory optimization result

The terms that consist of function integrals are approximately computed using Simpson's quadrature rule [36].

The setting of the constraints and costs is the result of repeated trial and tuning. It should be emphasized that CBF constraint is enforced in the trajectory optimization. We hope that by enforcing CBF condition on the training set, the policy generated by supervised learning inherits this property.

Figure 7 shows an example trajectory with initial lateral deviation $y_0 = 0.5$ m and road yaw rate $r_d = 0.02$ rad/s. The plot of y and the Bezier output z shows that the trajectory is converging to the lane center. The plot of the CBF value and the control input shows that the trajectory generated by direct collocation satisfies the input and CBF constraints.

The trajectory optimization is solved with FROST, which uses a symbolic calculation to boost the nonlinear optimization [37]. The trajectory optimization for each initial condition can be finished within 10 s.

4.2 Generating the Training Set.

It is impossible to perform trajectory optimization for all the initial conditions offline, so instead, we use supervised learning to train the mapping from initial conditions to desired trajectories with a finite trajectory library, which is generated by the above-described trajectory optimization process.

By varying the initial conditions and generating the corresponding trajectories with direct collocation, we hope to "train" the neural network to generate good trajectories for various initial conditions. The inputs to the neural network are called features, denoted as Φ; in our case, they are variables that describe the initial condition. The output of the neural network is a vector of control parameters, denoted as $\mathcal{A}$, in this case, the Bezier coefficients

$$\pi : \Phi \to \mathcal{A} \tag{28}$$

The selection of initial conditions is done in a grid fashion. We define a grid on the feature space and perform trajectory optimization on each grid point. Since the zero dynamics is stable, $z(t) \to h_{\rm des}(t)$ for $h_{\rm des} \in C^2$ implies $x(t) \to x_{\rm des}(t)$, where $x_{\rm des}$ is the desired state trajectory corresponding to $h_{\rm des}$. This implies that we only need two states to determine the asymptotic behavior of the system, but not necessarily the transient behavior. In practice, the more states we use to parameterize the initial condition, the finer the trajectory library will be.

However, under a grid fashion of drawing samples, the number of samples needed grows exponentially with the state dimension. Therefore, the dimension of Φ is limited by available computation power. We let Φ contain six features, including five states and r_d

$$\Phi = [y, \psi, r, \psi_a, v_y, r_d] \quad (29)$$

Under this setup, the computation needed to generate the trajectory library is manageable (about 20 h on a desktop). With more computation power, a higher dimensional Φ can lead to a finer trajectory library.

Even though most driving behavior is mild, it is important that the controller be able to handle bad initial conditions. We generate, therefore, two training sets, denoted as S_1 and S_2, where S_1 consists of trajectories defined for a duration of 1 s, and the features of the trajectories have a wider span and S_2 consists of trajectories defined over a three-second window, with the features more concentrated around the origin. S_1 is used to train a mapping for severe initial conditions and transients, and S_2 is used to train a mapping for mild situations and normal driving. Some of the initial conditions might render the trajectory optimization infeasible, therefore only the feasible cases are included in the training sets. In the implementation, the CH controller will choose which mapping to use based on the severity of the situation.

The parameters for the training are included in Table 2. In total, there are 62,825 trajectories in S_1, and 29,300 trajectories in S_2.

4.3 Supervised Learning.

With the training set ready, there are several choices for the learning method, such as linear regression, Gaussian process, and neural networks. In our problem, since there is no structural information about the trajectory generator and we need strong expressive power to capture the potentially complicated mapping from the initial condition to the desired trajectory, we choose a neural network for its strong expressive power.

We train a neural network that has six hidden layers with 200 neurons in each layer and use the ReLU function as the rectifier. The training is performed using TENSORFLOW [38]. Over 85% of the data are used for training and 15% are used for testing. Table 3 shows the mean-squared-error (MSE) of the training result.

5 Implementation of Learning Based Controller

5.1 Continuous Hold Feedback Control.

Once the trajectory generator is trained, we can generate a finite horizon desired trajectory for a given initial condition. In order to piece together the finite horizon trajectories and synthesize a controller from the trajectory generator, we employ a CH controller. The name continuous hold comes from the analogy with a zero-order hold and an nth-order hold. While an nth-order hold approximates the segment between two consecutive sampling times with an nth-order polynomial, continuous hold executes a predefined continuous trajectory.

The idea of continuous hold is not claimed to be novel; a motion primitive is a special type of continuous hold [39]. The trajectory is updated in an event-triggered fashion, which will be discussed in detail in Sec. 5.2. While event-triggered finite-horizon control is studied in Ref. [40], in the CH setting, it should be noted that the control action between triggering events is a continuous function of time and states instead of being a constant.

For the truck example, the basic continuous hold controller [41] must be extended to systems with exogenous disturbances. The stability and the set invariance property of the CH controller are proved, including the analysis for the case when only a subset of the state is used for feedback, in Appendix B.

5.2 Event-Triggered Update of the Continuous Hold Controller.

The CH controller uses the mapping trained by supervised learning to generate a desired trajectory h_{des} for the output z based on the current state and r_d, then track the desired trajectory with the control law in Eq. (16). The desired trajectory will be updated under three circumstances:

- The desired trajectory is executed to the end.
- There is a significant change in road curvature.
- The trajectory tracking error becomes large.

In the first case, since the trajectory optimization has a finite horizon (1 s or 3 s), the neural network will use the current value of the features to generate a new desired trajectory. In the second case, if the road curvature r_d differs much from that used to generate the current desired trajectory, the trajectory should be updated since r_d is assumed to be constant during the entire horizon of the trajectory. The rest of the features are simply initial conditions, so their change does not trigger an update of the desired trajectory.

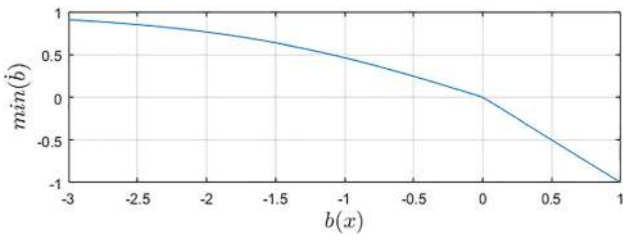

Fig. 8 Lower bound for $\dot{b}$

Fig. 9 Animation with a 312 state model in TRUCKSIM

Table 2 Training set parameter setting

Feature	S_1	S_2
y range	$[-0.5, 0.5]$[m]	$[-0.3, 0.3]$[m]
v_y range	$[-1, 1]$[m/s]	$[-1, 1]$[m/s]
ψ range	$[-0.04, 0.04]$[rad]	$[-0.04, 0.04]$[rad]
r range	$[-0.06, 0.06]$[rad/s]	$[-0.03, 0.03]$[rad/s]
r_d range	$[-0.03, 0.03]$[rad/s]	$[-0.025, 0.025]$[rad/s]
ψ_a range	$[-0.04, 0.04]$[rad]	$[-0.04, 0.04]$[rad]

Table 3 Training result

	S_1	S_2
MSE of training data	0.13	0.0023
MSE of testing data	0.16	0.0024

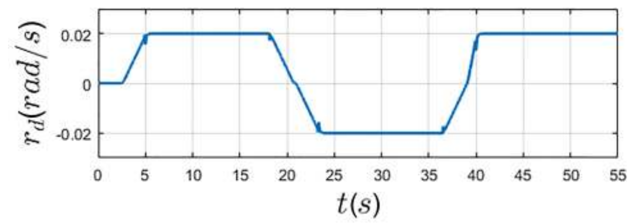
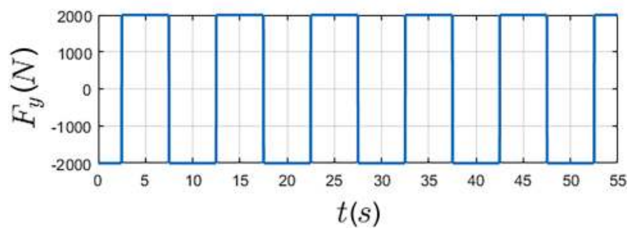
Fig. 10 Disturbance to the system

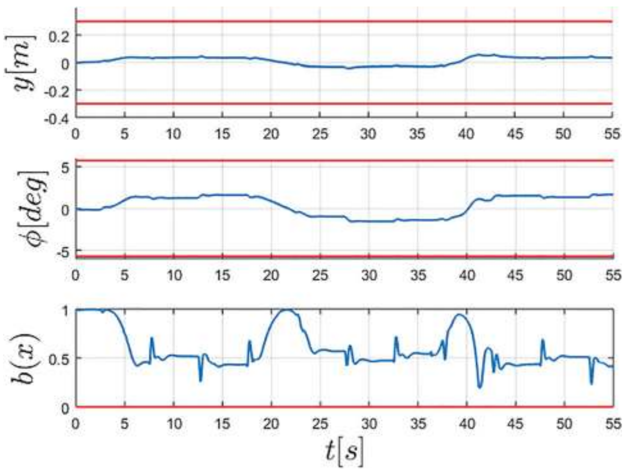
Fig. 11 Value of the CBF and key states during simulation

In the third case, when the trajectory deviates too far from the desired trajectory, replanning is called for. This is likely to be caused by an unexpected disturbance, such as wind gust.

When switching from one trajectory to the next, smoothing is performed to make sure that h_{des} is twice differentiable, which ensures that the control signal is continuous. The smoothing process is explained in Appendix C.

5.3 Control Barrier Function as a Supervisory Controller. Even though the CBF condition is enforced in the trajectory optimization used in the training set, after supervised learning, there is no guarantee that the trajectory generated by the neural network always satisfies the CBF condition. Therefore, CBF is still implemented as a supervisory controller on top of the CH controller, as shown in Fig. 3. The CBF solves the following optimization:

$$\begin{aligned} &\min_{\Delta u} w_1\|\Delta u\|^2 + w_2\|\Delta u - \Delta u_{\mathrm{old}}\|^2 \\ &s.t.\, \mathcal{C}(x, d, u_0 + \Delta u) \geq 0, \quad u_0 + \Delta u \in \mathcal{U} \end{aligned} \tag{30}$$

where Δu is the intervention of the CBF, Δu_{old} is the intervention of the previous time instant, and $\mathcal{C}(\cdot)$ is the CBF condition. The reason for the second penalty term is to prevent chattering if intervention is necessary. The CBF condition is defined as

$$\begin{cases} \dot{b} + \gamma b \geq 0, & \text{if } b(x) \geq 0 \\ \dot{b} + \gamma \dfrac{e^b - 1}{e^b + 1} \geq 0, & \text{if } b(x) \leq 0 \end{cases} \tag{31}$$

where the transition at $b(x) = 0$ is continuous, i.e., the two constraints coincide at $b(x) = 0$.

Remark 7. When $b(x) \geq 0$, the existence of Δu is guaranteed by the construction of the CBF; when $b(x) \leq 0$, there is no guarantee of feasibility. When Eq. (30) is infeasible, the input is saturated by $\mathcal{U}$.

6 Simulation Result

We validate our control design on TRUCKSIM, a high-fidelity physics-based simulation software that is widely acknowledged

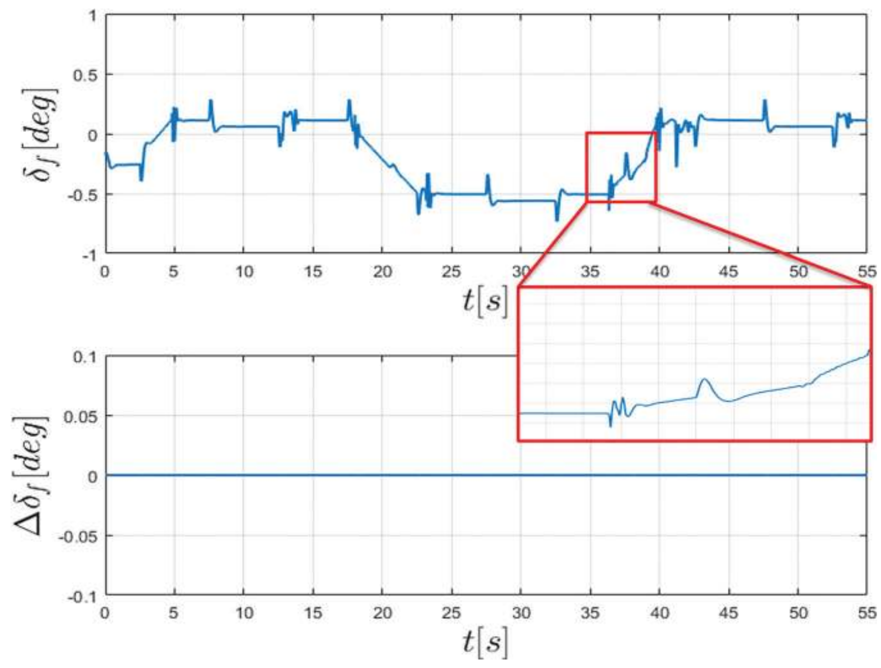
Fig. 12 Input and intervention of CBF during simulation

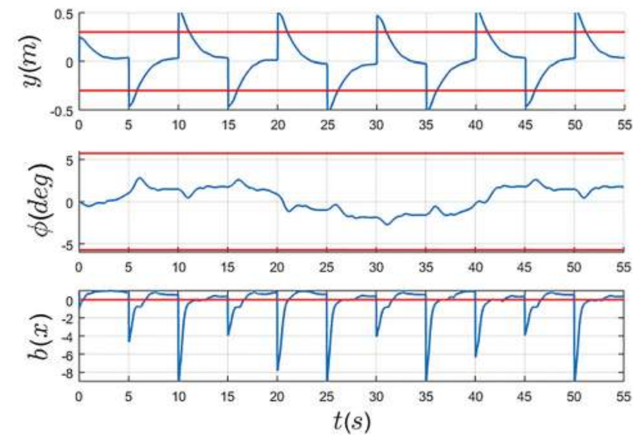

Fig. 13 Value of CBF and key states with large initial deviations

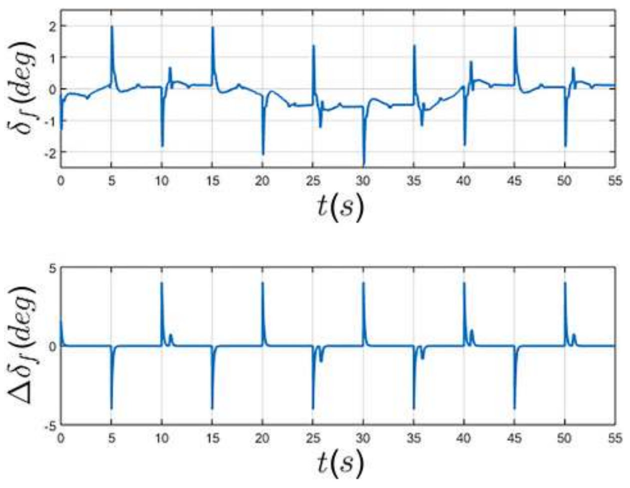

Fig. 15 Simulation result with LQR as student controller

by the trucking industry. The model picked for simulation has 312 states and is a tractor-semitrailer with heavy cargo in the trailer, weighing 35 tons in total; see Fig. 9.

The truck is asked to drive on a road with a minimum turning radius of 1000 m at 20 m/s. A side-wind is simulated as a lateral force and roll moment to the truck. Because of the heavy cargo, the truck has a high center of gravity. Hence, the roll motion in the simulation is significant and the commanded maneuvers are aggressive.

As shown in Fig. 10, the road profile consists of segments with constant curvature (per U.S. road design standards). Though rather extreme, the side-wind is a square wave with maximum allowed magnitude.

Figure 11 shows the value of the CBF and two key states. Lateral deviation y and roll angle ϕ noever exceed the desired limits (plotted in red) and $b(x)$ was always above zero, showing that the CBF (safety) bound was never breached.

The steering input trajectory is shown in Fig. 12. We zoom in the input to show a 5 s period of input. The input is reasonably smooth. The bumps are necessary to counter the side-wind when it changes direction. The lower plot shows $\Delta\delta_f$, the intervention from the CBF, and its constant value of zero indicates that no interventions from CBF occurred.

To demonstrate the controller's ability to handle bad initial conditions, we perturb the lateral deviation with a square wave, simulating the situation when the initial position is 0.5 m from the lane center, as shown in Figs. 13 and 14.

Figure 14 shows the input under a large deviation. The CBF intervened three times, and the interventions are mild compared to the size of u_0. When $b(x)$ was below zero, the learned controller was able to drive the system back to the safe set without the intervention of the CBF.

As a comparison, we tuned an LQR controller with feedforward control of r_d, and it performed very well under normal driving conditions. However, when the initial condition is bad (under the same setting as Fig. 13), the LQR controller triggered intervention from the CBF multiple times (11 times) and the jerk was severe, as shown in Fig. 15.

Though the LQR controller was fine-tuned, it triggered severe intervention from the CBF frequently. On the other hand, we observed none or very mild interventions from the CBF under the

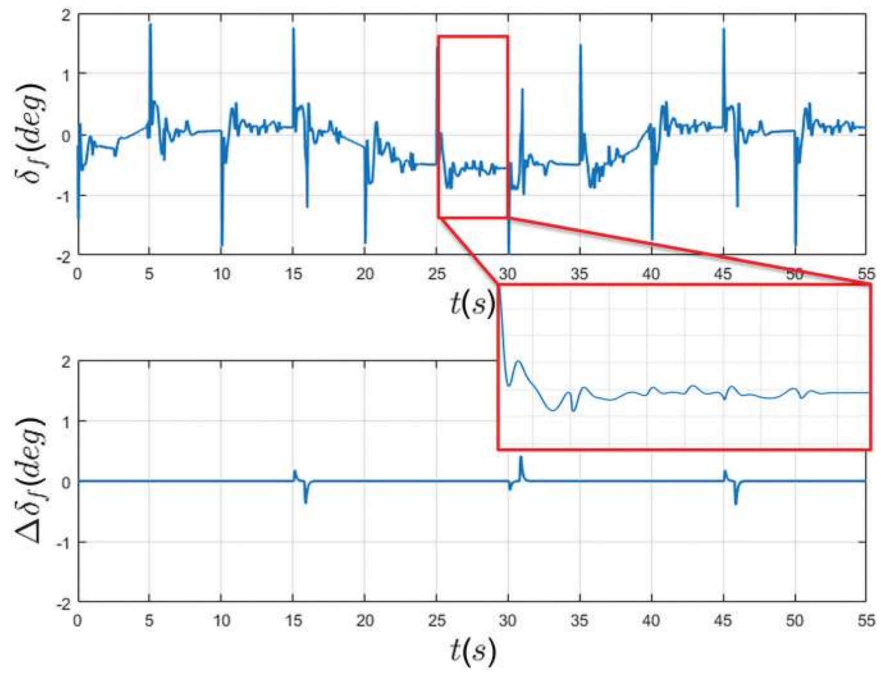

Fig. 14 Input and intervention of CBF with large initial deviations

learning-based controller in all trial simulations when the states are within the span of the training set.

7 Conclusion and Discussion

We propose a supervised learning approach to construct controllers with smooth performance and a provable safety guarantee. The idea is to use trajectory optimization to generate a training set consisting of trajectories that satisfy a CBF safety constraint, then use supervised learning to extract a mapping from system initial conditions to desired trajectories. The policy generated with supervised learning inherits the good properties of the training set, though nothing can be proved. On top of that, a safety guarantee is formally imposed with a CBF as a supervisory controller. The simulations showed that the proposed approach is able to reduce the intervention of the CBF and therefore provide high-quality closed-loop performance while guaranteeing safety.

We chose to learn a mapping from initial conditions to the desired output trajectory, instead of a mapping from the initial condition to the desired input trajectory. Trajectory tracking was implemented with a CH controller. The CH control structure is able to transform the synthesis problem into a trajectory optimization problem, which may be much simpler for complicated nonlinear systems such as trucks and robots [41].

There are problems to be solved for the proposed method. First, when the initial condition is not contained inside the feature range of the training set, i.e., when the neural network is doing extrapolation rather than interpolation, the performance can be poor. The proposed method can and should switch to other controllers that can handle more severe initial conditions when the initial condition is not covered in the training set. Though rather obvious, it is important to emphasize that to obtain good performance over a wide range, one needs to have training data with adequate coverage. Second, when training data from a large range of features are stacked together, the regression accuracy may drop and the performance suffers. To solve this, one might need a more complicated neural network structure or use multiple neural networks for different situations.

Acknowledgment

The work of Yuxiao Chen, A. Hereid, and Huei Peng is supported by NSF Grant CNS-1239037. The work of J. Grizzle is supported by Toyota Research Institute (TRI).

Funding Data

- National Science Foundation (Grant No. CNS-1239037; Funder ID: 10.13039/501100008982).
- Toyota Research Institute (TRI) (Funder ID: 10.13039/501100004405).

Nomenclature

C^n =	sets of functions with continuous nth derivatives
$\mathbb{R}$ =	the set of real number
$\mathbb{R}^n$ =	the n-dimensional Euclidean space
$\mathbb{R}[x]$ =	the space of all polynomials of x
$\Sigma[x]$ =	the cone of SOS polynomials, a subset of $\mathbb{R}[x]$

For a scalar function $h:\mathbb{R}^n\to\mathbb{R}$ of $x\in\mathbb{R}^n$ and a vector field $f:\mathbb{R}^n\to\mathbb{R}^n$, the Lie derivative is defined as $\mathcal{L}_f h(x)=(dh/dx)f(x)$, which is a scalar function of x, and $\mathcal{L}_f^n h=\mathcal{L}_f\mathcal{L}_f^{n-1}h$, with $\mathcal{L}_f^0 h=h$

Appendix A: Zero Dynamics of the Truck Lateral Dynamics

To show the zero dynamics, we use the following state transformation that renders a new choice of states:

$$\chi = T = \begin{bmatrix} z & \dot{z} & \sigma^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{A1}$$

where T is a full-rank linear transformation matrix

$$T=\begin{bmatrix} 1 & 0 & v_xT_0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & v_x & v_xT_0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & -B^4 & 0 & B^2 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & -B^6 & 0 & 0 & 0 & B^2 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & -B^8 & 0 & 0 & 0 & 0 & 0 & B^2 \end{bmatrix} \tag{A2}$$

and B^i is the ith entry of B in Eq. (12). The transformation is linear and full rank. The dynamics under χ is

$$\dot{\chi}=\bar{A}\chi+\bar{B}u+\bar{E}r_d \tag{A3}$$

Moreover

$$\bar{B}=TB=\begin{bmatrix} 0 & CAB & \mathbf{0}_{1\times 6}\end{bmatrix}^{\mathrm{T}} \tag{A4}$$

where $C=\begin{bmatrix}1 & 0 & T_0v_x & \mathbf{0}_{1\times 5}\end{bmatrix}$. Therefore, the dynamics under χ can be written as

$$\dot{\chi}=\begin{bmatrix}\dot{\sigma}\\ \dot{z}\\ \ddot{z}\end{bmatrix}=\begin{bmatrix}\Gamma(\sigma,z,\dot{z})\\ \dot{z}\\ CA^2x\end{bmatrix}+\begin{bmatrix}\mathbf{0}_{6\times 1}\\ 0\\ CAB\end{bmatrix}u+\begin{bmatrix}\mathbf{0}_{6\times 1}\\ 0\\ CAE_1\end{bmatrix}r_d \tag{A5}$$

where $\Gamma(\sigma,z,\dot{z})$ has exponentially stable zero dynamics, meaning $\dot{\sigma}=\Gamma(\sigma,0,0)$ is exponentially stable.

Appendix B: Analysis of the Continuous Hold Controller

In this section, we present the stability and set invariance analysis of the CH controller.

B.1 Continuous Hold Controller With Full State Parameterization. Systems of the following form are considered:

$$\dot{x}=f(x,u,d) \tag{B1}$$

where x, u, and d are the state, the input, and the measured disturbance, respectively.

Remark 8. The result about the CH controller is applicable to general nonlinear dynamic model; the control-affine nonlinear model in Eq. (1) (assuming $d_2=0$) and the linear model of the truck are special cases.

We make the following assumptions about the system dynamics:

Assumption 5. *$f:\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^p\to\mathbb{R}^n$ is locally Lipschitz continuous in x, u, and d.*

Assumption 6. *$\exists\mathcal{D}\subseteq\mathbb{R}^p$ and mappings $\eta_x:\mathcal{D}\to\mathbb{R}^n$, $\eta_u:\mathcal{D}\to\mathbb{R}^p$, Lipschitz continuous, such that $f(\eta_x(d),\eta_u(d),d)=0$, i.e., for every exogenous disturbance $d\in\mathcal{D}$, there exist two mappings η_x and η_u that map any $d\in\mathcal{D}$ to a unique equilibrium point and a unique input that maintains the equilibrium.*

Remark 9. There may be nonunique equilibrium points of the system due to the cyclic coordinates, i.e., states that do not affect the dynamics, see Ref. [42] for details. Therefore, a function η_x is needed to select a single equilibrium point given d. Assumption 2 gives a possible definition of η_x.

Assumption 7. *$\forall d \in \mathcal{D}$, there is an open ball $B_d \subset \mathbb{R}^n$ about the origin, and a positive-definite, locally Lipschitz-continuous function $V_d : B_d \to \mathbb{R}$, and constants $0 \le \alpha_1 \le \alpha_2$ such that $\forall x \in B_d + \eta_x(d)$*

$$\begin{aligned}&\bar{x} = x - \eta_x(d),\\ &\alpha_1 \bar{x}^{\mathrm{T}}\bar{x} \le V_d(\bar{x}) \le \alpha_2 \bar{x}^{\mathrm{T}}\bar{x}\end{aligned} \tag{B2}$$

Assumption 8. *$\exists \mathcal{S} \subseteq \mathbb{R}^n$, compact, such that $\forall d \in \mathcal{D}, \eta_x(d) \in \mathcal{S}$. There exists CBF $b(x)$, such that $\forall x \notin \mathcal{S}, b(x) \le 0$; $\forall d \in \mathcal{D}$, $b(\eta_x(d)) > 0$. Moreover, $\forall \xi \in \mathcal{S}, \forall d \in \mathcal{D}$, there exist $u_\xi^d : [0, T_p] \to \mathbb{R}^m$ and a corresponding state trajectory $\varphi_\xi^d : [0, T_p] \to \mathbb{R}^n$ satisfying*

$$\begin{aligned}&V_d\left(\varphi_\xi^d(T_p) - \eta_x(d)\right)\\ &\quad \le c_1 V_d\left(\varphi_\xi^d(0) - \eta_x(d)\right) \frac{db}{dx}\bigg|_{\varphi_\xi^d(t)} \dot{\varphi}_\xi^d(t) + \kappa b\left(\varphi_\xi^d(t)\right) \ge 0\\ &\forall t \in [0, T_p], \quad \lim_{\xi \to \eta_x(d)} \varphi_\xi^d(t) = \eta_x(d), \quad \lim_{\xi \to \eta_x(d)} u_\xi^d(t) \equiv \eta_u(d)\end{aligned} \tag{B3}$$

where $T_p > 0$ is the horizon, $\kappa > 0$, $1 > c_1 > 0$ are predefined constants.

A CH controller maintains a timer $\hat{t}$ that is reset to 0 when the triggering event occurs and the desired trajectory is updated. In between events, the timer increases at a constant rate equal to 1. An update is triggered when either the trajectory is executed to its end, i.e., $\hat{t} = T_p$, or when an interruption is detected. A possible interruption includes a change in d or an unexpected disturbance that makes the tracking error too big. The CH input is

$$u^{\mathrm{CH}}(\hat{t}, x, d) = u_\xi^d(\hat{t}) + u^{\mathrm{fb}}(x, \varphi_\xi^d(\hat{t})) \tag{B4}$$

where ξ is the initial state when $\hat{t} = 0$, and $u^{\mathrm{fb}} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$ is a feedback controller that tracks φ_ξ^d.

The closed-loop system under CH feedback is then

$$\dot{x} = f^{\mathrm{CH}}(\hat{t}, x, d) := f(x, u_\xi^d(\hat{t}) + u^{\mathrm{fb}}(x, \varphi_\xi^d(\hat{t})), d) \tag{B5}$$

Assumption 9. *For any trajectory φ_ξ^d in Assumption 8 that a CH controller tries to follow, there exists a feedback controller $u^{fb} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$ that makes φ_ξ^d uniformly locally exponentially stable, i.e., the closed-loop system in Eq. (B5) satisfies*

$$\begin{aligned}&\exists B \subseteq \mathbb{R}^n, \quad s.t. \forall 0 \le t_1 \le t_2 \le T_p, \quad (x(t_1) - \varphi_\xi^d(t_1)) \in B,\\ &\|x(t_2) - \varphi_\xi^d(t_2)\| \le e^{-c_2(t_2 - t_1)}\|x(t_1) - \varphi_\xi^d(t_1)\|\end{aligned} \tag{B6}$$

for some $c_2 > 0$.

Next, we present the result on the stability property of the CH controller. First, consider the case when d is fixed.

Theorem 1. *Under Assumptions 5–9, for an initial condition $\xi \in \{x \mid b(x) \ge 0\}$, the closed-loop system in Eq. (B5) will stay inside $\{x \mid b(x) \ge 0\}$, and if d stops changing after $T \ge 0$, the state will converge to $\eta_x(d)$ exponentially.*

Proof. From Assumption 8, since $\xi \in \{x | b(x) \ge 0\}$, $\xi \in \mathcal{S}$, which means the feedback control in Eq. (B4) is well defined. From Eq. (B3), the CBF $b(x)$ remains non-negative as discussed in Sec. 3, which proves that the state will stay inside $\{x \mid b(x) \ge 0\}$, and thus $x(t) \in \mathcal{S}, \forall t \ge 0$.

When d stops changing, from the Lyapunov condition in Assumption 8 $V_d(x(nT_p) - \eta_x(d)) \le c_1^{n-1} V_d(x(0) - \eta_x(d))$, $n = 1, \ 2, 3, \ldots$, which implies $\lim_{n \to \infty} V_d(x(nT_p)) = 0$. From Assumption 7, the sequence $x(nT_p)$ converges to $\eta_x(d)$. Therefore, from the last assumption in Eq. (B3), the state stays at $\eta_x(d)$. ■

B.2 State Decomposition and Dimension Reduction. As discussed in Sec. 4.2, under a grid fashion sampling of the initial condition, the computation power limits the dimension of the feature that describes the initial condition. To parameterize the initial condition with a subset of states, we decompose the states into two parts: $x = [x_1; x_2]$, where in practice $x_1 \in \mathbb{R}^{n_1}$ are states with slow dynamics and $x_2 \in \mathbb{R}^{n_2}$ are states with fast and stable dynamics. We consider the case where the trajectory and the tracking feedback u^{fb} are parameterized by only x_1.

Definition 1. *A locally Lipschitz continuous function $\gamma : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ such that $\gamma(0) = 0$ and satisfies*

$$\begin{aligned}&\forall d \in \mathcal{D}, \eta_x(d) = [\eta_x^1(d); \eta_x^2(d)],\\ &\eta_x^2(d) = \gamma(\eta_x^1(d))\end{aligned} \tag{B7}$$

is called an insertion map.

The condition in Eq. (B7) states that for any $d \in \mathcal{D}$, the insertion map maps the steady-state of x_1 to the steady-state of x_2. To extend the previous conclusion to cases where trajectories are parameterized with only x_1, we make the following assumptions:

Assumption 10. *$\exists \mathcal{S} \subseteq \mathbb{R}^n$, compact, such that $\forall d \in \mathcal{D}$, $\eta_x(d) \in \mathcal{S}$. There exists CBF $b(x)$, such that. $\forall x \notin \mathcal{S}, b(x) \le 0$; $\forall d \in \mathcal{D}$, $b(\eta_x(d)) \ge 0$; $\forall d \in \mathcal{D}, \xi = [\xi_1; \xi_2] \in \mathcal{S}$, $\xi_2 = \gamma(\xi_1)$, with $1 > c \ge 0$, there exists $u_{\xi_1}^d : [0, T_p] \to \mathbb{R}^m$ and a corresponding state trajectory, $\varphi_{\xi_1}^d : [0, T_p] \to \mathbb{R}^n$ satisfying*

$$\begin{aligned}&V_d\left(\varphi_{\xi_1}^d(T_p) - \eta_x(d)\right) \le c V_d\left(\varphi_{\xi_1}^d(0) - \eta_x(d)\right),\\ &\frac{db}{dx}\bigg|_{\varphi_{\xi_1}^d(t)} \dot{\varphi}_{\xi_1}^d(t) + \kappa \varphi_{\xi_1}^d(t) \ge 0\end{aligned} \tag{B8}$$

$$\begin{aligned}&\lim_{[\xi_1, \gamma(\xi_1)] \to \eta_x(d)} \varphi_\xi^d(t) = \eta_x(d),\\ &\lim_{[\xi_1, \gamma(\xi_1)] \to \eta_x(d)} u_\xi^d(t) \equiv \eta_u(d)\end{aligned} \tag{B9}$$

$$\varphi_{2\xi_1}^d(T_p) = \gamma(\varphi_{1\xi_1}^d(T_p)) \tag{B10}$$

Assumption 11. *There exists a feedback $u_1^{fb} : \mathbb{R}^{n_1} \times \mathbb{R}^{n_1} \to \mathbb{R}^m$ that $u_1^{fb}(x_1, \varphi_{1\xi_1}^d(\hat{t}))$ makes $\varphi_{\xi_1}^d$ uniformly locally exponentially stable, i.e., Eq. (B6) is satisfied with $u(\hat{t}) = u_\xi^d(\hat{t}) + u_1^{fb}(x_1, \varphi_{1\xi_1}^d(\hat{t}))$.*

Remark 10. The subscript $\varphi_{1\xi_1}^d$ means the desired trajectory of x_1 with initial condition $x_1(0) = \xi_1$, *and* $\varphi_{2\xi_1}^d$ means the desired trajectory of x_2, $\varphi_{\xi_1}^d = [\varphi_{1\xi_1}^d; \varphi_{2\xi_1}^d]$. Assumption 11 is possible if the dynamic subsystem of x_2 is locally exponentially stable.

Theorem 2. Under Assumptions 5–7, 10, and 11, $\forall d \in \mathcal{D}$, $\forall \xi = [\xi_1; \gamma(\xi_1)] \in \{x | b(x) \ge 0\}$, the closed-loop system under CH feedback will stay inside $\{x \mid b(x) \ge 0\}$, and if d stops changing after some $T \ge 0$, the state will converge to $\eta_x(d)$ exponentially.

Proof. By Assumption 11, the closed-loop system exponentially converges to the CH desired trajectory. From Assumption 10, by CBF condition, $\{x | b(x) \ge 0\}$ is invariant under the CH controller. When d stops changing, the closed-loop system exponentially converges to φ_ξ^d and $V_d(x(nT_p) - \eta_x(d)) \le c^{n-1} \ V_d(\xi - \eta_x(d))$, for $n = 1, 2, 3, \ldots$, and satisfies $x_2(nT_p) = \gamma(x_1(nT_p))$. So every time the desired trajectory is executed to the end, there exists $\varphi_{x_1(nT_p)}^d$ that follows the previous trajectory. By definition of the insertion map, Eq. (B7) makes sure that when $x_1 \to \eta_x^1(d)$, $\gamma(x_1) \to \eta_x^2(d)$. By Eq. (B9), $x(t)$ converges to $\eta_x(d)$ exponentially. ■

Remark 11. When the dynamics of x_2 is stable and fast, $y := x_2 - \varphi_{2\xi_1}$ converges to zero quickly, the influence of initial condition of x_2 is small enough to be neglected. Therefore, the CH can be parameterized only by x_1.

Now consider a CH controller with trajectories generated with the procedure described in Eq. (27). Assumptions 5 and 6 are trivially satisfied by the linear dynamics, where η_x is defined such that it maps r_d to the equilibrium point that renders $z = h(x) = 0$, which is unique. It can be shown that η_x is Lipschitz continuous. We use the cost-to-go function V of a LQR as the Lyapunov function by solving the Riccati equation. Since V is quadratic and the truck dynamic is linear, V satisfies Assumption 7 for all r_d. The CBF condition and Lyapunov condition in Assumption 10 are enforced in the trajectory optimization by the last two constraints in Eq. (27). Pick $x_1 = [z \quad \dot{z} \quad \psi \quad -B_4 v_y + B_2 r \psi_a]$, since z and $\dot{z}$ are part of x_1, the closed-loop dynamics is indeed stable under the PD control that only depends on x_1, which is the direct result of a stable zero dynamics, therefore satisfies the exponential stability condition in Assumption 11. Note that the initial conditions in the training set are parameterized by Φ, which is a full rank linear transformation of x_1 and r_d. By Theorem 2, the closed-loop system with CH feedback stays within $\{x \,|\, b(x) \geq 0\}$, and converges to $\eta_x(d)$ exponentially once r_d stops changing.

Appendix C: Smoothing of the Desired Trajectory

The smoothing of a Bezier curve is very simple. For an mth-order Bezier curve, the values for zeroth to second derivative at $s = 0$ are

$$\begin{aligned} \mathbf{B}(0) &= \alpha_0, \\ \mathbf{B}'(0) &= m\alpha_1 - m\alpha_0, \\ \mathbf{B}''(0) &= m(m-1)(\alpha_2 + \alpha_0 - 2\alpha_1) \end{aligned} \tag{C1}$$

Solving for α_0, α_1, and α_2

$$\begin{aligned} \alpha_0 &= h^0_{\text{des}}, \\ \alpha_1 &= \frac{\dot{h}^0_{\text{des}}}{m} + \alpha_0, \\ \alpha_2 &= \frac{\ddot{h}_{\text{des}}0}{m(m-1)} + 2\alpha_1 - \alpha_0 \end{aligned} \tag{C2}$$

where h^0_{des}, $\dot{h}^0_{\text{des}}$, and $\ddot{h}_{\text{des}}0$ are the value and derivatives of the desired trajectory before the update. The smoothing process requires that the Bezier order should be high enough so that the influence of the smoothing is limited to only the beginning of the curve. We choose the Bezier order to be 8.

References

[1] Ames, A. D., Grizzle, J. W., and Tabuada, P., 2014, "Control Barrier Function Based Quadratic Programs With Application to Adaptive Cruise Control," 53rd IEEE Conference on Decision and Control (CDC), Los Angeles, CA, Dec. 15–17, pp. 6271–6278.

[2] Xu, X., Grizzle, J. W., Tabuada, P., and Ames, A. D., 2016, "Correctness Guarantees for the Composition of Lane Keeping and Adaptive Cruise Control," preprint arXiv: 1609.06807.

[3] Hsu, S.-C., Xu, X., and Ames, A. D., 2015, "Control Barrier Function Based Quadratic Programs With Application to Bipedal Robotic Walking," American Control Conference (ACC), Chicago, IL, July 1–3, pp. 4542–4548.

[4] Chen, Y., Peng, H., and Grizzle, J., 2017, "Obstacle Avoidance for Low-Speed Autonomous Vehicles With Barrier Function," IEEE Trans. Control Syst. Technol., **26**(1), pp. 194–206.

[5] Ames, A. D., Xu, X., Grizzle, J. W., and Tabuada, P., 2016, "Control Barrier Function Based Quadratic Programs With Application to Automotive Safety Systems," preprint arXiv: 1609.06408.

[6] Gomi, H., and Kawato, M., 1993, "Neural Network Control for a Closed-Loop System Using Feedback-Error-Learning," Neural Networks, **6**(7), pp. 933–946.

[7] Da, X., Hartley, R., and Grizzle, J. W., 2017, "Supervised Learning for Stabilizing Underactuated Bipedal Robot Locomotion, With Outdoor Experiments on the Wave Field," IEEE International Conference on Robotics and Automation (ICRA), Singapore, May 29—June 3, pp. 3476–3483.

[8] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K., 2016, "End to End Learning for Self-Driving Cars," preprint arXiv: 1604.07316.

[9] Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S., 2016, "End-to-End Deep Reinforcement Learning for Lane Keeping Assist," preprint arXiv: 1612.04340.

[10] Oh, S.-Y., Lee, J.-H., and Choi, D.-H., 2000, "A New Reinforcement Learning Vehicle Control Architecture for Vision-Based Road Following," IEEE Trans. Veh. Technol., **49**(3), pp. 997–1005.

[11] Bertsekas, D., 1995, *Dynamic Programming and Optimal Control*, **1**, Athena Scientific, Belmont, MA.

[12] Gillula, J. H., and Tomlin, C. J., 2012, "Guaranteed Safe Online Learning Via Reachability: Tracking a Ground Target Using a Quadrotor," IEEE International Conference on Robotics and Automation (ICRA), St. Paul, MN, May 14–18, pp. 2723–2730.

[13] Akametalu, A. K., Fisac, J. F., Gillula, J. H., Kaynama, S., Zeilinger, M. N., and Tomlin, C. J., 2014, "Reachability-Based Safe Learning With Gaussian Processes," IEEE 53rd Annual Conference on Decision and Control (CDC), Los Angeles, CA, Dec. 15–17, pp. 1424–1431.

[14] Smit-Anseeuw, N., Vasudevan, R., and Remy, C. D., 2017, "Safe Online Learning Using Barrier Functions," Proceedings of Dynamic Walking, Mariehamn, Finland, June 4–9.

[15] Berkenkamp, F., and Schoellig, A. P., 2015, "Safe and Robust Learning Control With Gaussian Processes," European Control Conference (ECC), Linz, Austria, July 15–17, pp. 2496–2501.

[16] Isidori, A., 2013, *Nonlinear Control Systems*, Springer Science & Business Media, London.

[17] Khalil, H. K., 1996, *Nonlinear Systems*, Vol. 2, Prentice Hall, Upper Saddle River, NJ, pp. 5–1.

[18] Westervelt, E. R., Grizzle, J. W., Chevallereau, C., Choi, J. H., and Morris, B., 2007, *Feedback Control of Dynamic Bipedal Robot Locomotion*, CRC Press, Boca Raton, FL.

[19] Shiriaev, A., Perram, J. W., and Canudas-de-Wit, C., 2005, "Constructive Tool for Orbital Stabilization of Underactuated Nonlinear Systems: Virtual Constraints Approach," IEEE Trans. Autom. Control, **50**(8), pp. 1164–1176.

[20] Maggiore, M., and Consolini, L., 2013, "Virtual Holonomic Constraints for Euler–Lagrange Systems," IEEE Trans. Autom. Control, **58**(4), pp. 1001–1008.

[21] Grizzle, J. W., Di Benedetto, M. D., and Lamnabhi-Lagarrigue, F., 1994, "Necessary Conditions for Asymptotic Tracking in Nonlinear Systems," IEEE Trans. Autom. Control, **39**(9), pp. 1782–1794.

[22] Miege, A. J., and Cebon, D., 2005, "Optimal Roll Control of an Articulated Vehicle: Theory and Model Validation," Veh. Syst. Dyn., **43**(12), pp. 867–884.

[23] Wong, J. Y., 2008, *Theory of Ground Vehicles*, Wiley, Hoboken, NJ.

[24] Xu, X., Tabuada, P., Grizzle, J. W., and Ames, A. D., 2015, "Robustness of Control Barrier Functions for Safety Critical Control," IFAC-PapersOnLine, **48**(27), pp. 54–61.

[25] Parrilo, P. A., 2003, "Semidefinite Programming Relaxations for Semialgebraic Problems," Math. Program., **96**(2), pp. 293–320.

[26] Bochnak, J., Coste, M., and Roy, M.-F., 2013, *Real Algebraic Geometry*, Vol. 36, Springer Science & Business Media, New York.

[27] Stengle, G., 1974, "A Nullstellensatz and a Positivstellensatz in Semialgebraic Geometry," Math. Ann., **207**(2), pp. 87–97.

[28] Papachristodoulou, A., and Prajna, S., 2002, "On the Construction of Lyapunov Functions Using the Sum of Squares Decomposition," 41st IEEE Conference on Decision and Control (CDC), Las Vegas, NV, Dec. 10–13, pp. 3482–3487.

[29] Prajna, S., and Jadbabaie, A., 2004, "Safety Verification of Hybrid Systems Using Barrier Certificates," International Workshop on Hybrid Systems: Computation and Control (HSCC), Philadelphia, PA, Mar. 25–27, pp. 477–492.

[30] Prajna, S., Papachristodoulou, A., and Wu, F., 2004, "Nonlinear Control Synthesis by Sum of Squares Optimization: A Lyapunov-Based Approach," Fifth Asian Control Conference (ASCC), Melbourne, Australia, July 20–23 , pp. 157–165.

[31] Goh, K. C., Turan, L., Safonov, M. G., Papavassilopoulos, G. P., and Ly, J. H., 1994, "Biaffine Matrix Inequality Properties and Computational Methods," American Control Conference (ACC), Baltimore, MD, June 29–July 1, pp. 850–855.

[32] Chen, Y., Peng, H., and Grizzle, J. W., 2018, "Validating Noncooperative Control Designs Through a Lyapunov Approach," IEEE Trans. Control Syst. Technol., **27**(2), pp. 527–539.

[33] Betts, J. T., 2010, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, SIAM, Philadelphia, PA.

[34] Hereid, A., Cousineau, E. A., Hubicki, C. M., and Ames, A. D., 2016, "3D Dynamic Walking With Underactuated Humanoid Robots: A Direct Collocation Framework for Optimizing Hybrid Zero Dynamics," IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, May 16–21, pp. 1447–1454.

[35] Ames, A. D., Xu, X., Grizzle, J. W., and Tabuada, P., 2017, "Control Barrier Function Based Quadratic Programs for Safety Critical Systems," IEEE Trans. Autom. Control, **62**(8), pp. 3861–3876.

[36] Stoer, J., and Bulirsch, R., 2013, *Introduction to Numerical Analysis*, Vol. 12, Springer Science & Business Media, New York.

[37] Hereid, A., and Ames, A. D., 2017, "FROST*: Fast Robot Optimization and Simulation Toolkit," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, Sept. 24–28, pp. 719–726.

[38] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y.,

and Zheng, X., 2016, “Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” preprint arXiv: 1603.04467.

[39] Frazzoli, E., Dahleh, M. A., and Feron, E., 2002, “Real-Time Motion Planning for Agile Autonomous Vehicles,” J. Guid., Control, Dyn., **25**(1), pp. 116–129.

[40] Tabuada, P., 2007, “Event-Triggered Real-Time Scheduling of Stabilizing Control Tasks,” IEEE Trans. Autom. Control, **52**(9), pp. 1680–1685.

[41] Da, X., and Grizzle, J., 2017, “Combining Trajectory Optimization, Supervised Machine Learning, and Model Structure for Mitigating the Curse of Dimensionality in the Control of Bipedal Robots,” preprint arXiv: 1711.02223.

[42] Arnol’d, V. I., 2013, *Mathematical Methods of Classical Mechanics*, Vol. 60, Springer Science & Business Media, New York.