Synthesis of safe controller via supervised learning for truck lateral control

Yuxiao Chen[*a], Ayonga Hereid[b], Huei Peng[a] and Jessy Grizzle[b]

***Abstract*—Correct-by-construction techniques such as control barrier function (CBF) have been developed to guarantee safety for control systems as supervisory controller. However, when the supervisor intervenes, the performance is typically compromised. On the other hand, machine learning is used to synthesize controllers that inherit good properties from the training data, but safety is typically not guaranteed due to the difficulty of analysis. In this paper, supervised learning is combined with CBF to synthesize controllers that enjoy good performance with safety guarantee. First, a training set is generated by trajectory optimization that incorporates the CBF constraint for multiple initial conditions. Then a policy is trained via supervised learning that maps the feature representing the initial condition to a parameterized desired trajectory. Finally, the learning based controller is used as the 'student' controller, and a CBF based supervisory controller on top of that guarantees safety. A case study of lane keeping for articulated trucks shows that the 'student' controller trained by the supervised learning inherits the good performance of the training set and the CBF supervisor never or rarely intervenes.**

***Index Terms*—Control Barrier Function, Supervised Learning, trajectory optimization**

I. Introduction

CORRECT-by-construction control synthesis has been a promising direction of research that brings formalism and safety guarantee to controller design. Among which, Control Barrier Functions (CBF) is a typical technique developed to impose safety in a plug-and-play fashion [1, 2]. The key idea is to compute a forward invariant set that contains the safe set and excludes the danger set. CBF is typically implemented in a supervisory control structure to guarantee safety with minimum interventions. As shown in Fig. 1. u_0 denotes the control input from the student controller, which can be designed by any existing method, and u denotes the input signal after the intervention of the supervisory controller. If u_0 respects the safety constraint, $u = u_0$; otherwise a minimum intervention is applied. Depending on the form of the barrier function, the implementation can be quadratic programming [1, 3], mixed integer programming [4] or in other forms.

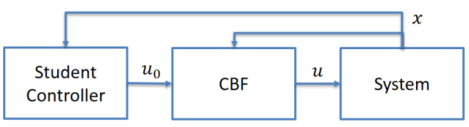


Fig. 1. Block diagram of the supervisory control

Since the student can be legacy controllers that are already designed, the supervisory control structure provides a plug-and-play type of safety guarantee to existing control systems. When safety is not threatened, the CBF lets the student controller control the system and therefore the good properties of the student controller are intact as long as it does not lead to danger.

However, if the student controller is not properly designed, or is designed in a way that is not compatible with the CBF, the CBF may get triggered frequently and typically lead to undesirable performance. Moreover, since the CBF is typically designed in a myopia fashion, meaning that the CBF only looks at current state, the last minute intervention to guarantee safety can be dramatic. In [5], for simple ACC dynamics, the CBF input shows a few spikes when activated. In [6], when the student controller is designed without any obstacle avoidance consideration, the CBF has to intervene frequently and severely to ensure obstacle avoidance. These examples on one hand demonstrate the power of CBF as a safety guarantee, but they also show the room for improvement. If the student controller is designed in a way that takes the CBF into account, the interventions can be reduced and the system performance can be improved.

On the other hand, machine learning has been used extensively in system control. Supervised learning can be used to learn the control policy with structure [7, 8], deep learning recently was used to generate end-to-end LK policy, i.e., a mapping directly from the pixels captured by the camera to the steering input [9], and reinforcement learning can be used to generate control policy by a 'explore and evaluate' manner [10-12]. However, one major deficit of machine learning is its 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 set and barrier function. In this sense, machine learning and CBF are good complement to each other. Existing methods that combine learning with safety guarantee includes HJB reachable set based learning scheme that can guarantee safety for online learning of control policy [13], and barrier function based online learning scheme [14]. This paper aims at problems related to trajectory planning and tracking. The method we propose is to perform trajectory optimization offline, generate a trajectory library consisting of trajectories with good properties, including satisfying the CBF condition, and use supervised learning to train a student controller that inherits the properties of the trajectory library. Then CBF acts as a supervisor on top of the learning based controller, as shown in Fig. 1. Since the CBF condition is enforced in the training set, the intervention from the supervisor is rarely triggered.

The main contribution of this paper are the following two points. First, we propose a supervised learning based scheme to design student controller that takes the CBF condition into account, applicable to a large region of initial conditions, and rarely triggers the intervention from supervisory controller. With supervised learning, the design of a safe student controller is transformed into design of safe trajectories, which is much easier, as shown in Fig. 2.

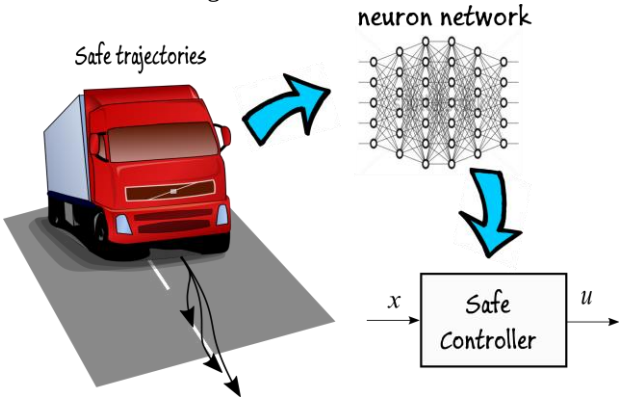


Fig. 2. Structure of the proposed supervisory control

Second, 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 are able to provide safety guarantee to Lane Keeping control (LK) of an articulated truck while achieving good ride comfort.

The remainder of the paper is organized as follows. Section II presents the lateral dynamic model for the articulated truck, Section III introduces the synthesis process of the CBF, Section IV presents the CH feedback controller and provides analysis on its stability and other properties, Section V presents the trajectory optimization with direct collocation for the LK control, Section VI presents the implementation of learning based controller, Section VII shows the simulation result with Trucksim, and finally we draw the conclusion in Section VIII.

II. DYNAMICS MODEL

In this section, we present the dynamic model for the articulated truck.

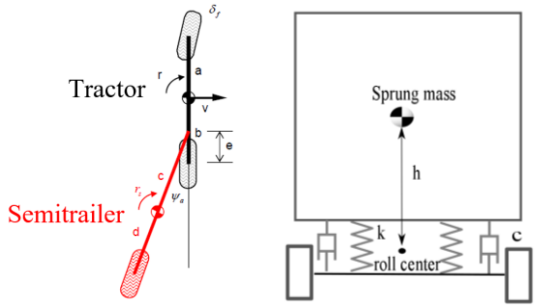


Fig. 3. Lateral-yaw-roll model of articulated truck

Fig. 3 shows the lateral-yaw, and roll motion of the truck. We make the following assumptions about the model:

- 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 semitrailer are the same;
- The pitch and vertical motion are weakly coupled with the lateral, yaw and roll motion, and are ignored in the model;
- The angles are small, therefore the model can be approximated by a linear dynamic model.

The state of the model are chosen as

$$x = \begin{bmatrix} y & v_y & \psi & r & \psi_a & r_s & \phi & p \end{bmatrix}^T, \qquad (1)$$

where y is the lateral deviation from the lane center to the tractor CG, 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 semitrailer), r_s is the yaw rate of the semitrailer, ϕ is the roll angle and p is the roll rate. The yaw motion has the following relationship:

$$\psi_s = \psi + \psi_a, \dot{\psi}_a = r_s - r, \qquad (2)$$

where ψ_s is the heading angle of the semitrailer.

The linear dynamic model takes the steering angle of the front axle δ_f as control input, and there are two exogenous disturbance: road yaw rate r_d and side-wind F_y.

$$\dot{x} = Ax + Bu + Ed, u = \delta_f, d = \begin{bmatrix} r_d & F_y \end{bmatrix}^T, \qquad (3)$$

The linear model is identified as a grey box, i.e. a model that maintains a certain structure. All zero entries remains zero, while all physics based parameters are identified with the data from Trucksim, which is a physics based high fidelity model used widely by the industry. The model matching is shown in Fig. 4.

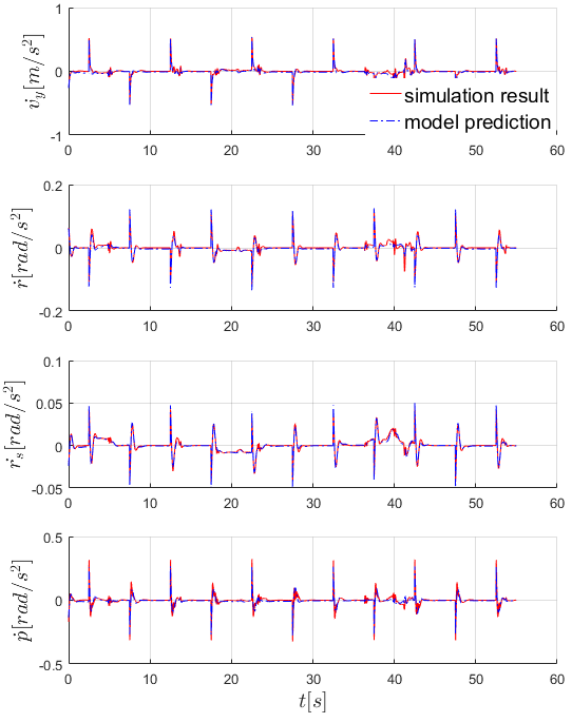


Fig. 4. System Identification of the truck model

Bounds are imposed on the input and the disturbance as $u \in \mathcal{U}, d \in \mathcal{D}$, where $\mathcal{U}$ and $\mathcal{D}$ are semialgebraic sets. The

key parameters are listed in TABLE I.

TABLE I LK PROBLEM PARAMETERS

v_x	$20m/s$
Bound on y	$\pm 0.3m$
Bound on v_y	$\pm 2m/s$
Bound on ψ	$\pm 0.04rad$
Bound on r	$\pm 0.25rad/s$
Bound on ψ_a	$\pm 0.04rad$
Bound on r_s	$\pm 0.25rad/s$
Bound on ϕ	$\pm 0.1rad$
Bound on p	$\pm 0.3rad/s$
Bound on r_d	$\pm 0.02rad/s$ (turning radius of 1000m)
Bound on F_y	$\pm 2000N$
Bound on δ_f	$\pm 0.2rad$

III. SYNTHESIS OF CBF

In this section, we discuss the synthesis of the CBF. The supervisory controller solves the following optimization problem:

$$\begin{aligned} &\min_u \|u-u_0\| \\ &s.t.\ \mathcal{C}(x,d,u)\geq 0 \end{aligned}, \tag{4}$$

where $\mathcal{C}(x,d,u)\geq 0$ denotes the CBF constraint, and $\|u-u_0\|$ is the norm of the intervention. The CBF constraint is a constraint of input u, and may depend on current state and exogenous disturbances. Here we use the zeroing CBF introduced in [15]. The 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

$$\dot{b}+\kappa\alpha(b)\geq 0, \tag{5}$$

where $\kappa>0$ is a positive constant, and $\alpha(b)$ is an extended class $\mathcal{K}$ function. An extended class $\mathcal{K}$ function $f:\mathbb{R}\to\mathbb{R}$ satisfies

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

When $b(x)>0$, $\dot{b}$ can be negative, but is lower bounded by $-\kappa\alpha(b)$; at the boundary, $\dot{b}$ should be nonnegative, which make the set $\{x\,|\,b(x)\geq 0\}$ controlled invariant; and if $b(x)<0$, the condition in (5) enforces convergence to the set $\{x\,|\,b(x)\geq 0\}$ by setting upper bound $\dot{b}\geq -\kappa\alpha(b)>0$.

The synthesis of CBF is not trivial. The key idea is to find $b(x)$ such that the barrier condition is always feasible inside $\{x\,|\,b(x)\geq 0\}$, which makes the set robustly controlled invariant. The truck lateral dynamics can be approximated with a linear model, and we use Sum of Squares (SOS) technique to synthesize the CBF.

A. Brief introduction of SOS programming

SOS has been widely used in the computation of invariant sets and barrier functions for continuous dynamic systems, and it can be efficiently solved with semidefinite programming (SDP). In addition, Positivestallensatz can enforce SOS condition on semialgebraic sets via multipliers [16].

Here we present a brief introduction of SOS techniques used in our approach. We denote $\Sigma[x]$ as the cone of SOS polynomials, a subset of $\mathbb{R}[x]$ which is the space of all polynomials of x. For a SOS problem to be solvable by SDP, all the coefficients should appear as linear functions of the optimization variables, otherwise the problem may not be convex. The SOS constraint is transformed to a SDP constraint, and solved by SDP solvers.

To enforce a SOS constraint on a semialgebraic set, SOS multiplier is used. E.g., $f(x)-s(x)h(x)\in\Sigma[x], s(x)\in\Sigma[x]$ with SOS multiplier $s(x)$ is a sufficient condition that $f(x)$ is nonnegative when $h(x)\geq 0$. Indeed, since $s(x)$ is SOS, it is nonnegative; when $h(x)\geq 0$, it implies $f(x)\geq s(x)h(x)\geq 0$. Positivestallensatz states that the condition is also necessary with sufficient number and order of multipliers. For more information, see [2, 17-22] . However, the due to the limited computation power of the SDP solver, the degree of the multipliers is limited, and some multipliers make the problem nonconvex, therefore we merely use it as a sufficient condition.

B. SOS for CBF

We focus on dynamic system with the following affine structure:

$$\begin{aligned} &\dot{x}=f(x)+g(x)u+g_d(x)d \\ &x\in\mathbb{R}^n, u\in\mathcal{U}\subseteq\mathbb{R}^m, d\in\mathcal{D}\subseteq\mathbb{R}^l \end{aligned}, \tag{6}$$

where $\mathcal{U}$ and $\mathcal{D}$ are known semialgebraic sets. In CBF synthesis, we set $\alpha(b)=b$, and seek a polynomial CBF $b(x)$ that satisfies the following:

$$\{b(x)\geq 0\}\subseteq X_{safe};, \tag{7}$$

$$\forall x\in\{b(x)\geq 0\}, \forall d\in\mathcal{D}, \exists u\in\mathcal{U},\ s.t.\ \frac{db}{dx}(f(x)+g(x)u+g_d(x)d)+\kappa b\geq 0, \tag{8}$$

where X_{safe} is the danger set, $\kappa>0$ is a positive constant, and (8)is referred to as the CBF condition. In the LK case, X_{safe} is defined by the bounds for each of the 8 states, which forms a hyper-box in $\mathbb{R}^8$.

In practice, the condition in (8) is hard to implement as a SOS program due to the existence condition of u, so we seek a

conservative approximation, in which we assume the control input u comes from a polynomial controller of x and d.

$$\begin{aligned}&\{b(x)\geq 0\}\subseteq X_{safe};\\&\forall x\in\{b(x)\geq 0\}, u(x,d)\in\mathcal{U};\\&\forall x\in\{b(x)\geq 0\}, \forall d\in\mathcal{D},\\&\frac{db}{dx}\left(f(x)+g(x)u(x,d)+g_d(x)d\right)+\kappa b(x)\geq 0.\end{aligned}\tag{9}$$

The input may depend on measured disturbance, but not on unmeasured disturbance.

Even with the simplification, there are two bilinear terms that should be handled to make the problem solvable by SOS. The first bilinear term is between $b(x)$ and $u(x,d)$. We use bilinear alternation [2, 23], which iterates the following two steps:

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

Suppose $\mathcal{U}$ and $\mathcal{D}$ are semialgebraic sets of the following form:

$$\mathcal{U}=\{u\mid h_u(u)\geq 0\},\ \mathcal{D}=\{u\mid h_d(u)\geq 0\}\tag{10}$$

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

$$\begin{aligned}&\min e \text{ s.t.}\\&h_u(u(x,d))-s_1(x,d)b(x)-\sum_i s_2^i(x,d)h_d^i(d)\in\Sigma[x,d]\\&\frac{db}{dx}\left(f(x)+g(x)u(x,d)+g_d(x)d\right)+\kappa b(x)+s_3(x,d)b(x)\\&-\sum_i s_4^i(x,d)h_d^i(d)+eQ(x,d)\in\Sigma[x,d]\\&s_1,s_2,s_3,s_4\in\Sigma[x,d],\end{aligned}\tag{11}$$

where s_1, s_2, s_3, s_4 are the SOS multipliers. $Q(x,d)$ is a fixed SOS polynomial of x and d; e is a relaxation scalar variable that makes this SOS program feasible. When $e\leq 0$, (11) is a sufficient condition of (9). s_3 is used to enforce the CBF condition only when $b(x)\geq 0$. The first SOS constraint restricts the input to be bounded by $\mathcal{U}$; the second SOS constraint enforces the CBF condition.

Remark 1: $Q(x,d)$ depends on the order of the polynomial required to be SOS, in many cases, $Q(x,d)$ can simply be x^Tx.

The algorithm then searches for a better CBF with a fixed controller. Meanwhile, the second bilinear term emerges from the constraint that the CBF is enforced only when $b(x)\geq 0$, which creates a bilinear term between $b(x)$ and the SOS multiplier. We use perturbation to solve this bilinear term. Suppose X_d, the compliment of X_{safe}, is an semialgebraic set defined as

$$X_d=\{x\mid h_{xd}(x)\geq 0\}\tag{12}$$

The following SOS program solves for small a perturbation to the existing barrier candidate that helps reducing e:

$$\begin{aligned}&\min e \text{ s.t.}\\&-b_0(x)-\Delta b(x)-\sum_i s_1^i(x)h_{xd}^i(x)\in\Sigma[x]\\&\frac{d(b+\Delta b)}{dx}\left(f(x)+g(x)u(x,d)+g_d(x)d\right)+\kappa(b_0+\Delta b)(x)\\&-\sum_i s_2^i(x,d)h_d^i(x)+s_3(x,d)b_0(x)+eQ(x,d)\in\Sigma[x,d]\\&s_1\in\Sigma[x],\ s_2,s_3\in\Sigma[x,d],\ \|\Delta b\|\leq\epsilon\|b_0\|,\end{aligned}\tag{13}$$

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

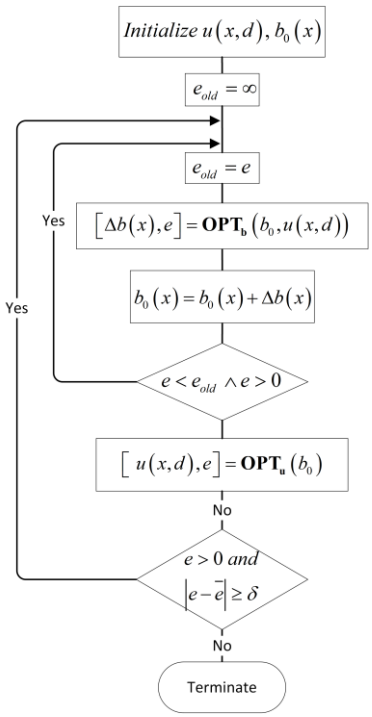


Fig. 5. Synthesis of CBF with SOS

In summary, there are two loops in the algorithm. The inner loop iterates the perturbation process, updating b_0 with $b_0+\Delta b$ until no progress can be made; the outer loop iterates between updating $b(x)$ and updating $u(x,d)$. Denote the optimization in (11) as $[u(x,d),e]=\mathbf{OPT_u}(b(x))$, with $b(x)$ as input, $u(x,d)$ and e as output; and denote the optimization in (13) as $[\Delta b(x),e]=\mathbf{OPT_b}(b(x),u(x,d))$, with $b(x)$ and $u(x,d)$ as input, $\Delta b(x)$ and e as output. The iteration terminates when a valid CBF is found or no improvement can be made, as shown in Fig. 5.

IV. CONTINUOUS HOLD FEEDBACK CONTROL

In this section, we propose a continuous hold (CH) controller which will be the structure of the student controller, and its properties are analyzed. The name continuous hold comes from the comparison with zero order hold and n^{th} order hold. While n^{th} order hold approximates the segment between two consecutive sampling time with n^{th} order polynomial, continuous hold executes a predefined continuous trajectory. The idea of continuous hold is not a new one, e.g. motion primitive is a special type of continuous hold [24]. The trajectory is updated in an event-triggered fashion, which will be discussed in detail in Section VI.A. Event-triggered finite horizon control is studied in [25], but in the CH setting, the control between triggering event is continuous. Here we consider a continuous hold controller that does not discretize the state space or input space. This idea, together with continuous update controller is discussed in [26]. We extend the idea in [26] to systems with exogenous disturbance and analyze the controller's stability and its ability to inherit the set invariance property from the training set.

A. *Continuous hold controller for system with disturbance*

System of the following form is considered:

$$\dot{x} = f(t,x,u,d). \tag{14}$$

We make the following assumptions about the system dynamics.

A-1: $f:[0,\infty)\times\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^p\to\mathbb{R}^n$ is locally Lipschitz continuous in t, x, u and d, and there exists $T_p>0$ such that $\forall(t,x,u,d)$, $f(t+T_p,x,u,d)=f(t,x,u,d)$.

Remark 2: The periodicity is trivially satisfied by any time invariant system.

A-2 : $\exists D\subseteq\mathbb{R}^p, \text{s.t. } \forall d\in D, \exists! x\in\mathbb{R}^n, \exists! u\in\mathbb{R}^m, \forall t\in[0,\infty)$, $f(t,x,u,d)=0$, i.e., for every exogenous disturbance $d\in D$, there exists a unique equilibrium point with unique input that maintains the equilibrium. Let $\eta_x: D\to\mathbb{R}^n, \eta_u: D\to\mathbb{R}^p$ denote the mapping from exogenous disturbance to the corresponding equilibrium point and input so that $f(t,\eta_x(d),\eta_u(u),d)=0$. Note that the equilibrium point and the input is independent of t.

A-3: $\forall d\in D$, there is an open ball about $\eta_x(d)$, $B_d\subset\mathbb{R}^n$, and let $\overline{B_d}$ be B_d shifted back to the origin. There exists a positive-definite, locally Lipschitz-continuous function $V_d:\overline{B_d}\to\mathbb{R}$, and constants $0\le\alpha_1^d\le\alpha_2^d$ such that $\forall x\in B_d$,

$$\begin{aligned}&\bar{x}=x-\eta_x(d)\\&\alpha_1^d\bar{x}^T\bar{x}\le V_d(\bar{x})\le\alpha_2^d\bar{x}^T\bar{x}\end{aligned}. \tag{15}$$

A-4: $\exists S\subseteq\mathbb{R}^n$, compact, such that $\forall d\in D, \eta_x(d)\in S$. There exists CBF $b(x)$, such that $\forall x\notin S, b(x)\le 0$, $\forall d\in D$, $b(\eta_x(d))>0$. Moreover, $\forall\xi\in S, \forall d\in D$, there exists $u_\xi^d:[0,T_p]\to\mathbb{R}^m$ and a corresponding state trajectory $\varphi_\xi^d:[0,T_p]\to\mathbb{R}^n$ satisfying

$$\begin{aligned}&V_d(\varphi_\xi^d(T_p)-\eta_x(d))\le c_1V_d(\varphi_\xi^d(0)-\eta_x(d))\\&\left.\frac{db}{dx}\right|_{\varphi_\xi^d(t)}\dot{\varphi}_\xi^d(t)+\kappa b(\varphi_\xi^d(t))\ge 0\\&\lim_{\xi\to\eta_x(d)}\varphi_\xi^d(t)=\eta_x(d),\ \lim_{\xi\to\eta_x(d)}u_\xi^d(t)\equiv\eta_u(d)\end{aligned}, \tag{16}$$

where $\kappa>0$, $1>c_1>0$ are a predefined constants.

A CH controller keeps a timer $\hat{t}$ that is reset to 0 when the triggering event happens and desired trajectory is updated, then keep flowing as the trajectory is being executed. The update can happen when the trajectory is executed to the 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

$$u^{CH}(t,x,d)=u_\xi^d(\hat{t})+u^{fb}(x(t),\varphi_\xi^d(\hat{t})), \tag{17}$$

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

The closed loop system under CH feedback is then

$$\dot{x}=f^{CH}(t,x,d):=f(t,x,u_\xi^d(\hat{t})+u^{fb}(x(t),\varphi_\xi^d(\hat{t})),d). \tag{18}$$

A-5: For any trajectory φ_ξ^d in **A-4** 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 (18) satisfies

$$\begin{aligned}&\exists B\subseteq\mathbb{R}^n, s.t.\ \forall 0\le t_1\le t_2\le T_p, (x(t_1)-\varphi_\xi^d(t_1))\in B,\\&\|x(t_2)-\varphi_\xi^d(t_2)\|\le e^{-c_2(t_2-t_1)}\|x(t_1)-\varphi_\xi^d(t_1)\|\end{aligned} \tag{19}$$

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.

Proposition 1: Under assumptions **A-1**, **A-2**, **A-3**, **A-4** and **A-5**, for an initial condition $\xi\in\{x\,|\,b(x)\ge 0\}$ the closed loop system in (18) will stay inside $\{x\,|\,b(x)\ge 0\}$, and if d stops changing after $T\ge 0$, the state will converge to $\eta_x(d)$ exponentially.

Proof: From **A-4**, since $\xi\in\{x\,|\,b(x)\ge 0\}$, $\xi\in S$, which means the feedback control in (17) is well defined. From (16), the CBF $b(x)$ remains nonnegative as discussed in Section II, which proves that the state will stay inside $\{x\,|\,b(x)\ge 0\}$, and thus $x(t)\in S, \forall t\ge 0$.

When d stops changing, from the Lyapunov condition in **A-4**, $V(x(nT_p)-\eta_x(d))\le c_1^{n-1}V(x(0)-\eta_x(d))$, $n=1,2,3,\ldots$, which implies $\lim_{n\to\infty}V(x(nT_p))=0$. From **A-3**, the sequence $x(nT_p)$ converges to $\eta_x(d)$. Therefore from the last assumption in (16), the state stays at $\eta_x(d)$.

■

B. State decomposition and dimension reduction

It is impractical to compute input and state trajectory for every initial condition for a continuous state space, since there are uncountably many of them. We would draw samples in the state space and use the corresponding trajectories as the training set for supervised learning, then a continuous mapping from state to trajectory can be obtained via supervised learning. However, under a grid fashion of drawing samples, the number of samples needed grows exponentially with the state dimension. To reduce the number of samples needed, we decompose the states into two groups: $x=[x_1;x_2]$, 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 tracking feedback u^{fb} is 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

$$\begin{aligned}\forall d\in D,\eta_x(d)&=\left[\eta_x^1(d);\eta_x^2(d)\right],\\ \eta_x^2(d)&=\gamma\left(\eta_x^1(d)\right)\end{aligned}\tag{20}$$

is called an *insertion map*. To extend the previous conclusion to cases where trajectories are parameterized with only x_1, we make the following assumptions.

A-6: $\exists S\subseteq\mathbb{R}^n$, compact, such that $\forall d\in D,\eta_x(d)\in S$. There exists CBF $b(x)$, *s.t.* $\forall x\notin S, b(x)\le 0$, $\forall d\in D, b\left(\eta_x(d)\right)\ge 0$, $\forall d\in D,\forall\xi=[\xi_1;\xi_2]\in S,\quad \xi_2=\gamma(\xi_1)$, and there exists $u_{\xi_1}^d:\left[0,T_p\right]\to\mathbb{R}^m$ and a corresponding state trajectory, $\varphi_{\xi_1}^d:\left[0,T_p\right]\to\mathbb{R}^n$ satisfying

$$\begin{aligned}&V_d\left(\varphi_{\xi_1}^d(T_p)-\eta_x(d)\right)\le cV_d\left(\varphi_{\xi_1}^d(0)-\eta_x(d)\right)\\ &\frac{db}{dx}|_{\varphi_{\xi_1}^d(t)}\dot{\varphi}_{\xi_1}^d(t)+\kappa\varphi_{\xi_1}^d(t)\ge 0\end{aligned},\tag{21}$$

$$\begin{aligned}&\lim_{[\xi_1,\gamma(\xi_1)]\to\eta_x(d)}\varphi_\xi^d(t)=\eta_x(d),\\ &\lim_{[\xi_1,\gamma(\xi_1)]\to\eta_x(d)}u_\xi^d(t)\equiv\eta_u(d)\end{aligned},\tag{22}$$

$$\varphi_{2\xi_1}^d(T_p)=\gamma\left(\varphi_{1\xi_1}^d(T_p)\right).\tag{23}$$

A-7: There exists a feedback $u_1^{fb}:\mathbb{R}^{n_1}\times\mathbb{R}^{n_1}\to\mathbb{R}^m$ that $u_1^{fb}\left(x_1(t),\varphi_{1\xi_1}^d(\hat{t})\right)$ makes $\varphi_{\xi_1}^d$ uniformly locally exponentially stable, i.e. (19) is satisfied with $u(t)=u_\xi^d(\hat{t})+u_1^{fb}\left(x_1(t),\varphi_{1\xi_1}^d(\hat{t})\right)$.

Remark 3: 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, $\varphi_{\xi_1}^d=\left[\varphi_{1\xi_1}^d;\varphi_{2\xi_1}^d\right]$. **A-7** is possible if the dynamic subsystem of x_2 is locally exponentially stable.

Proposition 2: Under **A-1**, **A-2**, **A-3**, **A-6** and **A-7**, $\forall d\in D$, $\forall\xi=\left[\xi_1;\gamma(\xi_1)\right]\in\left\{x\mid b(x)\ge 0\right\}$, the closed loop system under CH feedback will stay inside $\left\{x\mid b(x)\ge 0\right\}$, and if d stops changing after some $T\ge 0$, the state will converge to $\eta_x(d)$ exponentially.

Proof: By **A-7**, the closed loop system exponentially converges to the CH desired trajectory. From **A-6**, by CBF condition, $\left\{x\mid b(x)\ge 0\right\}$ is invariant under the CH controller. When d stops changing, the closed loop system exponentially converges to φ_ξ^d and $V\left(x(nT_p)-\eta_x(d)\right)\le c^{n-1}V\left(\xi-\eta_x(d)\right)$, for $n=1,2,3,\ldots$, and satisfies $x_2(nT_p)=\gamma\left(x_1(nT_p)\right)$. 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, (20) makes sure that when $x_1\to\eta_x^1(d)$, $\gamma(x_1)\to\eta_x^2(d)$. By (22), $x(t)$ converges to $\eta_x(d)$ exponentially.

■

Remark 4: 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.

Remark 5: All the above mentioned assumptions will be enforced in the trajectory optimization process discussed in Section V.

V. Virtual Constraint and Trajectory Optimization

Based on the analysis of Section. IV, trajectories that satisfies the stability and safety conditions for various initial conditions need to be constructed. In this section, the trajectory optimization setup is presented.

In order to simplify the parameterization of the trajectory, feedback linearization is used to reduce the order of the system dynamics. After feedback linearization, the system is simplified to a double integrator. Virtual constraint together with Bezier curve is used to parameterize the desired trajectory. The trajectory optimization follows direct collocation framework to generate desired trajectories for multiple initial conditions, which are then used to train a mapping from initial conditions to desired trajectories via supervised learning.

A. Virtual constraint

It is hard to parameterize the trajectories of all 8 states of the truck, therefore a reduced order representation of the state trajectory is needed. We employ the virtual constraint approach commonly used in the robotics literature [27, 28]. First, feedback linearization is used to decompose the dynamics. For the lateral-yaw-roll dynamics of the articulated truck, we pick the preview of lateral displacement as the output:

$$z=y+T_0v_x\psi,\tag{24}$$

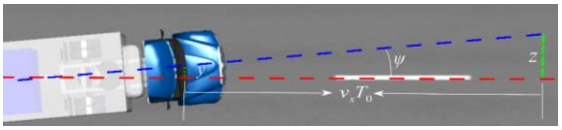


Fig. 6. Lateral-yaw-roll model of articulated truck

where T_0 is the preview time and v_x is the forward speed. To connect our model with the standard feedback linearization literature, (3) is rewritten in the general nonlinear form:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u + g_d(x)d \\ z &= h(x) = Cx \end{aligned}, \tag{25}$$

where $f(x) = Ax$, $g(x) = B$, $C = \begin{bmatrix} 1 & 0 & T_0 v_x & \mathbf{0}_{1\times 5} \end{bmatrix}$, and $g_d(x) = E$. Here we assume $F_y = 0$ because it is not measured, therefore cannot be countered by feedback linearization.

The output z has relative degree 2, i.e.,

$$\mathcal{L}_g h = CB = 0,\ \mathcal{L}_g \mathcal{L}_f h = CAB \neq 0, \tag{26}$$

where $\mathcal{L}$ denotes the Lie derivative. To be more specific,

$$\begin{aligned} z &= Cx \\ \dot{z} &= CAx + CE_1 r_d \\ \ddot{z} &= CA^2 x + CABu + CAE_1 r_d + CE_1 \dot{r}_d \end{aligned}, \tag{27}$$

where E_1 is the first column of E matrix. We assume $\dot{r}_d = 0$ since no measurement of $\dot{r}_d$ is available. To show the zero dynamics, we use the following state transformation that renders a new choice of states:

$$\chi = Tx = \begin{bmatrix} z & \dot{z} & \sigma^T \end{bmatrix}^T, \tag{28}$$

where T is a full rank matrix:

$$T = \begin{bmatrix} 1 & 0 & v_x T_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & v_x & v_x T_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -B^4 & 0 & B^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -B^6 & 0 & 0 & 0 & B^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -B^8 & 0 & 0 & 0 & 0 & 0 & B^2 \end{bmatrix}, \tag{29}$$

and B^i is the i^{th} entry of B. The transformation is linear and full rank. The dynamics under χ is

$$\dot{\chi} = \bar{A}\chi + \bar{B}u + \bar{E}r_d, \tag{30}$$

Moreover,

$$\bar{B} = TB = \begin{bmatrix} 0 & CAB & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T, \tag{31}$$

so the dynamics under χ can be written as

$$\dot{\chi} = \begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\sigma} \end{bmatrix} = \begin{bmatrix} \dot{z} \\ CA^2 x \\ \Gamma(z, \dot{z}, \sigma) \end{bmatrix} + \begin{bmatrix} 0 \\ CAB \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ CAE_1 \\ 0 \end{bmatrix} r_d, \tag{32}$$

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

Next, we use virtual constraint to shape the trajectory of the output. The following virtual constraint is imposed on z:

$$z = h_d(t), \tag{33}$$

where $h_d \in C^2$. Differentiate (33) twice:

$$\begin{aligned} z &= h_d(t) \\ \dot{z} &= \dot{h}_d(t) \\ \ddot{z} &= \ddot{h}_d = CA^2 x + CABu + CAE_1 r_d \end{aligned}. \tag{34}$$

Pick the input as

$$\begin{aligned} u^\star &= \frac{\ddot{h}_d - CA^2 x - CAE_1 r_d}{CAB}, \\ u_{cl} &= \frac{-K_p(z - h_d(t)) - K_d(\dot{z} - \dot{h}_d(t))}{CAB}, \\ u &= u^\star + u_{cl} \end{aligned} \tag{35}$$

where $u^\star$ is the feedforward part and u_{cl} is the PD feedback part, with K_p and K_d as the PD feedback gain. With this input, the system dynamic is simplified to

$$\begin{bmatrix} \dot{z} - \dot{h}_d \\ \ddot{z} - \ddot{h}_d \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -K_p & -K_d \end{bmatrix} \begin{bmatrix} z - h_d \\ \dot{z} - \dot{h}_d \end{bmatrix}. \tag{36}$$

The stable zero dynamic along with the PD control makes the system stable with feedback that only depends on z and $\dot{z}$. To this end, if we have a desired h_d trajectory, the system behaves like a double integrator, and will asymptotically converge to h_d under the PD control.

Remark 6: Since the zero dynamics is stable, $z(t) \to h_d(t)$ for $h_d \in C^2$ implies $x(t) \to x_d(t)$, where x_d is the desired state trajectory corresponding to h_d. This implies that we only need two states to determine the asymptotic behavior of the system, but not necessarily the transient dynamics. In practice, the more states we use to parameterize the initial condition, the finer the trajectory library will be. We pick x_1 that contains 5 states because the computation needed to generate the trajectory library is manageable (about 20 hours on a desktop). With more computation power, a higher dimensional x_1 can lead to a finer trajectory library.

The desired trajectory h_d is parameterized as Bezier curve, which is widely used in computer graphics and related fields. A Bezier curve of order N is defined on $[0,1]$ as

$$\mathbf{B}(s) = \sum_{i=0}^{N} \alpha_i \binom{N}{i} s^i (1-s)^{N-i}, \tag{37}$$

where α_i are the Bezier coefficients. It is a parameterized polynomial of order N. Compared to the parameterization with standard monomial bases ($[1, s, s^2, \ldots]$), Bezier curve has the following nice properties:

- The representation of initial and final values are simple: $\mathbf{B}(0) = \alpha_0, \mathbf{B}(1) = \alpha_N$,
- The derivative of Bezier curve is also a Bezier curve, and the dependence of the new Bezier coefficient on the original Bezier coefficient is sparse.

Note that Bezier curve can parameterize trajectories of any length by scaling the input. Suppose the horizon of h_d is T, then the input is defined as $s = t/T$.

B. Direct collocation for trajectory optimization

The above discussion about feedback linearization and virtual constraint is useful in tracking of the desired trajectory, but it hides the zero dynamics, which might cause problem. The excitation of x_1 on x_2 should not be ignored since the CBF depends on x_2 as well. Therefore in the trajectory optimization problem, we consider the full dynamics of the system. Direct collocation is used to optimize the trajectory, while enforcing the virtual constraint.

The direct collocation method has been widely employed in trajectory optimization problems due to its effectiveness and robustness. 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 (NLP)[29].

In this paper, we utilize a modified Hermite-Simpson scheme based direct collocation trajectory optimization method[30]. Particularly, the flow (a.k.a. trajectory), $x(t)$, of the continuous dynamical system in (3) 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, they must satisfy the system dynamics equation given in (3). Further, if these discrete states satisfy the following defect constraints at all interior $i \in [1,3,\ldots,\mathrm{N}-1]$,

$$\begin{aligned} \zeta^i &:= \dot{x}^i - \frac{3N}{2T}\left(x^{i+1} - x^{i-1}\right) + \frac{1}{4}\left(\dot{x}^{i-1} + \dot{x}^{i+1}\right) = 0 \\ \delta^i &:= x^i - \frac{1}{2}\left(x^{i+1} + x^{i-1}\right) - \frac{T}{8N}\left(\dot{x}^{i-1} - \dot{x}^{i+1}\right) = 0 \end{aligned}, \quad (38)$$

then they are accurate approximations of the given continuous dynamics. (38) defines the modified Harmite-Simpson conditions for the direct collocation trajectory optimization [30].

Based on the above formulation, now we can construct a constrained nonlinear programming problem to solve for the virtual constraint excited trajectory optimization of the articulated truck model. To incorporate the virtual constraints based feedback control with the trajectory optimization, we enforce the output dynamics equation given in (36) at each node. Then the control input u^i will be implicitly determined via this constraint without explicitly enforcing it as in (35). Further, the output z and its derivative $\dot{z}$ should equal to the desired trajectory $h_d(t)$ at $t = 0$ to ensure that the system lies on the zero dynamics manifold $\forall t \in [0,t]$. Let $\mathcal{J}(\cdot)$ be the cost function to be minimized, the trajectory optimization problem can be stated as:

$$\begin{aligned} &\arg\min\ \mathcal{J}(\cdot)\quad s.t. \\ &\zeta^i = 0, \delta^i = 0 \\ &\dot{x}^i = Ax^i + Bu^i + Ed, \\ &\ddot{z}^i - \ddot{h}_d(t_i) + K_p\left(z^i - h_d(t_i)\right) + K_d\left(\dot{z} - \dot{h}_d(t_i)\right) = 0, \\ &z^0 - h_d(t_0) = 0, \\ &\dot{z}^0 - \dot{h}_d(t_0) = 0, \\ &-u_{\max} \le u^i \le u_{\max} \\ &\dot{b}\left(x^i, \dot{x}^i\right) + \kappa\frac{e^{b(x^i)} - 1}{e^{b(x^i)} + 1} \ge 0 \\ &V\left(x^N - \eta_x(d)\right) \le c_1 V\left(x^0 - \eta_x(d)\right) \\ &\left\| x_2(T) - \gamma\left(x_1(T)\right)\right\| \le c_3 \end{aligned}, \quad (39)$$

where $z^i = z\left(x^i\right)$, $\dot{z}^i = \dot{z}\left(x^i\right)$, and $\ddot{z}^i = \ddot{z}\left(x^i, \dot{x}^i\right)$, respectively. The first 3 lines of constraints correspond to the colocation constraint; 4th and 5th line correspond to the virtual constraint; the 6th line is the input constraint; the 7th line is the CBF constraint, 8th line is the Lyapunov constraint and the line is the insertion map constraint. $c_3 > 0$ is a small constant that makes x_2 close to $\gamma(x_1)$ at the end of the trajectory. We use a cost-to-go function from a particular LQR design as the Lyapunov function:

$$V(x) = x^T P x, \quad (40)$$

where P is the cost to go matrix calculated from Riccati equation. $\eta_x(d)$ represents the equilibrium point under d, as defined in **A-2**. Again, we assume $F_y = 0$, and $\eta_x(d)$ depends only on r_d.

The cost function in (39) is a weighted sum of multiple cost functions, consisting of the following terms:

- Final value cost $V\left(x(T) - \eta_x(d)\right)$, where the Lyapunov function is the same as that in the constraint.
- $\int z^2 dt$, the square integral of z
- $\int \ddot{z}^2 dt$, the square integral of jerk
- $\|y\|_\infty$, maximum deviation from road center
- $\|r\|_\infty$, maximum yaw rate
- $\int u^2 dt$, square integral of input
- $|\alpha_N|$, penalty on the last Bezier coefficient(facilitate convergence of the Bezier curve)

The terms that consist of function integrals are approximately computed using the Simpson's quadrature rule [31].

Remark 7: The CBF condition is modified based on (5). Since $\frac{e^b - 1}{e^b + 1}$ 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 $b(x) = 0$, $\frac{e^b - 1}{e^b + 1} = 0$, which resembles the original CBF condition in (7).

Since $\frac{e^b-1}{e^b+1}$ is still an extended class $\mathcal{K}$ function, by Proposition 1 in [32], $\{x \mid b(x) \geq 0\}$ is still invariant under the modified constraint, and **Proposition 2** still holds.

The setting of the constraints and costs seem complicated, they are 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.

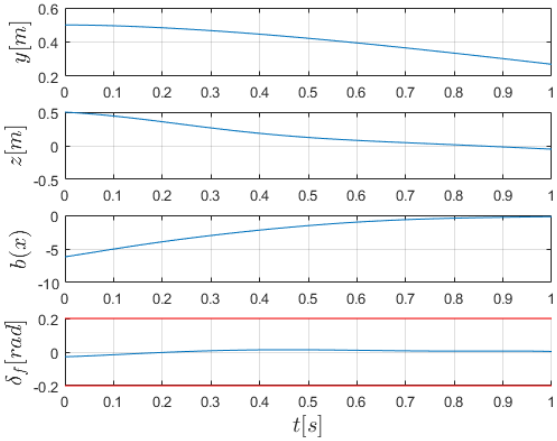


Fig. 7. Example of trajectory optimization result

Fig. 7 shows an example trajectory with initial lateral deviation $y_0 = 0.5m$ and road yaw rate $r_d = 0.02rad/s$. 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.

C. Trajectories used in CH framework

Now consider a CH controller with trajectories generated with the procedure described in this section. Recall **Proposition 2**, **A-1**, **A-2** are trivially satisfied by the linear dynamics and the LQR cost-to-go function satisfies **A-3**. The CBF condition and Lyapunov condition in **A-6** are enforced in the optimization, as shown in (39). Pick $x_1 = \begin{bmatrix} z & \dot{z} & \psi & -B_4 v_y + B_2 r & \psi_a \end{bmatrix}$, which is the first 5 entries of χ, and $x_2 = \chi_{6:8}$. Since z and $\dot{z}$ are part of x_1, dynamic 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 **A-7**. By **Proposition 2**, the closed loop system with CH feedback stays within $\{x \mid b(x) \geq 0\}$, and converges to $\eta_x(d)$ exponentially once r_d stops changing.

D. Generating the training set

The good convergence and set invariance property guaranteed by **Proposition 2** is true if we have such trajectory built for every initial condition, but this is impossible for continuous state space. We instead use supervised learning to train the mapping from initial conditions to desired trajectories with a finite trajectory library, which is generated by the above mentioned trajectory optimization process.

By varying the initial conditions and generating the corresponding trajectories with direct collocation, we hope to 'teach' the neural network to generate good trajectories for various initial conditions. The input to the neural network is called features, denoted as Φ, 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.

$$\pi : \Phi \to \mathcal{A}. \tag{41}$$

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 of the grid points. Following the discussion in Section V.A, the features picked for the training set are

$$\Phi = \begin{bmatrix} y, \psi, r, \psi_a, v_y, r_d \end{bmatrix}, \tag{42}$$

which is a linear transformation of x_1 plus r_d.

Since most driving behavior is mild, yet the controller should be able to handle bad initial conditions, we generate two training sets, denoted as S_1 and S_2. S_1 consists of trajectories that last for 1 second, and the features of the trajectories have a wider span; S_2 consists of trajectories that last for 3 seconds, and the features are 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 II.

TABLE II TRAINING SET PARAMETER SETTING

Feature	S_1	S_2
y range	$[-0.5, 0.5]$[m]	$[-0.3, 0.3]$[m]
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	$[-0.04, 0.04][rad]$	$[-0.04, 0.04][rad]$

In total, there are 62825 trajectories in S_1, and 29300 trajectories in S_2.

The neural network has 6 hidden layers and ReLU is used as the rectifier. The training is done using Tensorflow [33] and the result of the training is shown in TABLE III.

TABLE III TRAINING RESULT

	S_1	S_2
Correlation(R)	98.1%	99.9%
MSE	0.016	0.00021

VI. IMPLEMENTATION OF LEARNING BASED CONTROLLER

In this section, we present the implementation of the learned CH controller. The CH controller use the mapping trained by supervised learning to generate a desired trajectory for the output z based on the current state and r_d, then track the desired trajectory with the control law in (35). The desired trajectory is updated when certain triggering events happen. Then CBF acts as a supervisory controller following the structure in Fig. 1.

A. *Triggering events for CH trajectory updates*

There are three circumstances under which the CH trajectory will be updated.

- The Bezier curve is executed to the end
- Significant change in r_d
- The trajectory tracking error is too big

In the first case, since the trajectory optimization has a limited horizon (1s or 3s), the neural network will use the current feature to generate a new vector of Bezier coefficients, and the controller will continue with the new Bezier curve.

In the second case, if r_d differs much from the r_d used to generate the current Bezier curve, the Bezier coefficient should be updated based on the current r_d, since r_d is assumed to be constant during the whole horizon of the trajectory. The rest of the features are simply initial conditions, so their change do not trigger update of the Bezier coefficients.

In the third case, when the trajectory deviates too far from the desired trajectory, it calls for a replan. This is likely to be caused by some unexpected disturbance.

When switching from one trajectory to the succeeding trajectory, it is necessary to perform smoothing to make sure that h_d is sufficiently smooth. Since the input has relative degree 2, making $h_d \in C^2$ will make the input continuous. The smoothing for Bezier curve is very simple. For an N^{th} order Bezier curve, the value for 0th to 2nd derivative at $s=0$ are

$$\begin{aligned} \mathbf{B}(0) &= \alpha_0 \\ \mathbf{B}'(0) &= N\alpha_1 - N\alpha_0 \\ \mathbf{B}''(0) &= N(N-1)(\alpha_2 + \alpha_0 - 2\alpha_1) \end{aligned} \quad . \tag{43}$$

Solving for α_0, α_1 and α_2:

$$\begin{aligned} \alpha_0 &= h_d \\ \alpha_1 &= \frac{\dot{h}_d}{N} + \alpha_0 \\ \alpha_2 &= \frac{\ddot{h}_d}{N(N-1)} + 2\alpha_1 - \alpha_0 \end{aligned} \quad , \tag{44}$$

where h_d, $\dot{h}_d$ and $\ddot{h}_d$ are the current value and derivatives of h_d. 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.

B. *CBF as supervisory controller*

Even though the CBF condition is enforced in the trajectory optimization for the training set, after supervised learning, there is no guarantee that the trajectory generated by the neural network will always satisfy the CBF condition. Therefore CBF is still implemented as supervisory controller on top of the learning based controller. The CBF solves the following optimization:

$$\begin{aligned} &\min_{\Delta u} \; w_1 \|\Delta u\|^2 + w_2 \|\Delta u - \Delta u_{old}\|^2 \\ &s.t.\; \mathcal{C}(x, d, u_0 + \Delta u) \geq 0,\; u_0 + \Delta u \in \mathcal{U} \end{aligned} \quad , \tag{45}$$

where Δu is the intervention of the CBF, Δu_{old} is the intervention of the previous time instant, $\mathcal{C}(\cdot)$ is the CBF condition and w_1, w_2 are weights. The reason for the second penalty term is to prevent chattering if intervention is necessary. The CBF condition is defined as

$$\begin{cases} \dot{b} + \gamma b \geq 0, & if\ b(x) \geq 0 \\ \dot{b} + \gamma \dfrac{e^b - 1}{e^b + 1} \geq 0, & if\ b(x) \leq 0 \end{cases} \quad , \tag{46}$$

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

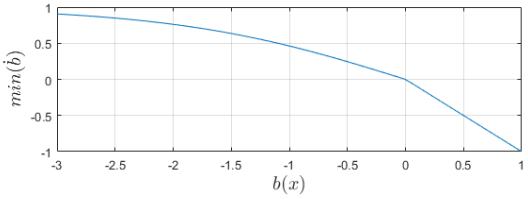


Fig. 8. lower bound for $\dot{b}$

Remark 8: When $b(x) \geq 0$, the existence of Δu is guaranteed by construction of the CBF; when $b(x) \leq 0$, there is no guarantee of feasibility. When (45) is infeasible, the input is saturated by $\mathcal{U}$.

The control structure is shown in Fig. 9. x_d is the desired trajectory generated via the mapping trained by supervised learning, and the student controller can be simplified to a trajectory tracking controller given the desire trajectory. The dashed arrow indicates that the trajectory generation block takes the CBF condition into account.

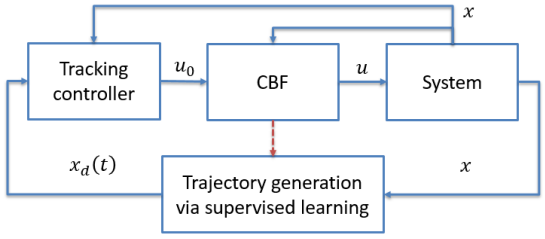


Fig. 9. Structure of the proposed supervisory control

VII. SIMULATION RESULT

We validate our control design on Trucksim, a physics based simulation software that has high fidelity. The model picked for simulation has 312 states and is a tractor semitrailer with heavy cargo in the trailer, weighing 35 tons in total, as shown in Fig. 10.

Fig. 10. Animation with Trucksim

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

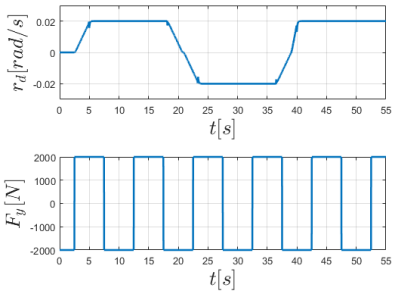


Fig. 11. Disturbance to the system

As shown in Fig. 11, the road profile consists of segments with different curvatures， and the side-wind is a square wave with maximum allowed magnitude.

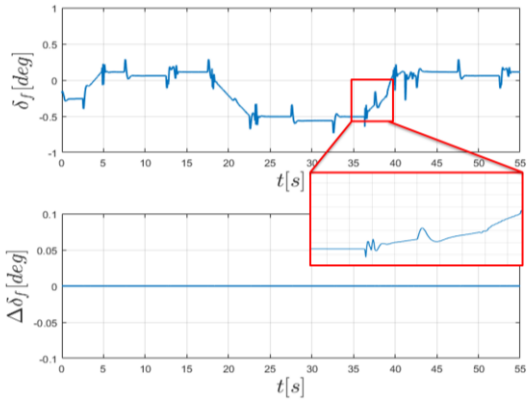


Fig. 12. Input to the truck

The steering input trajectory is shown in Fig. 12. The CBF never intervened during the simulation. We zoom in the input to show a 5 second period of input. The input is actually pretty smooth. The little bumps are necessary to counter the side-wind when it changes direction.

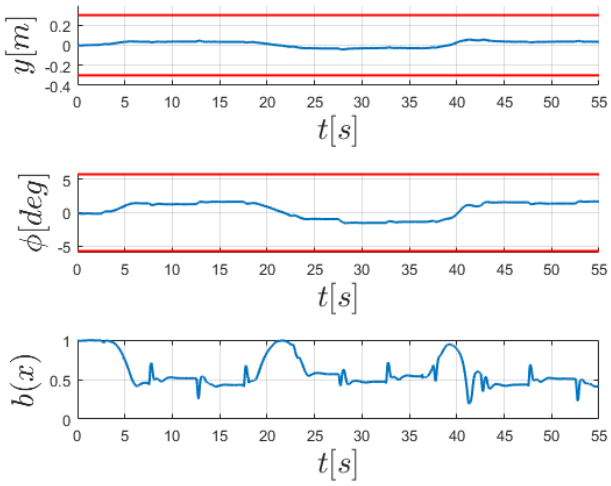


Fig. 13. Value of CBF and key states during simulation

Fig. 13 shows the value of CBF and two key states. Lateral deviation y and roll angle ϕ never exceed the limit (plotted as red line), and $b(x)$ is always positive, showing that the CBF bound was never breached. More importantly, the CH controller never cause CBF to intervene, which is due to the supervised learning that inherits the good properties of the training set.

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

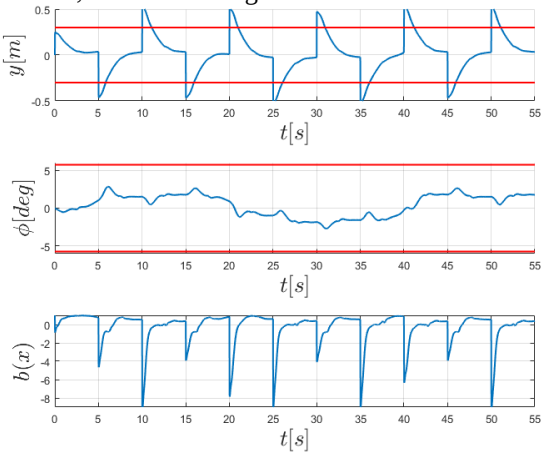


Fig. 14. Simulation with big initial deviation

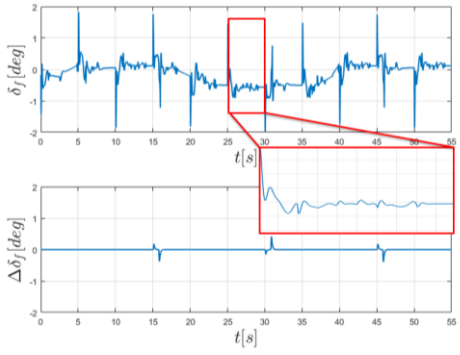


Fig. 15. Input with big initial deviation

Fig. 15 shows the input under big deviation. The CBF intervention is very mild compared to the size of u_0, and the learning based controller is able to handle the bad initial condition most of the time.

As a comparison, we tune a LQR controller with feedforward control of r_d, and it performs very well under normal driving conditions. However, when the initial condition is bad (under the same setting as Fig. 14 shows), the LQR controller triggers intervention from the CBF many times and the jerk is severe, as shown in Fig. 16.

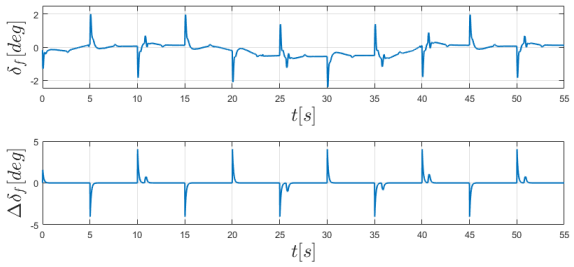


Fig. 16. Simulation result with LQR

Though the LQR controller is fine-tuned, it triggers severe intervention from the CBF frequently. On the other hand, we observe none or very mild intervention from the CBF in all trial simulations when the states are within the training set span under the learning based controller. When the states are outside the span of the training set, the neural network is asked to extrapolate instead of interpolate, therefore the performance is not guaranteed.

VIII. Conclusion and Discussion

We propose a supervised learning based approach to combine CBF with learning, and construct controllers with smooth performance and safety guarantee. The idea is to use trajectory optimization technique to generate training set consisting of trajectories that satisfy the CBF constraint, then use supervised learning to learn the mapping from the initial condition to a parameterized desired trajectory. The policy generated with supervised learning will inherit the good properties of the training set, though not rigorously. On top of that, safety guarantee is formally imposed with CBF as supervisory controller. The simulation shows that the proposed approach is able to reduce the intervention of the CBF and therefore improve the performance while guaranteeing safety.

We choose to let the supervised learning learn the mapping from initial condition to the desired output trajectory, instead of the mapping from the initial condition to the desired input trajectory. The stability is guaranteed under the CH framework, whereas there is no guarantee that following the desired input trajectory will lead to desired performance.

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 is poor. In other words, to get good performance in a wide range, one need to have training data with enough coverage. Second, when training data from a large range of features are stacked together for the training, the regression accuracy drops and the performance is bad. To solve this, one might need more complicated NN structure, or use multiple NN for different situations.

Acknowledgement

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).

References

[1] 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*, ed: IEEE, 2014, pp. 6271-6278.

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

[3] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *American Control Conference (ACC), 2015*, 2015, pp. 4542-4548.

[4] Y. Chen, H. Peng, and J. Grizzle, "Obstacle Avoidance for Low-Speed Autonomous Vehicles With Barrier Function," *IEEE Transactions on Control Systems Technology*, 2017.

[5] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to automotive safety systems," *arXiv preprint arXiv:1609.06408*, 2016.

[6] C. Yuxiao, H. Peng, and J. Grizzle, "Obstacle Avoidance for Low Speed Autonomous Vehicles," 2016.

[7] H. Gomi and M. Kawato, "Neural network control for a closed-loop system using feedback-error-learning," *Neural Networks*, vol. 6, pp. 933-946, 1993.

[8] X. Da, R. Hartley, and J. W. Grizzle, "Supervised Learning for Stabilizing Underactuated Bipedal Robot Locomotion, with Outdoor Experiments on the Wave Field."

[9] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

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

[11] S.-Y. Oh, J.-H. Lee, and D.-H. Choi, "A new reinforcement learning vehicle control architecture for vision-based road following," *IEEE Transactions on Vehicular Technology*, vol. 49, pp. 997-1005, 2000.

[12] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control* vol. 1: Athena Scientific Belmont, MA, 1995.

[13] J. H. Gillula and C. J. Tomlin, "Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 2723-2730.

[14] N. Smit-Anseeuw, R. Vasudevan, and C. D. Remy, "Safe Online Learning Using Barrier Functions."

[15] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of Control Barrier Functions for Safety Critical Control** This work is partially supported by the National Science Foundation Grants 1239055, 1239037 and 1239085," *IFAC-PapersOnLine*, vol. 48, pp. 54-61, 2015.

[16] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, pp. 293-320, 2003.

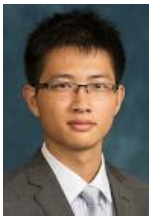
[17] J. Bochnak, M. Coste, and M.-F. Roy, *Real algebraic geometry* vol. 36: Springer Science & Business Media, 2013.

[18] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, pp. 293-320, May 2003.

[19] G. Stengle, "A Nullstellensatz and a Positivstellensatz in semialgebraic geometry," *Mathematische Annalen*, vol. 207, pp. 87-97, 1974.

[20] A. Papachristodoulou and S. Prajna, "On the construction of Lyapunov functions using the sum of squares decomposition," in

Decision and Control, 2002, Proceedings of the 41st IEEE Conference on, 2002, pp. 3482-3487.
[21] S. Prajna and A. Jadbabaie, "Safety Verification of Hybrid Systems Using Barrier Certificates," in *International Workshop on Hybrid Systems: Computation and Control*, Berlin Heidelberg, 2004, pp. 477-492.
[22] S. Prajna, A. Papachristodoulou, and F. Wu, "Nonlinear control synthesis by sum of squares optimization: A Lyapunov-based approach," in *Control Conference, 2004. 5th Asian*, 2004, pp. 157-165.
[23] K. C. Goh, L. Turan, M. G. Safonov, G. P. Papavassilopoulos, and J. H. Ly, "Biaffine matrix inequality properties and computational methods," in *American Control Conference, 1994*, 1994, pp. 850-855.
[24] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-Time Motion Planning for Agile Autonomous Vehicles," in *Journal of Guidance, Control, and Dynamics* vol. 25, ed, 2002, pp. 116-129.
[25] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, pp. 1680-1685, 2007.
[26] X. Da and J. Grizzle, "Combining Trajectory Optimization, Supervised Machine Learning, and Model Structure for Mitigating the Curse of Dimensionality in the Control of Bipedal Robots," *arXiv preprint arXiv:1711.02223*, 2017.
[27] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, "Feedback control of dynamic bipedal robot locomotion," ed: CRC press, 2007.
[28] A. Shiriaev, J. W. Perram, and C. Canudas-de-Wit, "Constructive tool for orbital stabilization of underactuated nonlinear systems: Virtual constraints approach," *IEEE Transactions on Automatic Control*, vol. 50, pp. 1164-1176, 2005.
[29] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*: SIAM, 2010.
[30] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 2016, pp. 1447-1454.
[31] J. Stoer and R. Bulirsch, *Introduction to numerical analysis* vol. 12: Springer Science & Business Media, 2013.
[32] 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, pp. 3861-3876, 2017.
[33] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.



Yuxiao Chen is a Ph.D. candidate at the University of Michigan. He received his Bachelors degree in Mechanical Engineering from Tsinghua University, Beijing, China in 2013. His research interests are control theory, cyber physical system and particularly the application in vehicle control.

Ayonag Hereid received the Ph.D. in Mechanical Engineering from the Georgia Institute of Technology in 2016. He is currently a postdoctoral research fellow in the EECS department at the University of Michigan, Ann Arbor. His research interests lie at the intersection of nonlinear control and optimization theory, with a particular focus on developing elegant and principled control solutions for complex robotic systems, including bipedal robots and exoskeletons. He was the recipient of the Best Student Paper Award in 2014 from the ACM International Conference on Hybrid System: Computation and Control and was nominated as the Best Conference Paper Award Finalists in 2016 at the IEEE International Conference on Robotics and Automation.

Huei Peng received his Ph.D. in Mechanical Engineering from the University of California, Berkeley in 1992. He is now a Professor at the Department of Mechanical Engineering at the University of Michigan. His research interests include adaptive control and optimal control, with emphasis on their applications to vehicular and transportation systems. His current research focuses include design and control of electrified vehicles, and connected/automated vehicles. Huei Peng has been an active member of the Society of Automotive Engineers (SAE) and the American Society of Mechanical Engineers. He is both an SAE fellow and an ASME Fellow. He received the National Science Foundation (NSF) Career award in 1998.

Jessy W. Grizzle received the Ph.D. in electrical engineering from The University of Texas at Austin in 1983. He is currently a Professor of Electrical Engineering and Computer Science at the University of Michigan, where he holds the titles of the Elmer Gilbert Distinguished University Professor and the Jerry and Carol Levin Professor of Engineering. He jointly holds sixteen patents dealing with emissions reduction in passenger vehicles through improved control system design. Professor Grizzle is a Fellow of the IEEE and IFAC. He received the Paper of the Year Award from the IEEE Vehicular Technology Society in 1993, the George S. Axelby Award in 2002, the Control Systems Technology Award in 2003, the Bode Prize in 2012 and the IEEE Transactions on Control Systems Technology Outstanding Paper Award in 2014. His work on bipedal locomotion has been the object of numerous plenary lectures and has been featured on CNN, ESPN, Discovery Channel, The Economist, Wired Magazine, Discover Magazine, Scientific American and Popular Mechanics.