Safe Online Learning Using Barrier Functions

Nils Smit-Anseeuw, Ram Vasudevan, and C. David Remy

***Abstract*— We present a method for guaranteeing the safety of online learning schemes. The method uses barrier certificates and Sums-of-Squares programming to find a safe region of state space and a controller which renders that space positively invariant. This safe set and controller are then used to create "training wheels", which can be added to any controller to create a guaranteed safe controller. These training wheels alter the given controller only when the state approaches the edge of the guaranteed safe region in state space. Thus, except for where it would otherwise lead to falling, the original controller remains unchanged. For a given learning scheme, this projection is performed for each controller rollout to generate a safety guarantee for the scheme with minimal interference. We present simulation results for a simple car model and a simple hopping model, and plan to demonstrate safe learning of a walking controller for a compass gait walker.**

I. Introduction

Online learning is a valuable tool for achieving high performance behaviour in physical systems when modelling accuracy is limited. This is particularly true for legged robots since it is inherently difficult to accurately model contact events. However, such learning schemes can be particularly challenging for legged robots due to the high cost of falling (which can require lengthy hardware repairs). As such, successful learning implementations on walking robots have been largely limited to hardware in which either the likelihood or the cost of falling is low [1], [2].

One way to mitigate the risk of falling is to determine the space of "safe" controllers and states. That is, the space of initial conditions and control inputs which avoid failure states for all time. Once found, we can restrict a given learning scheme to search for controllers within this space.

In this work, we present a method to compute the set of safe states using polynomial barrier functions. We use this set to compute a mask that can take any unsafe controller and render it safe with minimal modification. This approach is similar to that in [3], in which barrier functions are used to guarantee the safety of Lyapunov-based controllers.

At this time, we have implemented the method on a simple car model and a simple hopper model. By the time of the conference, we intend to use this approach to safely learn a hopping controller on hardware for the robot RAM*one*[4].

II. Methods

The class of systems we consider in this paper are those with dynamics of the form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g_u}(\mathbf{x})\mathbf{u} + \mathbf{g_d}(\mathbf{x})\mathbf{d}. \tag{1}$$

Where $\mathbf{f}$, $\mathbf{g_u}$ and $\mathbf{g_d}$ are polynomials in $\mathbf{x}$, $\mathbf{u} \in U$ is the control input, which takes values from the bounded set U, and $\mathbf{d} \in D$ is the uncontrolled, time varying disturbance input which takes values from the bounded set D.

A. Safe Set Computation

We begin by defining the set of safe states:

$$X_s = \{\mathbf{x_0} \,|\, \exists \mathbf{u}(\mathbf{x}) \in U \text{ s.t. } \mathbf{x}(\mathbf{x_0}, \mathbf{u}(\mathbf{x}), \mathbf{d}(t), \tau) \notin X_F \quad \forall \tau \in [0, \infty), \forall \mathbf{d}(t) \in D\}. \tag{2}$$

Where X_f are the failure states, and the notation $\mathbf{x}(\mathbf{x_0}, \mathbf{u}(\mathbf{x}), \mathbf{d}(t), \tau)$ represents the flow forward of the state $\mathbf{x_0}$ under state feedback controller $\mathbf{u}(\mathbf{x})$ and disturbance signal $\mathbf{d}(t)$ after time τ. This can be seen as the largest forward control invariant set [3] that doesn't include the failure states.

To find X_s, we use a barrier function approach similar to that in [5], to define the following optimization problem:

$$\begin{aligned} &\max_{v, \mathbf{u_s}} \int v(x) \\ &\text{s.t. } v(\mathbf{x}) \leq 1 \quad \forall \mathbf{x} \in X, \qquad v(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in X_f \\ &\qquad \dot{v}(\mathbf{x}, \mathbf{u_s}(\mathbf{x}), \mathbf{d}) > 0 \quad \forall \mathbf{x} \in \{\mathbf{x} | v(\mathbf{x}) = 0\}, \forall \mathbf{d} \in D \\ &\qquad -1 \leq \mathbf{u_s}(\mathbf{x}) \leq 1 \qquad \forall \mathbf{x} \in X \end{aligned} \tag{3}$$

This is an infinite dimensional bilinear program that can be approximated as a bilinear matrix inequality using polynomial basis functions and Sums-of-Squares. To solve this problem, we use an alternation approach similar to [5].

Once this problem is solved, an inner approximation of the safe set is given by $X_s = \{\mathbf{x} | \mathbf{v}(\mathbf{x}) \geq 0\}$, and a controller that keeps states within this set is given by $\mathbf{u_s}(\mathbf{x})$.

B. Control Masking

Note that the only requirement for forward invariance of the set X_s is that the flow of the system is inward on the boundary of X_s. This means that any controller will be safe so long as it enforces this flow condition on the edge of the safe set. Thus we define a safety mask that modifies controllers only in the neighborhood of $v(\mathbf{x}) = 0$.

Since a controller that is discontinuous on the boundary of the safe set would pose difficulties for systems with finite bandwidth, we additionally must ensure that the new controller is continuous near the boundary.

To define the mask, we smoothly interpolate between the initial controller and the controller $\mathbf{u_s^{(i)}}(\mathbf{x})$, which we know satisfies the safety condition at the boundary (see Fig. 1).

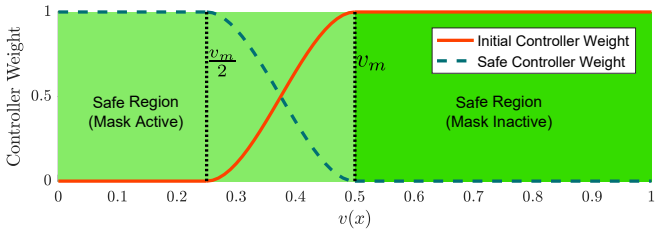
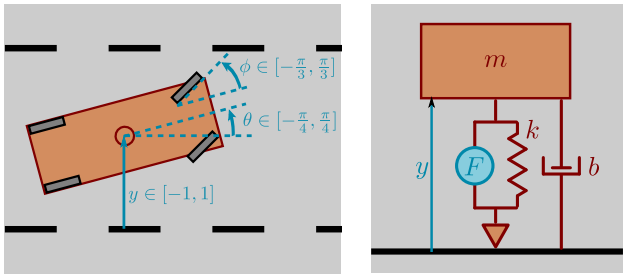


Fig. 1: Scaling weights of the unmasked controller $\mathbf{u_0}$ and the "training wheels" controller $\mathbf{u_s}$. The weights satisfy $w_0 + w_s = 1$ and are used to form the masked controller $\mathbf{u_m} = w_0\mathbf{u_0} + w_s\mathbf{u_s}$.



(a) Dubin's car. (b) 1d hopper.

Fig. 2: Simple models for demonstrating control masking. The Dubin's car has state $[y, \theta]^T$ and steering angle input ϕ. The hopper is a hybrid model with state $[y, \dot{y}]^T$, force input $F > 0$, and a guard at $\{y = 0.5\,\mathrm{m}, \dot{y} > 0\,\mathrm{m/s}\}$ (foot liftoff). At this guard, the velocity is reflected about $\dot{y} = 0\,\mathrm{m/s}$.

C. Safe Online Learning

Once this safety mask has been computed, the result can be applied in real-time to arbitrary control input. Simply monitor the state until it approaches the boundary of X_s, then follow the interpolation scheme to determine the safe input. As such, this approach can be used as a last step in any learning scheme to ensure that any controller used on the hardware is safe.

To ensure that safety is maintained despite modelling errors, the disturbance bounds must be large enough to capture all observed differences between model and hardware.

III. Preliminary Results

At the time of this abstract, we have successfully implemented this method on a Dubin's car, and a hybrid extension of this method on a 1-dimensional hopper (see Fig. 2).

The safe set and the result of the control masking are shown for the Dubin's car in Fig. 3 and for the hopper in figure Fig. 4.

IV. Future Work

The next goal of this project is to increase the complexity of the example models. To this end, we are working on implementations for a 5 dimensional car model and for a compass gait walker.

Additionally, we are interested in demonstrating the method on hardware using a small remote control car.

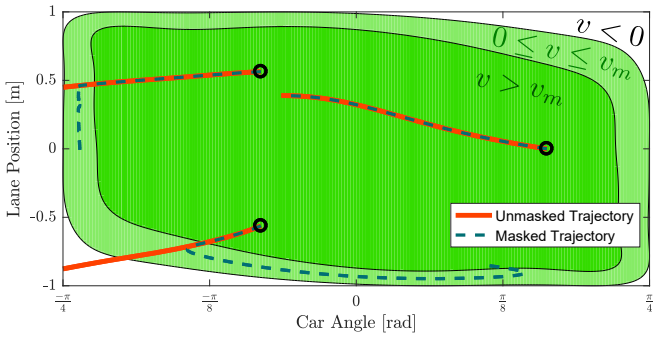


Fig. 3: Control masking for the Dubin's car model. Shown in solid red are trajectories generated by a randomly generated controller. In dashed blue, we show the trajectories from the masked controller. In the dark green region (where $v(x) > v_m$), the original controller is unaltered, so the two trajectories coincide. In the light green region ($0 \le v(x) \le v_m$), we modify the original controller to keep the masked trajectories from failing. The green regions together represent the set of safe states X_s.

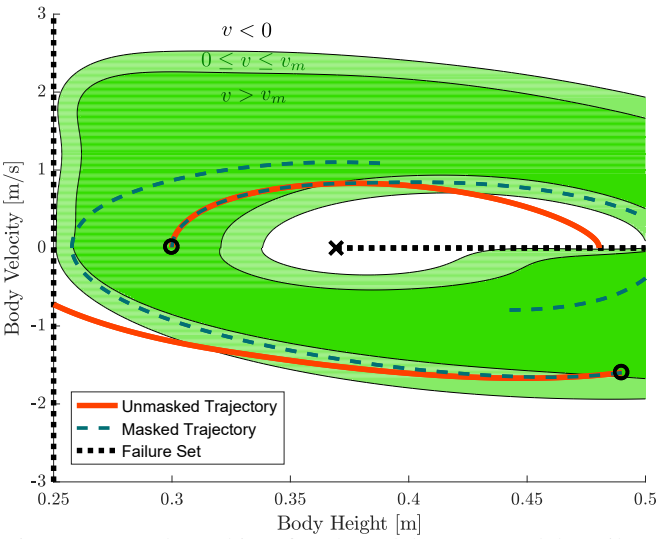


Fig. 4: Control masking for the 1d hopper model. Failure occurs when the body height goes below $0.25\,\mathrm{m}$ or when the hopper fails to leave the ground.

References

[1] R. Tedrake, T. W. Zhang, M.-f. Fong, and H. S. Seung, "Actuating a simple 3d passive dynamic walker," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5, pp. 4656–4661, IEEE, 2004.

[2] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 3, pp. 2619–2624, IEEE, 2004.

[3] 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*, 2016.

[4] N. Smit-Anseeuw, R. Gleason, R. Vasudevan, and C. D. Remy, "The energetic benefit of robotic gait selection: A case study on the robot ramone," *IEEE Robotics and Automation Letters*, 2017.

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