ShieldNN: A Provably Safe NN Filter for Unsafe NN Controllers

James Ferlez[*†], **Mahmoud Elnaggar**[* * †], **Yasser Shoukry**[*], **and Cody Fleming**[* * *]
[*]Electrical Engineering and Computer Science, University of California, Irvine
[**]Electrical and Computer Engineering, University of Virginia
[* * *]Engineering Systems and Environment, University of Virginia [*†]

Abstract

In this paper, we consider the problem of creating a safe-by-design Rectified Linear Unit (ReLU) Neural Network (NN), which, when composed with an arbitrary control NN, makes the *composition* provably safe. In particular, we propose an algorithm to synthesize such NN filters that safely correct control inputs generated for the continuous-time Kinematic Bicycle Model (KBM). ShieldNN contains two main novel contributions: first, it is based on a novel Barrier Function (BF) for the KBM model; and second, it is itself a provably sound algorithm that leverages this BF to a design a safety filter NN with safety guarantees. Moreover, since the KBM is known to well approximate the dynamics of four-wheeled vehicles, we show the efficacy of ShieldNN filters in CARLA simulations of four-wheeled vehicles. In particular, we examined the effect of ShieldNN filters on Deep Reinforcement Learning trained controllers in the presence of individual pedestrian obstacles. The safety properties of ShieldNN were borne out in our experiments: the ShieldNN filter reduced the number of obstacle collisions by 99.4%-100%. Furthermore, we also studied the effect of incorporating ShieldNN during training: for a constant number of episodes, 28% less reward was observed when ShieldNN wasnt used during training. This suggests that ShieldNN has the further property of improving sample efficiency during RL training.

1 Introduction

In this paper, we will consider the safety of data-trained memoryless feedback controllers[3] in the following context: we will assume an autonomous vehicle that is described by the Kinematic Bicycle Model (KBM) dynamics (a good approximation for four-wheeled vehicles [1]), and a safety property that the autonomous vehicle should avoid a stationary, fixed-radius disk in the plane. In particular, we propose ShieldNN, an algorithm to design Rectified Linear Unit (ReLU) safety networks for this scenario: a ShieldNN network composed in series with *any* memoryless feedback controller[1] makes the composition of the two controllers provably safe by the aforementioned criterion. This structure itself distinguishes ShieldNN from most other work on safe data-trained controllers: instead of designing a single safe controller, ShieldNN uses the KBM dynamics to design a *controller-agnostic* NN that corrects – in real-time – unsafe control actions generated by *any* controller. In other words, ShieldNN designs a "safety-filter" NN that takes as input the instantaneous control action generated by a controller (along with the state of the system) and outputs a safe control for the KBM dynamics; this safety-filter NN thus replaces *unsafe* controls generated by the original controller with safe

[*†] Equally contributing first authors

[†]This work was partially sponsored by the NSF awards #CNS-2002405, #CNS-2013824 and #CPS-1739333.

[3]RL-trained feed-forward neural networks, for example.

Preprint. Under review.

controls, whereas *safe* controls generated by the original controller are passed through unaltered – i.e., unsafe controls are "filtered" out. A block diagram of this structure is depicted in Fig. 1.

The benefits of this approach are manifest, especially for data-trained controllers. On the one hand, existing controllers that have been designed *without* safety in mind can be made safe by merely incorporating the safety filter in the control loop. In this scenario, the safety filter can also be seen as a countervailing factor to controllers trained to mimic experts: the expert learning can be seen as a design for "performance", and the safety filter is added to correct unanticipated unsafe control behavior as needed. On the other hand, the controller-agnostic nature of our proposed filter means that ShieldNN itself may be incorporated into training. In this way, the safety filter can be seen to function as a kind of "safety expert" during training, and this can potentially improve sample efficiency by eliminating training runs that end in unsafe states.

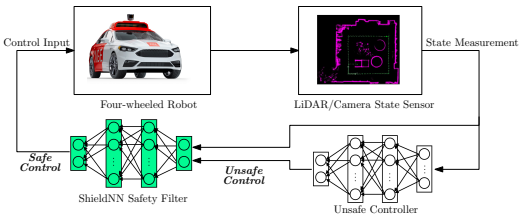


Figure 1: Block diagram of ShieldNN in the control loop for a four-wheeled autonomous vehicle.

The main theoretical contribution of this paper is thus the development of the ShieldNN algorithm. The central pillar of ShieldNN is the notion of a barrier function, because a barrier function allows the safety problem for a *feedback* controller to be recast as a set membership problem for the *outputs* of said controller. In particular, this recasting reduces the safety-filter design problem into one of designing a *prediction-style* NN whose outputs are constrained to lie in a specific set. As a prerequisite for ShieldNN then, we propose a novel class of candidate barrier functions for the KBM dynamics that is characterized by three real-valued parameters (one of which is the safety radius). The necessity for a *class* of candidate barrier functions stems from the difficulty in analytically designing a barrier function for the KBM dynamics. Thus, ShieldNN is functionally divided into two parts: a verifier, which soundly verifies a particular choice of barrier function (from the aforementioned class), and a synthesizer, which designs the actual ShieldNN filter.

Furthermore, we also validated these theoretical results with experimental validation on four-wheeled vehicle models. In particular, we apply ShieldNN safety-filters both before and after RL training for an autonomous vehicle simulated in CARLA [2]. Our results show that incorporating ShieldNN dramatically improved the safety of the vehicle: it reduced the number of obstacle collisions by 99.4%-100% in our safety experiments. We also studied the effect of incorporating ShieldNN during training: for a constant number of episodes, 28% less reward was observed when ShieldNN wasnt used during training. This suggests that ShieldNN has the further property of improving sample efficiency during RL training.

Related Work. Motivated by the lack of safety guarantees in Deep RL, recent works in the literature of safe RL have focused on designing new RL algorithms that can take safety into account. The work in this area can be classified into three categories. The works in the first category focus on how to modify the training algorithm to take into account safety constraints. Representative examples of this work include reward-shaping [3], Bayesian and robust regression [4, 5, 6], and policy optimization with constraints [7, 8, 9, 10]. Unfortunately, such approaches do not provide provable guarantees on the safety of the trained controller. The second category of literature focuses on using ideas from control theory to augment the RL agent and provide safety guarantees. Examples of this literature include the use of Lyapunov methods [11, 12, 13], safe model predictive control [14], reachability analysis [15, 16, 17], barrier certificates [18, 19, 20, 21, 22, 23, 24, 18, 25], and online learning of uncertainties [26]. Unfortunately, such methods suffer from either being computationally expensive, specific to certain controller structures or training algorithms or require certain assumptions on the system model. Finally, the third category focus on applying formal verification techniques (e.g., model checking) to verify formal safety properties of pretrained RL agents. Representative examples of this category are the use of SMT-like solvers [27, 28, 29] and hybrid-system verification [30, 31, 32]. These techniques only assess the safety of a given RL agent instead of designing a safe agent.

2 Problem Statement

As described above, we consider the kinematic bicycle model (KBM) as the dynamical model for our autonomous car. However, the usual KBM is defined in terms of the absolute Cartesian position of the bicycle , which is inconsistent with the sensing modalities typically available to an autonomous robot. Thus, we instead describe the bicycle kinematics in terms of relative positional variables that are directly measurable via LiDAR or visual sensors. In particular, the dynamics for the distance to the obstacle, $\|\vec{r}\|$, and the angle of the bicycle with respect to the obstacle, ξ, comprise a system of ordinary differential equations. These quantities describe a state-space model that is given by:

$$\begin{pmatrix} \dot{r} \\ \dot{\xi} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v\cos(\xi-\beta) \\ -\frac{1}{r}v\sin(\xi-\beta) - \frac{v}{\ell_r}\sin(\beta) \\ a \end{pmatrix}; \quad \beta \triangleq \tan^{-1}\left(\frac{\ell_r}{\ell_f+\ell_r}\tan(\delta_f)\right) \tag{1}$$

where $r(t) \triangleq \|\vec{r}(t)\|$; a is the linear acceleration input; δ_f is the front-wheel steering angle input[4]; and $\psi + \xi = \tan^{-1}(y/x)$. For the sake of intuition, we note a few special cases: when $\xi = \pm\pi/2$, the bicycle is oriented tangentially to the obstacle, and when $\xi = \pi$ or 0, the bicycle is pointing directly at or away from the obstacle, respectively (see Fig. 2). β is an intermediate quantity, an *invertible function* of δ_f.

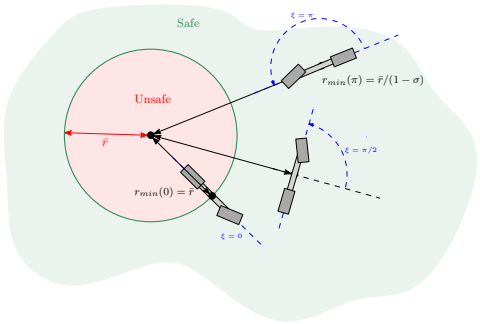


Figure 2: Obstacle specification and minimum barrier distance as a function of relative bicycle orientation, ξ.

We make the further assumption that the KBM has a control constraint on δ_f such that $\delta_f \in [-\delta_{f\,\max}, \delta_{f\,\max}]$. To simplify further notation, we will consider β directly as a control variable; this is without loss of generality, since there is a bijection between β and the actual steering control angle, δ_f. Thus β is also constrained: $\beta \in [-\beta_{\max}, \beta_{\max}]$. Finally, we define the state and control vectors for the KBM as: $\chi \triangleq (\xi, r, v)$ and $\omega \triangleq (a, \beta)$, where $\omega \in \Omega_{\text{admis.}} \triangleq \mathbb{R} \times [-\beta_{\max}, \beta_{\max}]$, the set of admissible controls.

Problem 1. *Consider a KBM robot with maximum steering angle $\delta_{f\,max}$, length parameters $l_f = l_r$ and maximum velocity v_{max}[5]. Consider also a disk-shaped region of radius $\bar{r}$ centered at the origin, $U = \{x \in \mathbb{R}^2 : \|x\| \leq \bar{r}\}$. Find a set of safe initial conditions, S_0, and a ReLU NN:*

$$\mathscr{NN} : (\chi, \omega) \mapsto \hat{\omega} \tag{2}$$

such that for any globally Lipschitz continuous controller $\mu : \chi \mapsto \omega \in \Omega_{admis.}$, the state feedback controller:

$$\mathscr{NN}(\chi, \mu(\chi)) : \chi \mapsto \hat{\omega} \tag{3}$$

*is **guaranteed** to prevent the robot from entering the unsafe region U if it was started from a state in S_0. Equivalently, applying feedback controller $\mathscr{NN}(\cdot, \mu(\cdot))$ ensures that $r > \bar{r}$ for all time when the initial condition is chosen in S_0.*

3 Approach

The most important feature of Problem 1 is that $\mathscr{NN}$ is a memoryless function that must correct the output of a feedback controller *instantaneously*. The existence of such a corrective function is not a priori guaranteed for the KBM dynamics. However, the well-known theory of Barrier Functions (BFs) provides a mechanism for ensuring the safety of a dynamical systems: in short, barrier functions are real-valued functions of the system state whose properties ensure that the value of the function remains greater than zero along trajectories of the system [33, 34]. Thus, if a barrier function is designed so that its zero super-level set is contained inside the set of safe states, then that subset is forward-invariant; i.e. if the system starts from a safe state, then it will stay safe for all future time. In this way, barrier functions can be used to to convert safety properties into an instantaneous – albeit state-dependent – set membership problem for control actions.

[4] That is the steering angle can be set instantaneously, and the dynamics of the steering rack can be ignored.

[5] In our KBM model, this technically requires a feedback controller on a, but this won't affect our results.

Thus, in the spirit of Problem 1, we employ the usual theory of autonomous barrier functions to control systems under *state-feedback* control: i.e. a control system $\dot{x} = f(x,u)$ in closed loop with a state-feedback controller $\pi : x \mapsto u$. In this scenario, a feedback controller in closed loop converts the control system into an autonomous one – the autonomous vector field $f(\cdot, \pi(\cdot))$. Moreover, the conditions for a barrier function can be translated into a set membership problem for the outputs of such a feedback controller. This is explained in the following corollary.

Corollary 1. *Let $\dot{x} = f(x,u)$ be a control system that is Lipschitz continuous in both of its arguments on a set $\mathcal{D} \times \Omega_{admis.}$; furthermore, let $h : \mathbb{R}^n \to \mathbb{R}$ with $\mathcal{C}_h \triangleq \{x \in \mathbb{R}^n | h(x) \geq 0\} \subseteq \mathcal{D}$, and let α be a class $\mathcal{K}$ function. If the set*

$$R_{h,\alpha}(x) \triangleq \{u \in \Omega_{admis.} | \nabla_x^T h(x) \cdot f(x,u) + \alpha(h(x)) \geq 0\} \tag{4}$$

is non-empty for each $x \in \mathcal{D}$, and a feedback controller $\pi : x \mapsto u$ satisfies

$$\pi(x) \in R_{h,\alpha}(x) \quad \forall x \in \mathcal{D} \tag{5}$$

then $\mathcal{C}_h$ is forward invariant for the closed-loop dynamics $f(\cdot, \pi(\cdot))$.

Proof. This follows directly from an application of zeroing barrier functions [35, Theorem 1]. □

Corollary 1 is the foundation of ShieldNN: the only difference is that instead of designing a single controller π, we will design a safe "combined" controller $\mathcal{NN}(\cdot, \mu(\cdot))$. In this usage, when a controller μ generates a control action, $\mu(x)$, that lies outside of the set $R(x)$, $\mathcal{NN}$ must map it to a control *within* the set $R(x)$.

Thus, Corollary 1 admits the following three-step framework for developing ShieldNN filters.

ShieldNN Framework:

(1) Design a Candidate Barrier Function. For a function, h, to be a barrier function for a specific safety property, its zero super-level set, $\mathcal{C}_h$, must be contained in the set of safe states.

(2) Verify the Existence of Safe Controls. *(ShieldNN Verifier)* Show that the set $R_{h,\alpha}(x)$ is non-empty for each state $x \in \mathcal{C}_h$. This establishes that a safe feedback controller may exist.

(3) Design a Safety Filter. *(ShieldNN Synthesizer)* If possible, design $\mathcal{NN}_0$ such that $\mathcal{NN}_0 : x \in \mathcal{C}_h \mapsto \hat{u} \in R(x)$; then obtain a safety filter as:

$$\mathcal{NN}(x,u) := \begin{cases} u & \text{if } u \in R(x) \\ \mathcal{NN}_0(x) & \text{if } u \notin R(x). \end{cases} \tag{6}$$

ShieldNN thus hinges on the design of a barrier function, and then the design of two **prediction-type** NN functions: $\mathcal{NN}_0$, which generates a **safe** control at each $x \in \mathcal{C}_h$; and $\mathcal{NN}$, which **overrides any unsafe control** for a state with the associated value of $\mathcal{NN}_0$.

4 Barrier Function(s) for the KBM Dynamics: the Basis of ShieldNN

It difficult to analytically derive a single barrier function as a function of a particular robot and safety radius for the KBM. Thus, we instead define a class of *candidate* barrier functions for a specific robot: this class is further parameterized by a unit-less scaling parameter and the safety radius, and it has the property that there are guaranteed parameter choices that actually result in a barrier function. However, since the analytically guaranteed parameter choices are impractically conservative, this need a ShieldNN verifier algorithm to establish whether a particular (user-supplied) choice of barrier function parameters does indeed constitute a barrier function.

In particular, we propose the following class of candidate barrier functions to certify control actions so that the bicycle doesn't get within $\bar{r}$ units of the origin (Problem 1):

$$h_{\bar{r},\sigma}(\chi) = h_{\bar{r},\sigma}(\xi, r, v) = \frac{\sigma \cos(\xi/2) + 1 - \sigma}{\bar{r}} - \frac{1}{r} \tag{7}$$

where $\sigma \in (0,1)$ is an additional parameter whose function we shall describe subsequently. First note that the equation $h_{\bar{r},\sigma}(\chi) = 0$ has a unique solution, $r_{\text{min}}(\xi)$ for each value of ξ:

$$r_{\text{min}}(\xi) = \bar{r}/(\sigma \cos(\xi/2) + 1 - \sigma), \tag{8}$$

so the smallest value of $r_{\min}$ is $r_{\min}(0) = \bar{r}$. Thus, the function $h_{\bar{r},\sigma}$ satisfies the requirements of **(1)** in the ShieldNN framework: i.e. $\mathcal{C}_{h_{\bar{r},\sigma}}$, the zero super-level set of $h_{\bar{r},\sigma}$, is entirely contained in the set of safe states as proscribed by Problem 1, independent of the choice of σ. See Fig. 2, which also depicts another crucial value, $r_{\min}(\pm\pi) = \bar{r}/(1-\sigma)$.

Remark 1. *Note that $h_{\bar{r},\sigma}$ is independent of the velocity state, v. This will ultimately force ShieldNN filters to intervene only by altering the steering input.*

A barrier function also requires a class $\mathcal{K}$ function, α. For ShieldNN, we choose a linear function

$$\alpha_{v_{\max}}(x) = K \cdot v_{\max} \cdot x \tag{9}$$

where $v_{\max}$ is the assumed maximum linear velocity (see Problem 1), and K is a constant selected according to the following theorem.

Theorem 1. *Consider any fixed parameters $\bar{r}$, ℓ_r and σ. Assume that $0 \le v \le v_{max}$ (as specified by Problem 1). If K is chosen such that:*

$$K \ge K_{\bar{r},\sigma} \triangleq \max(\{1, 1/\bar{r}\}) \cdot \left(\tfrac{\sigma}{2\cdot\bar{r}} + 2\right) \tag{10}$$

then the Lie derivative $\nabla_\chi^T h_{\bar{r},\sigma}(x) \cdot f_{KBM}(\chi,\omega) + \alpha(h_{\bar{r},\sigma}(\chi))$ is a monotonically increasing function in r for all $r \ge \bar{r}$ for each fixed choice of $v \in (0, v_{max}]$ and the remaining state and control variables. In particular, for all $\chi \in \mathcal{C}_{h_{\bar{r},\sigma}}$ such that $v \in (0, v_{max}]$ it is the case that:

$$R_{h_{\bar{r},\sigma}}((r_{min}(\xi),\xi,v)) \subseteq R_{h_{\bar{r},\sigma}}(\chi). \tag{11}$$

In addition to concretely defining our class of candidate barrier functions, Theorem 1 is the essential facilitator of the ShieldNN algorithm. In particular, note that

$$\begin{aligned}\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi,\beta,v) &\triangleq \left[\nabla_\chi^{\mathrm{T}} h_{\bar{r},\sigma}(\chi)\cdot f_{\mathrm{KBM}}(\chi,\omega) + \alpha(h_{\bar{r},\sigma}(\chi))\right]_{\chi=(r_{\min}(\xi),\xi,v)} \\ &= v\left(\tfrac{\sigma}{2\cdot\bar{r}\cdot r_{\min}(\xi)}\sin(\tfrac{\xi}{2})\sin(\xi-\beta) + \tfrac{\sigma}{2\cdot\bar{r}\cdot\ell_r}\sin(\tfrac{\xi}{2})\sin(\beta) + \tfrac{1}{r_{\min}(\xi)^2}\cos(\xi-\beta)\right)\end{aligned} \tag{12}$$

since $h_{\bar{r},\sigma}((r_{\min}(\xi),\xi,v)) = 0$ and $\alpha_{v_{\max}}(0) = 0$. Hence, the set $R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,v))$ is independent of v, so (11) gives a sufficient condition for safe controls **(2)** in terms of a single state variable, ξ, and a single control variable β. This simplifies not only the ShieldNN verifier but also the ShieldNN synthesizer, as we shall demonstrate in the next section.

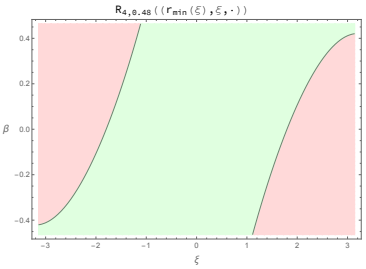
5 ShieldNN

ShieldNN Verifier: The overall ShieldNN algorithm has three inputs: the specs for a KBM robot ($\ell_f = \ell_r$, $\delta_{f,\max}$ and $v_{\max}$); the desired safety radius ($\bar{r}$); and the barrier parameter σ. From these inputs, the ShieldNN verifier first soundly verifies that these parameters lead to an actual barrier function for Problem 1. As per Theorem 1, it suffices to show that $R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,\cdot))$ is non-empty for each $\xi \in [-\pi,\pi]$.

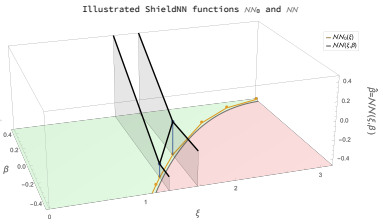
If the sets $R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,\cdot))$ have a complicated structure (both themselves and relative to each other), then establishing this could in principle be quite difficult. However, the barrier functions under consideration actually appear to generate quite nice regions of safe controls. In particular, it appears to the case that the set of safe steering angles in any particular orientation state is an *interval* clipped at the maximum/minimum steering inputs. That is each such set can be written as:

$$R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,\cdot)) = \left[\max\{-\beta_{\max}, \mathfrak{l}(\xi)\},\ \min\{\beta_{\max}, \mathfrak{u}(\xi)\}\right],$$

where $\mathfrak{l}$ and $\mathfrak{u}$ are continuous functions of ξ. Even more helpfully, the function $\mathfrak{l}$ generally appears to be *concave*, and the symmetry of the problem dictates that $\mathfrak{u}(\xi) =$



(a) Safe/unsafe steering controls. $\cup_\xi R_{4,0.48}((r_{\min}(\xi),\xi,\cdot))$ is shown in light green; $\mathfrak{l}$ and $\mathfrak{u}$ in dark green.



(b) Illustration of $\mathcal{NN}_0$ (orange) and two constant-ξ slices of the final ShieldNN filter, $\mathcal{NN}$ (black).

Figure 3: Illustrated ShieldNN products for $\ell_f = \ell_r = 2$ m, $\bar{r} = 4$ m, $\beta_{\max} = 0.4636$, $\sigma = 0.48$.

$-\mathfrak{l}(-\xi)$. See Fig. 3a for an example with parameters $\ell_f = \ell_r = 2$ m, $\bar{r} = 4$ m, $\beta_{\max} = 0.4636$ and $\sigma = 0.48$; $\cup_{\xi\in[-\pi,\pi]} R_{h_{\bar{r},\sigma}}((r_{\min}(\xi), \xi, \cdot))$ is shown in light green, and $\mathfrak{l}$ and $\mathfrak{u}$ are shown in dark green.

Of course these observations about $\mathfrak{l}$ and $\mathfrak{u}$ are difficult to show analytically, given the nature of the equations (c.f. (12)). Nevertheless, we can exhibit a sound algorithm to verify these claims for particular parameter values, and hence that the input parameters correspond to a legitimate barrier function. Due to space constraints, the details of this algorithm appear in the supplementary material.

ShieldNN Synthesizer: Given a verified barrier function, recall from **(3)** in Section 3 that synthesizing a ShieldNN filter requires two components: $\mathscr{NN}_0$ and $\mathscr{NN}$. That is $\mathscr{NN}_0$ chooses a *safe* control for each state, and $\mathscr{NN}$ overrides any *unsafe* controls with the output of $\mathscr{NN}_0$.

Design of $\mathscr{NN}_0$. This task is much easier than it otherwise would be, since the ShieldNN verifier also verifies the safe controls as lying between the continuous functions $\max\{-\beta_{\max}, \mathfrak{l}\}$ and $\min\{\beta_{\max}, \mathfrak{u}\}$ where $\mathfrak{l}$ and is concave and $\mathfrak{u}(\xi) = -\mathfrak{l}(-\xi)$. In particular, then, it is enough to design $\mathscr{NN}_0$ as any neural network such that

$$\max\{-\beta_{\max}, \mathfrak{l}\} \leq \mathscr{NN}_0 \leq \min\{\beta_{\max}, \mathfrak{u}\}. \tag{13}$$

This property can be achieved in several ways, including training against samples of $\max\{-\beta_{\max}, \mathfrak{l}\}$ for example. However, we chose to synthesize $\mathscr{NN}_0$ directly in terms of tangent line segments to $\mathfrak{l}$ (and thus exploit the *concavity* of $\mathfrak{l}$). A portion of just such a function $\mathscr{NN}_0$ is illustrated by the orange line in Fig. 3b.

Design of $\mathscr{NN}$. Since the value of $\mathscr{NN}_0$ is designed to lie inside the interval of safe controls, the function $\mathscr{NN}_0$ can itself be used to decide when an unsafe control is supplied. In particular, using this property and the symmetry $\mathfrak{u}(\xi) = -\mathfrak{l}(-\xi)$, we can simply choose

$$\mathscr{NN} : \beta \mapsto \min\{\max\{\mathscr{NN}_0(\beta), \beta\}, -\mathscr{NN}_0(-\beta)\}. \tag{14}$$

Note: in this construction, the closer $\mathscr{NN}_0$ approximates its lower bound, $\max\{-\beta_{\max}, \mathfrak{l}\}$, the less intrusive the safety filter will be. Two constant-ξ slices of such a $\mathscr{NN}$ are shown in Fig. 3b.

6 ShieldNN Evaluation

We conduct a series of experiments to evaluate ShieldNN's performance when applied to unsafe RL controllers. The CARLA Simulator [2] is used as our RL environment, and we consider an RL agent whose goal is to drive a simulated vehicle while avoiding the obstacles in the environment. The video recordings and other details of the experiments can be found here. The goals of the experiments are to assess the following:

1. The effect of ShieldNN when applied during RL training (Experiment 1) in terms of the average collected reward, obstacle avoidance, etc.
2. The safety of the RL agent when ShieldNN is applied after training (Experiment 2).
3. The robustness of ShieldNN when applied in a different environment than that used in training (Experiment 3).

RL Task: The RL task is to drive a simulated four-wheeled vehicle from point A to point B on a curved road that is populated with obstacles. The obstacles are static CARLA pedestrian actors randomly spawned at different locations between the two points. We define unsafe states as those in which the vehicle hits an obstacle. As ShieldNN is designed for obstacle avoidance, we do not consider the states when the vehicle hits the sides of the roads to be unsafe with respect to ShieldNN. Technical details and graphical representations are included in the Supplementary Materials.

Reward function and termination criteria: If the vehicle reaches point B, the episode terminates, and the RL agent gets a reward value of a 100. The episode terminates, and the agent gets penalized by a value of a 100 in the following cases: when the vehicle (i) hits an obstacle; (ii) hits one of the sides of the road; (iii) has a speed lower than 1 KPH after 5 seconds from the beginning of the episode; or (iv) has a speed that exceed the maximum speed (45 KPH). The reward function is a weighted sum of four terms, and the weights were tuned during training. The four terms are designed in order to incentivize the agent to keep the vehicle's speed between a minimum speed (35

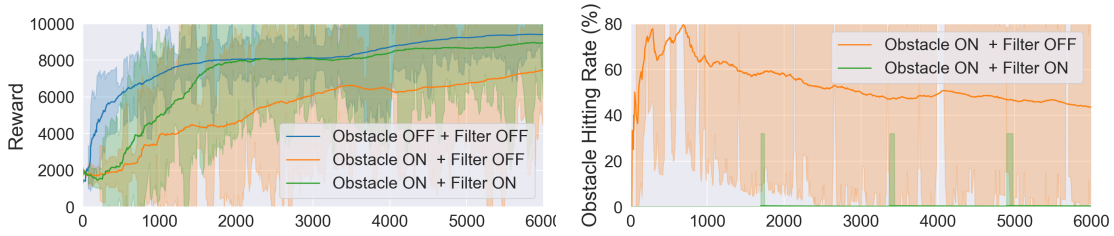
KPH) and a target speed (40 KPH), maintain the desired trajectory, align the vehicle's heading with the direction of travel, and keep the vehicle away from obstacles. The reward function is defined formally in the Supplementary Materials.

Proximal Policy Optimization (PPO) [36] was used to train a neural network to perform the desired RL task. The network receives measurements ξ and r, which are synthesized from CARLA position and orientation measurements to simulate LiDAR input. The network then outputs the new control actions: throttle, ζ, and steering angle, δ_f. The steering angle δ_f is subsequently processed by the ShieldNN filter to produce a corrected "safe" steering angle $\hat{\delta}_f$, which is applied to the simulated car with the original throttle input generated by the PPO agent. The ShieldNN filter is synthesized according to Section 5 with the parameters $\bar{r} = 4\,\text{m}, \sigma = 0.48$ and KBM parameters $\delta_{f_{\max}} = \pi/4, l_f = l_r = 2m$, and $v_{\max} = 20m/s$. The full details of the architecture and signal processing for the agent are provided in the supplementary materials.

Experiment 1: Effect of ShieldNN During RL Training The goal of this experiment is to study the effect of applying ShieldNN to an RL agent during training. We train three RL agents for 6000 episodes each in order to compare (i) the collected reward and (ii) the obstacle hit rate after an equal number of training episodes. The three agents are characterized as follows: Agent 1 is trained with no obstacles and without the ShieldNN filter in place (Obstacles OFF + Filter OFF); Agent 2 is trained with obstacles spawned at random but without ShieldNN in place (Obstacles ON + Filter OFF); and Agent 3 is trained with obstacles spawned at random and with the ShieldNN filter in place (Obstacles ON + Filter ON).

When obstacles are not present (Agent 1), the RL agent quickly learns how to drive the vehicle, as indicated by the rapid growth in the reward function shown in Fig. 4a. When obstacles are present but ShieldNN is not used (Agent 2), the RL agent's ability to learn the task degrades, as indicated by a 30% reduction in collected reward. However, when obstacles are present and the ShieldNN filter is in place (Agent 3), the agent collects 28% more reward on average than Agent 2, and collects a similar amount of reward to Agent 1. This is an indication that ShieldNN filters improves the training of the system by reducing the number of episodes that are terminated early due to collisions.

Similar behavior can be observed in Fig. 4b, which shows the obstacle collision rate (averaged across episodes). This figure shows that Agent 2 is slowly learning how to avoid obstacles, since its average obstacle collision rate decreases from 80% to 47% in 60000 episodes. However, Agent 3, which uses ShieldNN during training, has an obstacle collision rate of almost zero. In total, Agent 3 suffers only three collisions across all 60000 episodes. We believe that these three collisions are due to the discrepancy between the KBM and the dynamics of the vehicle used by the CARLA simulator.



(a) Reward (raw & smoothed data for 3 cases)

(b) Obstacle collision rate

Figure 4: Results of Experiment 1, evaluation of effect of ShieldNN during training.

Experiment 2: Safety Evaluation of ShieldNN The goal of this experiment is to validate the safety guarantees provided by ShieldNN when applied to non-safe controllers. To do this, we evaluate the three trained agents from Experiment 1 in the same environment they were trained in, and with obstacles spawned randomly according to the same distribution used during training. With this setup, we consider two evaluation scenarios: (i) when the ShieldNN filter is in place (ShieldNN ON) and (ii) when ShieldNN filter is not in place (ShieldNN OFF). Table 1 shows all six configurations of this experiment. For each configuration, we run 200 episodes and record three metrics: (i) the minimum distance between the center of the vehicle and the obstacles, (ii) the average percentage of track completion, and (iii) the percentage of hitting obstacles across the 200 episodes.

Fig. 5a and 5b show the histograms of the minimum distance to obstacles for each configuration. The figure also show two vertical lines at 2.3 m and 4 m: the former is the minimum distance at which a collision can occur, given the length of the vehicle, and the latter is the value of the safe

distance $\bar{r}$ used to design the ShieldNN filter. Whenever the ShieldNN was not used in the 200 testing episodes (ShieldNN OFF, Fig. 5a), the average of all the histograms is close to the 2.3 m line indicating numerous obstacle collisions. The exact percentage of obstacle hit rate is reported in Table Table 1. Upon comparing the histograms in Fig. 5a with those in 5b, we conclude that ShieldNN nevertheless renders all the three agents safe: note that the center of mass of the histograms shifts above the safety radius parameter, $\bar{r}$, used to design the ShieldNN filter. In particular, Agents 2 and 3 were able to avoid all the obstacles spawned in all 200 episodes, while Agent 1 hit only 0.5% of the obstacles spawned. Again, we believe this is due to the difference between the KBM used to design the filter and the actual dynamics of the vehicle. In general, the obstacle hitting rate is reduced by $99.4\%, 100\%$ and 100% for Agents 1, 2, and 3, respectively.

	Training		Testing	Experiment 2		Experiment 3A	
Config	**Obstacle**	**Filter**	**Filter**	**TC%**[1]	**OHR%**[2]	**TC%**[1]	**OHR%**[2]
1	OFF	OFF	OFF	7.59	99.5	27.53	79.5
2	OFF	OFF	ON	98.82	0.5	98.73	0.5
3	ON	OFF	OFF	94.82	8.5	71.88	34
4	ON	OFF	ON	100	0	100	0
5	ON	ON	OFF	62.43	44	50.03	60
6	ON	ON	ON	100	0	100	0

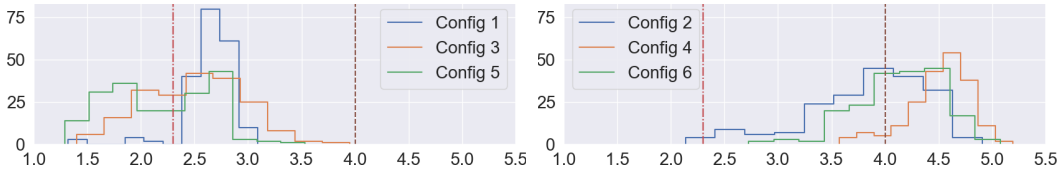
[1] TC% := Track Completion % [2] OHR% := Obstacle Hit Rate %

Table 1: Experiment 2 & 3, evaluation of safety and performance with and without ShieldNN.

Experiment 3: Robustness of ShieldNN in Different Environments The goal of this experiment is to test the robustness of ShieldNN when applied inside a different environment than the one used to train the RL agents. We split the experiment into two parts:

Part 3-A: We use the same setup and metrics as in Experiment 2, but we perturb the locations of the spawned obstacles by a Gaussian distribution $\mathcal{N}(0,\ 1.5)$m in the lateral and longitudinal directions. Fig. 5c and 5d show that despite this obstacle perturbation, ShieldNN is still able to maintain a safe distance between the vehicle and the obstacles whereas this is not the case when ShieldNN is OFF. Table 1 shows an overall increase of obstacle hit rate and a decrease in track completion rate when ShieldNN is OFF compared to the previous experiment. This is expected, as the PPO algorithm is trained with the obstacles spawned at locations with a different distribution than the one used in testing. However, ShieldNN continues to demonstrate its performance and safety guarantees by having almost 100% track completion rate and almost 0% obstacle hit rate.

Part 3-B: We use a completely different environment and track than the ones used in training, but we spawn the obstacles at locations with the same distribution used in training. We first perform transfer learning and train the pretrained Agents 2 and 3 for 500 episodes in the new environment. In this case, ShieldNN still achieves the desired safety distance on average, and achieving exactly zero obstacle hitting rates in both cases; it also achieves track completions of 98% and 97% respectively. Implementation details and results for Experiment 3B are included in the Supplementary Material.



(a) Experiment **2**, ShieldNN OFF (b) Experiment **2**, ShieldNN ON

(c) Experiment **3A**, ShieldNN OFF (d) Experiment **3A**, ShieldNN ON

Figure 5: Distributions of distance-to-obstacles for experiments 2 & 3, with and without ShieldNN.

Side Effects of ShieldNN: In our experiments, applying ShieldNN during training had the side effect of creating a higher curb hitting rate during both training and testing, as compared to the case when the agent was trained with ShieldNN OFF. In particular, after training for 6000 episodes, the

curb hitting rate for agent 2 went from 100% down to 8%. However for agent 3 it went from 100% down to 30%. This is due to the fact that ShieldNN forces the vehicle to steer away from facing an obstacle which, in turn, increases the probability of hitting one of the sides of the road. This side effect suggests the possibility for future research in generalizing ShieldNN to provide safety guarantees against hitting environment boundaries as well.

References

[1] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099, IEEE, 2015.

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[3] W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2067–2069, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[4] A. Liu, G. Shi, S.-J. Chung, A. Anandkumar, and Y. Yue, "Robust regression for safe exploration in control," *arXiv preprint arXiv:1906.05819*, 2019.

[5] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics," *arXiv preprint arXiv:1602.04450*, 2016.

[6] P. Pauli, A. Koch, J. Berberich, and F. Allgöwer, "Training robust neural networks using lipschitz bounds," *arXiv preprint arXiv:2005.02929*, 2020.

[7] C. Gaskett, "Reinforcement learning under circumstances beyond its control," 2003.

[8] T. M. Moldovan and P. Abbeel, "Safe exploration in markov decision processes," *arXiv preprint arXiv:1205.4810*, 2012.

[9] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," in *Advances in Neural Information Processing Systems*, pp. 4312–4320, 2016.

[10] L. Wen, J. Duan, S. E. Li, S. Xu, and H. Peng, "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization," *arXiv preprint arXiv:2003.01303*, 2020.

[11] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in neural information processing systems*, pp. 908–918, 2017.

[12] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, "Lyapunov-based safe policy optimization for continuous control," *arXiv preprint arXiv:1901.10031*, 2019.

[13] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Advances in neural information processing systems*, pp. 8092–8101, 2018.

[14] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6059–6066, IEEE, 2018.

[15] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *53rd IEEE Conference on Decision and Control*, pp. 1424–1431, IEEE, 2014.

[16] V. Govindarajan, K. Driggs-Campbell, and R. Bajcsy, "Data-driven reachability analysis for human-in-the-loop systems," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2617–2622, IEEE, 2017.

[17] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[18] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2460–2465, IEEE, 2018.

[19] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3387–3395, 2019.

[20] K. P. Wabersich and M. N. Zeilinger, "Scalable synthesis of safety certificates from data with application to learning-based control," in *2018 European Control Conference (ECC)*, pp. 1691–1697, IEEE, 2018.

[21] M. Srinivasan, A. Dabholkar, S. Coogan, and P. Vela, "Synthesis of control barrier functions using a supervised machine learning approach," *arXiv preprint arXiv:2003.04950*, 2020.

[22] A. J. Taylor, A. Singletary, Y. Yue, and A. D. Ames, "A control barrier perspective on episodic learning via projection-to-state safety," *arXiv preprint arXiv:2003.08028*, 2020.

[23] X. Li and C. Belta, "Temporal logic guided safe reinforcement learning using control barrier functions," *arXiv preprint arXiv:1903.09885*, 2019.

[24] R. Cheng, M. J. Khojasteh, A. D. Ames, and J. W. Burdick, "Safe multi-agent interaction through robust control barrier functions with learned uncertainties," *arXiv preprint arXiv:2004.05273*, 2020.

[25] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," *arXiv preprint arXiv:2004.03315*, 2020.

[26] G. Shi, X. Shi, M. OConnell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9784–9790, IEEE, 2019.

[27] X. Sun, H. Khedr, and Y. Shoukry, "Formal verification of neural network controlled autonomous systems," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 147–156, 2019.

[28] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Output range analysis for deep feedforward neural networks," in *NASA Formal Methods Symposium*, pp. 121–138, Springer, 2018.

[29] C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," *arXiv preprint arXiv:1903.06758*, 2019.

[30] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of lipschitz constants for deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 11423–11434, 2019.

[31] W. Xiang, D. M. Lopez, P. Musau, and T. T. Johnson, "Reachable set estimation and verification for neural network models of nonlinear dynamic systems," in *Safe, Autonomous and Intelligent Vehicles*, pp. 123–144, Springer, 2019.

[32] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: verifying safety properties of hybrid systems with neural network controllers," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 169–178, 2019.

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

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

[35] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of Control Barrier Functions for Safety Critical Control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

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

[37] A. Raffin, A. Hill, K. R. Traoré, T. Lesort, N. Díaz-Rodríguez, and D. Filliat, "Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics," *arXiv preprint arXiv:1901.08651*, 2019.

Proofs for Section 4

Additional Notation

Throughout the rest of this appendix we will use the following notation:

$$\begin{aligned}\mathscr{L}_{\bar{r},\sigma,\ell_r}(\chi,\beta) &\triangleq \nabla_\chi^{\mathrm{T}} h_{\bar{r},\sigma}(\chi)\cdot f_{\text{KBM}}(\chi,(\beta,a))\\ &= v\left(\frac{\sigma}{2\cdot\bar{r}\cdot r}\sin(\xi/2)\sin(\xi-\beta)+\frac{\sigma}{2\cdot\bar{r}\cdot\ell_r}\sin(\xi/2)\sin(\beta)+\frac{\cos(\xi-\beta)}{r^2}\right).\end{aligned} \tag{15}$$

Where f_{KBM} is the right-hand side of the ODE in (1) and the variable a is merely a placeholder, since the (40) doesn't depend on it at all. In particular, (40) has the following relationship with (12):

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi,\beta,v) = \mathscr{L}_{\bar{r},\sigma,\ell_r}((r_{\min}(\xi),\xi,v),\beta). \tag{16}$$

Moreover, we define the following set:

$$\tilde{\mathcal{C}}_{h_{\bar{r},\sigma}} \triangleq \left\{\chi'=(r',\xi',v') \;\middle|\; h(\chi')\geq 0 \;\wedge\; 0<v'\leq v_{\max}\right\}, \tag{17}$$

which is the subset of the zero-level set of $h_{\bar{r},\sigma}$ that is compatible with our assumption that $0<v\leq v_{\max}$ (see Problem 1).

Proofs for Section 4

There are two claims from Section 4 that require proof.

1. First, we stated Theorem 1 without proof.
2. Second, we claimed that for any KBM parameters $\ell_r=\ell_f$ and $\delta_{f_{\max}}$, there exists a safety radius, $\bar{r}$, and a barrier parameter, σ, such that $h_{\bar{r},\sigma}$ and $\alpha_{v_{\max}}$ (as defined in (7) and (9)) comprise a barrier function for the KBM.

We provide proofs for each of these in the next two subsections.

Proof of Theorem 1

We prove the first claim of Theorem 1 as the following Lemma.

Lemma 1. *Consider any fixed parameters $\bar{r}$, ℓ_r, σ and $v_{max}>0$. Furthermore, define*

$$K_{\bar{r},\sigma} \triangleq \max(\{1,1/\bar{r}\})\cdot\left(\tfrac{\sigma}{2\cdot\bar{r}}+2\right). \tag{18}$$

Now suppose that $h_{\bar{r},\sigma}$ is as in (7), and $\alpha_{v_{max}}$ is as in (9) with K is chosen such that $K\geq K_{\bar{r},\sigma}$. Then for each $(\xi,v,\beta)\in[-\pi,\pi]\times(0,v_{max}]\times[-\beta_{max},\beta_{max}]$, the function

$$L_{\xi,v,\beta}: r\in[\bar{r},\infty)\mapsto \mathscr{L}_{\bar{r},\sigma,\ell_r}((r,\xi,v),\beta)+\alpha_{v_{max}}(h_{\bar{r},\sigma}((r,\xi,v))) \tag{19}$$

is increasing on its domain, $\operatorname{dom}(L_{\xi,v,\beta})=[\bar{r},+\infty)$.

Remark 2. *Note the relationship between the function $L_{\xi,v,\beta}$ in (19) and the function used to to define $R_{h,\alpha}$ in Corollary 1. That is the set that we are interested in characterizing in Theorem 1.*

Proof. We will show that when $K\geq K_{\bar{r},\sigma}$, each such function $L_{\xi,v,\beta}$ has a strictly positive derivative on its domain. In particular, differentiating $L_{\xi,v,\beta}$ gives:

$$\begin{aligned}\frac{\partial}{\partial r}L_{\xi,v,\beta}(r) &= \frac{\partial}{\partial r}\left[\mathscr{L}_{\bar{r},\sigma,\ell_r}((r,\xi,v),\beta)+\alpha_{v_{\max}}(h_{\bar{r},\sigma}((r,\xi,v)))\right]\\ &= v\left(-\tfrac{\sigma}{2\cdot\bar{r}\cdot r^2}\sin(\xi/2)\sin(\xi-\beta)-2\tfrac{\cos(\xi-\beta)}{r^3}\right)+\tfrac{K\cdot v_{\max}}{r^2}\\ &\geq v\left(-\tfrac{\sigma}{2\cdot\bar{r}\cdot r^2}-\tfrac{2}{r^3}\right)+\tfrac{K\cdot v_{\max}}{r^2}.\end{aligned} \tag{20}$$

To ensure that this derivative is strictly positive, it suffices to choose K such that

$$v\left(-\frac{\sigma}{2\cdot\bar{r}\cdot r^2}-\frac{2}{r^3}\right)+\frac{K\cdot v_{\max}}{r^2}\geq 0. \tag{21}$$

For this, we consider two cases: $\bar{r}<1$ and $\bar{r}\geq 1$.

When $\bar{r}\geq 1$, then $1/r^3\leq 1/r^2$ for all $r\geq\bar{r}$. Thus it suffices to choose K such that

$$K\geq\frac{v}{v_{\max}}\left(\frac{\sigma}{2\cdot\bar{r}}+2\right), \tag{22}$$

which is assured under the assumption that $v\in(,v_{\max}]$ if

$$K\geq\frac{\sigma}{2\cdot\bar{r}}+2. \tag{23}$$

Now, when $\bar{r}<1$, choosing K according to (23) ensures that (21) is true for all $r\geq 1$. Thus, we also have to ensure (21) holds for $\bar{r}\leq r<1$. But in this case, $1/r^3\geq 1/r^2$, so (21) will be satisfied if

$$K\geq\frac{1}{\bar{r}}\left(\frac{\sigma}{2\cdot\bar{r}}+2\right). \tag{24}$$

Thus, the desired conclusion holds if we choose $K\geq K_{\bar{r},\sigma}$ as defined in the statement of the lemma. □

Now, we have the prerequisites to prove Theorem 1.

Proof. (Theorem 1) The first claim of Theorem 1 is proved as Lemma 1. Thus, it remains to show that for any $\chi=(r,\xi,v)\in\mathcal{C}_{h_{\bar{r},\sigma}}$ with $v\in(0,v_{\max}]$ — that is $\chi\in\tilde{\mathcal{C}}_{h_{\bar{r},\sigma}}$ — we have that (11) holds. However, this follows from Lemma 1.

In particular, choose an arbitrary $\chi'=(r',\xi',v')\in\tilde{\mathcal{C}}_{h_{\bar{r},\sigma}}$, and choose an arbitrary $\omega'=(\beta',a')\in R_{h_{\bar{r},\sigma}}((r_{\min}(\xi'),\xi',v'))$; as usual we will only need to concern ourselves with the steering control, β'. First, observe that by definition:

$$\begin{aligned}(\beta',a')\in R_{h_{\bar{r},\sigma}}((r_{\min}(\xi'),\xi',v'))\\ \implies \mathscr{L}_{\bar{r},\sigma,\ell_r}((r_{\min}(\xi'),\xi',v'),\beta')+\alpha_{v_{\max}}(h_{\bar{r},\sigma}((r_{\min}(\xi'),\xi',v')))\geq 0.\end{aligned} \tag{25}$$

However, the conclusion of this implication can be rewritten using the definition (19):

$$(\beta',a')\in R_{h_{\bar{r},\sigma}}((r_{\min}(\xi'),\xi',v'))\implies L_{\xi',v',\beta'}(r_{\min}(\xi'))\geq 0. \tag{26}$$

We now invoke Lemma 1: since $r_{\min}(\xi')\geq\bar{r}$ by construction, Lemma 1 indicates that $L_{\xi',v',\beta'}$ is strictly increasing on the interval $[r_{\min}(\xi'),r']$. Combining this conclusion with (26), we see that $L_{\xi',v',\beta'}(r')\geq 0$. Again using the definition of $L_{\xi',v',\beta'}$ in (26), we conclude that

$$\mathscr{L}_{\bar{r},\sigma,\ell_r}((r',\xi',v'),\beta')+\alpha_{v_{\max}}(h_{\bar{r},\sigma}(r',\xi',v')))\geq 0. \tag{27}$$

Thus, we conclude that $(\beta',a')\in R_{h_{\bar{r},\sigma}}(\chi')$ by the definition thereof (see the statement of Theorem 1). Finally, since χ' and ω' were chosen arbitrarily, we get the desired conclusion. □

Proof of That a Barrier Function Exists for Each KBM Instance

For $h_{\bar{r},\sigma}$ and $\alpha_{v_{\max}}$ to be a useful class of barrier functions, it should be that case that at least one of these candidates is in fact a barrier function for each instance of the KBM. We make this claim in the form of the following Theorem.

Theorem 2. *Consider any KBM robot with length parameters $\ell_r=\ell_f$; maximum steering angle $\delta_{f_{max}}$; and maximum velocity $v_{max}>0$. Furthermore, suppose that the following two conditions hold:*

(i) $\beta_{max}\leq\pi/2$, *or equivalently,* $\delta_{f\,max}\leq\frac{\pi}{2}$;

(ii) $\frac{1}{\ell_r}\left(\sigma(1-\sigma)\ell_r+\sigma\bar{r}\right)\sin(\frac{\pi}{4}+\frac{\beta_{max}}{2})\sin(\beta_{max})\geq 2$; *and*

Then for every $\chi = (r, \xi, v)$ such that $0 < v \leq v_{max}$ the set $R_{h_{\bar{r},\sigma}}(\chi)$ is non-empty. In particular, the feedback controller (interpreted as a function of ξ only):

$$\pi : \xi \mapsto \begin{cases} -\beta_{max} & \xi < -\epsilon \\ \xi & \xi \in [-\beta_{max}, \beta_{max}] \\ \beta_{max} & \xi > \epsilon \end{cases} \tag{28}$$

is safe.

Remark 3. *Note that there is always a choice of $\bar{r}$ and $\sigma \in (0, 1)$ such that condition (ii) can be satisfied. In particular, it suffices for $\bar{r}$ and σ to be chosen such that:*

$$2\frac{(\ell_r/\bar{r})}{\sin(\beta_{max})\sin(\pi/4 + \beta_{max}/2)} \leq \sigma. \tag{29}$$

Thus, by making $\bar{r}$ large enough relative ℓ_r, it is possible to choose a $\sigma \in (0, 1)$ such that the inequality (29) holds, and (ii) is satisfied.

Proof. (Theorem 2) As a consequence of Theorem 1, it is enough to show that $R_{h_{\bar{r},\sigma}}((r_{\min}(\xi), \xi, v_{\max}))$ is non-empty for every $\xi \in [-\pi, \pi]$.

The strategy of the proof will be to consider the control $\beta = \pi(\xi)$, and verify that for each $\chi = (r, \xi, v) \in \tilde{\mathcal{C}}_{h_{\bar{r},\sigma}}$ such that $\xi \in [0, \pi]$, we have:

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi, \pi(\xi)) \geq 0. \tag{30}$$

The symmetry of the problem will allow us to make a similar conclusion for $\xi \in [-\pi, 0]$.

We proceed by partitioning the interval $[0, \pi]$ into the following three intervals:

$$I_1 \triangleq [0, \beta_{\max}], \qquad I_2 \triangleq (\beta_{\max}, \pi/2 + \beta_{\max}], \qquad I_3 \triangleq (\pi/2 + \beta_{\max}, \pi].$$

and consider the cases that ξ is in each such interval separately.

Case 1 $\xi \in I_1$**:** In this case, $\pi(\xi) = \xi$, and $\xi \leq \beta_{\max} \leq \pi/2$ by assumption. It is direct to show that:

$$\cos(\xi - \pi(\xi)) = \cos(0) \geq 0 \tag{31}$$

and

$$\sin(\xi/2)\sin(\xi - \pi(\xi)) = 0. \tag{32}$$

Hence, the cos term in (12) can be lower bounded by zero, and the first term in (12) is identically zero by (32). Thus, in this case, (12) is lower bounded as as:

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi, \beta_{\max}) \geq \frac{\sigma \cdot v \cdot \sin(\xi/2)\sin(\pi(\xi))}{2 \cdot \bar{r} \cdot \ell_r}, \tag{33}$$

which of course will be greater than zero since $\xi \in I_1 = [0, \beta_{\max}]$ with $\beta_{\max} \leq \pi/2$ by assumption *(i)*.

Case 2 $\xi \in I_2$**:** In this case, $\pi(\xi) = \beta_{\max}$. Thus, for $\xi \in I_2$, we have that:

$$\cos(\xi - \beta_{\max}) \geq 0 \tag{34}$$

$$\sin(\xi/2)\sin(\xi - \beta_{\max}) \geq 0 \tag{35}$$

$$\sin(\xi/2)\sin(\beta_{\max}) \geq 0. \tag{36}$$

Consequently, (30) is automatically satisfied, since all of the quantities in the Lie derivative are positive.

Case 3 $\xi \in I_3$**:** In this case, $\pi(\xi) = \beta_{\max}$ as in Case 2. However, the cos term is now negative in this case:

$$0 > \cos(\xi - \beta_{\max}) \geq -\frac{1}{\bar{r}^2}. \tag{37}$$

Thus, since the other two terms are positive on this interval, we need to have:

$$\sin(\tfrac{1}{2}(\tfrac{\pi}{2} + \beta_{\max}))\Big(\tfrac{\sigma(1-\sigma)}{2\cdot\bar{r}^2}\sin(\pi - \beta) + \tfrac{\sigma}{2\cdot\bar{r}\cdot\ell_r}\sin(\beta)\Big) \geq \tfrac{1}{\bar{r}^2}. \tag{38}$$

This follows because on I_3, $\sin(\frac{\xi}{2}) \geq \sin(\frac{1}{2}(\frac{\pi}{2} + \beta_{\max}))$ and $\sin(\xi - \beta_{\max}) \geq \sin(\pi - \beta_{\max})$; i.e. we substituted the lower and upper end points of I_3, respectively. Noting that $\sin(\pi - \beta_{\max}) = \sin(\beta_{\max})$, we finally obtain:

$$\sin(\tfrac{1}{2}(\tfrac{\pi}{2} + \beta_{\max}))\sin(\beta_{\max})\left(\tfrac{\sigma(1-\sigma)}{2} + \tfrac{\sigma\bar{r}}{2\cdot\ell_r}\right) \geq 1. \tag{39}$$

The preceding is just another form of *(ii)* so we have the desired conclusion in (30).

The conclusion of the theorem then follows from the combined consideration of Cases 1-3 and Theorem 1 as claimed above. □

Proofs for Section 5

Additional Notation

Throughout the rest of this appendix we will use the following notation:

$$\begin{aligned}\mathscr{L}_{\bar{r},\sigma,\ell_r}(\chi,\beta) &\triangleq \nabla_\chi^{\mathrm{T}} h_{\bar{r},\sigma}(\chi)\cdot f_{\text{KBM}}(\chi,(\beta,a)) \\ &= v\left(\frac{\sigma}{2\cdot\bar{r}\cdot r}\sin(\xi/2)\sin(\xi-\beta)+\frac{\sigma}{2\cdot\bar{r}\cdot\ell_r}\sin(\xi/2)\sin(\beta)+\frac{\cos(\xi-\beta)}{r^2}\right). \quad (40)\end{aligned}$$

Where f_{KBM} is the right-hand side of the ODE in (1) and the variable a is merely a placeholder, since the (40) doesn't depend on it at all. In particular, (40) has the following relationship with (12):

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi,\beta,v) = \mathscr{L}_{\bar{r},\sigma,\ell_r}((r_{\min}(\xi),\xi,v),\beta). \quad (41)$$

Moreover, we define the following set:

$$\tilde{\mathcal{C}}_{h_{\bar{r},\sigma}} \triangleq \left\{\chi'=(r',\xi',v') \;\middle|\; h(\chi')\geq 0 \;\wedge\; 0<v'\leq v_{\max}\right\}, \quad (42)$$

which is the subset of the zero-level set of $h_{\bar{r},\sigma}$ that is compatible with our assumption that $0<v\leq v_{\max}$ (see Problem 1).

Proofs

ShieldNN Verifier

Recall that the main function of the ShieldNN verifier to soundly verify that

$$R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,\cdot)) = \left[\,\max\{-\beta_{\max},\mathfrak{l}(\xi)\},\;\min\{\beta_{\max},\mathfrak{u}(\xi)\}\,\right], \quad (43)$$

for a concave function $\mathfrak{l}$ and with $\mathfrak{u}(\xi)=-\mathfrak{l}(-\xi)$. The conclusion about $\mathfrak{u}$ follows directly from the symmetry of the problem, so we will focus on verifying the claims for $\mathfrak{l}$.

As a foundation for the rest of this subsection, we make the following observation.

Proposition 1. *Suppose that* (43) *holds with* $\mathfrak{u}(\xi)=-\mathfrak{l}(-\xi)$. *Then for any* $\xi'\in[-\pi,\pi]$ *such that* $\mathfrak{l}(\xi')\in(-\beta_{max},\beta_{max})$ *it is the case that*

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi',\mathfrak{l}(\xi'),\cdot)=0. \quad (44)$$

Proof. This follows directly from the definition of $R_{h_{\bar{r},\sigma}}$, and the fact that we are considering it *on* the barrier, i.e. for $\chi'=(r_{\min}(\xi'),\xi',v)$ which implies that $h(\chi')=0$ and hence that $\alpha_{v_{\max}}(h(\chi'))=0$. □

This suggests that we should start from (44) in order to establish the claim in (43). To this end, let $a<b$ be real numbers, and define:

$$\mathsf{bd}_{[a,b]} \triangleq \{(\xi',\beta')\in[a,b]\times[-\beta_{\max},\beta_{\max}] \;\mid\; \mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi',\beta',\cdot)=0\} \quad (45)$$

with the appropriate modifications for other interval types (a,b), $(a,b]$ and $[a,b)$. We also define a related quantity:

$$\text{dom}(\mathsf{bd}_{[a,b]}) = \{\xi\in[a,b]\mid\exists\beta.(\xi,\beta)\in\mathsf{bd}_{[a,b]}\}. \quad (46)$$

We can thus develop a sound algorithm to verify (43) and the concavity of $\mathfrak{l}$ by soundly verifying the following three properties in sequence:

- **Property 1.** Show that $\mathsf{bd}_{[-\pi,\pi]}\cap\left([-\pi,\pi]\times\{-\beta_{\max}\}\right)=\{(\xi_0,\beta_{\max})\}$; that is $\mathsf{bd}_{[-\pi,\pi]}$ intersects the lower control constraint a single orientation angle, ξ_0. And likewise $\mathsf{bd}_{[-\pi,\pi]}\cap\left([-\pi,\pi]\times\beta_{\max}\right)=\{(-\xi_0,\beta_{\max})\}$ by symmetry.
- **Property 2.** Verify that $\mathsf{bd}_{[\xi_0,\pi]}$ is the graph of a function (likewise for $\mathsf{bd}_{[-\pi,-\xi_0]}$ by symmetry), and that $\mathsf{bd}_{(-\xi_0,\xi_0)}=\emptyset$. Thus, define $\mathfrak{l}$ according to $\text{graph}(\mathfrak{l})\triangleq\mathsf{bd}_{[\xi_0,\pi]}$.

Property 3. Verify that $\mathfrak{l}$ as defined in **Property 2** is concave.

The ShieldNN verifier algorithm expresses each of these properties as the sound verification that a particular function is greater than zero on a subset of its domain. Naturally, the functions that are associated with these properties are either $\mathcal{L}$ itself or else derived from it (i.e. literally obtained by differentiating), and so each is an analytic function where the variables ξ and β appear only in trigonometric functions. Thus, these surrogate verification problems are easily approachable by over-approximation and the Mean-Value Theorem.

With this program in mind, the remainder of this appendix consists of one section each explaining how to express **Property 1**-**3** as minimum-verification problems. These are followed by a section that describes the main algorithmic component of the ShieldNN verifier, `CertifyMin`.

Verifying Property 1

To verify **Property 1**, we can start by using a numerical root finding algorithm to find a zero of $\mathcal{L}(\xi, -\beta_{\max}, \cdot)$, viewed as a function of ξ. However, there is no guarantee that this root, call it $\hat{\xi}_0$ is the only root on the set $[-\pi, \pi] \times \{-\beta_{\max}\}$. Thus, the property to be verified in this case in the assumptions of the following proposition.

Proposition 2. *Suppose that $\mathcal{L}(\hat{\xi}_0, -\beta_{max}, \cdot) = 0$. Furthermore, suppose that there exists an $\epsilon > 0$ such that:*

(i) $\forall \xi \in [-\pi, \hat{\xi}_0 - \epsilon]$. $\mathcal{L}(\hat{\xi}_0 - \epsilon, -\beta_{max}, \cdot) > 0$*;*

(ii) $\mathcal{L}(\hat{\xi}_0 - \epsilon, -\beta_{max}, \cdot) > 0$ *and* $\mathcal{L}(\pi, -\beta_{max}, \cdot) < 0$*;*

(iii) $\forall \xi \in [\hat{\xi}_0 - \epsilon, \pi]$. $\frac{\partial^2}{\partial \xi^2}\mathcal{L}(\xi, -\beta_{max}, \cdot) > 0$.

*Then $\hat{\xi}_0$ is the only root of $\mathcal{L}(\xi, -\beta_{max}, \cdot)$ on $[-\pi, \pi] \times \{-\beta_{max}\}$. That is **Property 1** is verified.*

Proof. If *(i)* is true, then there are obviously no zeros of $\mathcal{L}$ on $[-\pi, \hat{\xi}_0 - \epsilon]$.

If *(iii)* is true, then $\mathcal{L}(\xi, -\beta_{\max}, \cdot)$ is a convex function of ξ on the interval $[\hat{\xi}_0, \pi]$. But if *(ii)* is also true, then $\hat{\xi}_0$ must be the only zero of $\mathcal{L}$ on the same interval. This follows by contradiction from the assertion of convexity. If there were another zero on $(\hat{\xi}_0, \pi]$, then the line connecting $(\hat{\xi}_0, \mathcal{L}(\hat{\xi}_0, -\beta_{\max}, \cdot))$ and $(\pi, \mathcal{L}(\pi, -\beta_{\max}, \cdot))$ would lie below this point by assumption *(ii)*, hence contradicting convexity. A similar argument can be made if there were a zero on $[\hat{\xi}_0 - \epsilon, \hat{\xi}_0)$. □

Crucially, the conditions *(i)*-*(iii)* of Proposition 2 are conditions that can be checked either by verifying that a function is greater than 0 on an interval (as for *(ii)* and *(iii)*), or else that $\mathcal{L}$ as a particular sign for particular inputs (as in *(i)*). Thus, ShieldNN verifier can establish **Property 1** by means of the `CertifyMin` function that we propose later.

Verifying Property 2

Our verification of **Property 2** depends on the conclusion of **Property 1**. In particular, let $\xi_0 = \hat{\xi}_0$ be the single root of $\mathcal{L}$ on $\left([-\pi, \pi] \times \{-\beta_{\max}\}\right)$ as verified above. As before, we provide a proposition that gives us sufficient conditions to assert the conclusion of **Property 2**, and where verifying those conditions requires at worst checking the sign of some $\mathcal{L}$-derived function on an interval (or rectangle).

The main technique for proving that $\mathsf{bd}_{[\xi_0,\pi]}$ is the graph of a function is to note that constant-level curves of $\mathcal{L}$ are solutions to the ODE defined by its gradient. In particular, then, $\mathsf{bd}_{[\xi_0,\pi]}$ contains such a solutions in the rectangle of interest, since it is a subset of the zero-constant level curve of $\mathcal{L}$. Thus, we can verify the desired properties of $\mathsf{bd}_{[\xi_0,\pi]}$ by considering the aforementioned ODE, and demonstrating that it has only one solution in the rectangle of interest. This is the subject of the following proposition and the structure of its subsequent proof.

Proposition 3. *Let ξ_0 be as above. Now suppose the following two conditions are satisfied:*

(i) $\frac{\partial}{\partial \xi}\mathcal{L}(\xi_0, -\beta_{max}, \cdot) > 0$*;*

(ii) for all $(\xi, \beta) \in \big([\xi_0, \pi] \times [-\beta_{max}, \beta_{max}]\big)$ *it is the case that*

$$\frac{\partial}{\partial \beta}\mathcal{L}(\xi, \beta, \cdot) < 0; \tag{47}$$

and

(iii) there exists $\epsilon > 0$ *and* $\hat{\beta}_0 \in [-\beta_{max}, \beta_{max}]$ *such that*

(I) $\forall \xi \in [-\beta_{max}, \hat{\beta}_0 - \epsilon]$. $\mathcal{L}(\pi, \hat{\beta}_0 - \epsilon, \cdot) < 0$*;*
(II) $\forall \beta \in [\hat{\beta}_0 - \epsilon, \beta_{max}]$. $\frac{\partial}{\partial \beta}\mathcal{L}(\pi, \beta, \cdot) > 0$.

Then $\mathsf{bd}_{[\xi_0,\pi]}$ *is the graph of a function, and we can define the function* $\mathfrak{l}$ *on* $[\xi_0, \pi]$ *by graph*$(\mathfrak{l}) \triangleq \mathsf{bd}_{[\xi_0,\pi]}$.

Proof. Consider the ODE defined by:

$$\begin{aligned} \dot{\xi} &= -\tfrac{\partial}{\partial \beta}\mathcal{L}(\xi, \beta, \cdot) \\ \dot{\beta} &= \tfrac{\partial}{\partial \xi}\mathcal{L}(\xi, \beta, \cdot). \end{aligned} \tag{48}$$

The solutions to (48) are guaranteed to exist and be unique on $[\xi_0, \pi] \times [-\beta_{\max}, \beta_{\max}]$, since the vector field is locally Lipschitz on that rectangle (it is differentiable). Thus, any solution of (48) is guaranteed to follow a constant-level curve of $\mathcal{L}$ within this rectangle; the particular constant-level curve is decided by the value of $\mathcal{L}$ for its initial condition.

As a first step, we establish two facts about the solution of (48) with initial condition $(\xi_0, -\beta_{\max})$:

1. the β component of this solution is strictly increasing; and
2. the solution exits $[\xi_0, \pi] \times [-\beta_{\max}, \beta_{\max}]$ only through its $\xi = \pi$ edge.

First note that some initial portion of this solution must be contained in $[\xi_0, \pi] \times [-\beta_{\max}, \beta_{\max}]$ assumption *(i)* and assumption *(ii)* applied to $(\xi_0, -\beta_{\max})$. Statement 1 is thus established directly by assumption *(ii)*. Now we establish 2. Note that the solution cannot exit via the $\xi = \xi_0$ edge because its β component is strictly increasing. And it can't exit the $\beta = \mp\beta_{\max}$ edges either, because we have verified that $\mathcal{L}$ has only one root on each edge, $(\xi, -\beta_{\max})$ and $(-\xi_0, \beta_{\max})$, respectively. This leaves only the $\xi = \pi$ edge. The solution must leave this rectangle eventually, by the exclusion of the other edges and the fact that its β component is strictly increasing. Thus, it exits via the $\xi = \pi$ edge.

The final conclusion about the functionality follows if $\mathsf{bd}_{[\xi_0,\pi]}$ if $\mathsf{bd}_{[\xi_0,\pi]}$ corresponds *exactly* to the single, unique solution described above. To verify this, we need to verify that there is a single root of $\mathcal{L}$ along the $\xi = \pi$ edge, much as we did to verify **Property 1**; this is made possible by assumption*(iii)*, items *(I)-(II)*. □

As in the case of Proposition 2, the conditions of Proposition 3 are conditions that can be checked using the `CertifyMin` function that we will propose subsequently.

Verifying Property 3

We verify **Property 3** starting from the assumption that verifications of **Property 1** and **Property 2** were successful. In particular, we assume a function $\mathfrak{l}$ with domain $[\xi_0, \pi]$ that defines the lower boundary of the set $R_{h_{\bar{r},\sigma}}$, and which is characterized entirely by $\mathcal{L}(\xi, \mathfrak{l}(\xi), \cdot) = 0$.

Since $\mathfrak{l}$ corresponds exactly to such a constant-level contour, we can use derivatives of $\mathcal{L}$ to compute the derivative of $\mathfrak{l}$ with respect to ξ. That is if we define

$$\gamma'(\xi, \beta) \triangleq -\frac{\partial \mathcal{L}_{\bar{r},\sigma,\ell_r}/\partial \xi}{\partial \mathcal{L}_{\bar{r},\sigma,\ell_r}/\partial \beta}(\xi, \beta) \tag{49}$$

then $\mathfrak{l}'(\xi) = \gamma'(\xi, \mathfrak{l}(\xi))$.

By extension then, it is possible to derive the *second* derivative of $\mathfrak{l}$ using $\mathcal{L}$ if we define:

$$\gamma''(\xi, \beta) \triangleq \frac{\partial}{\partial \xi}\gamma'(\xi, \beta) + \frac{\partial}{\partial \beta}\gamma'(\xi, \beta) \cdot \gamma'(\xi, \beta) \tag{50}$$

so that $\mathfrak{l}''(\xi) = \gamma''(\xi, \mathfrak{l}(\xi))$. This gives us an obvious sufficient condition for the *concavity* of $\mathfrak{l}$.

Proposition 4. *Suppose that ξ_0 and graph$(\mathfrak{l}) \triangleq \mathsf{bd}_{[\xi_0,\pi]}$ as above. If for all $(\xi, \beta) \in [\xi_0, \pi] \times [-\beta_{max}, \beta_{max}]$ we have that*

$$\gamma''(\xi, \beta) < 0 \tag{51}$$

then $\mathfrak{l}$ is concave.

Proof. Direct from the calculations above. □

`CertifyMin` and the ShieldNN Verifier

We have listed a number of conditions, which when verified together, are sufficient to prove that a particular set of parameters leads $h_{\bar{r},\sigma}$ to be a barrier function for the KBM. Furthermore, each of these conditions involves asserting that $\mathcal{L}$ or its derivatives are strictly positive or negative on an interval or a rectangle.

Since $\mathcal{L}$ is composed of relatively simple functions, it is possible for a computer algebra system (CAS) to not only obtain each of these verification functions automatically, but to further differentiate each of them once *more*. Thus, we can combine an extra derivative with the Mean-Value Theorem to verify each of these individual claims. We describe this procedure as `CertifyMin` below.

Algorithm 1: `CertifyMin`.

input : function $f(\xi, \beta)$ that is either $\mathcal{L}$ or one of its derivatives; ξ-interval $[\xi_\ell, \xi_h]$; β-interval $[\beta_\ell, \beta_h]$; a sign $s = \pm 1$. **Either $\xi_\ell = \xi_h$ or $\beta_\ell = \beta_h$ may be true but not both.**

output: N_est

```
function CertifyMin(f, ξ_ℓ,ξ_h, β_ℓ, β_h, s)
    Df ← SymbolicGradient(f)
    if ξ_ℓ = ξ_h then
        DfNorm ← SymbolicNorm(SymbolicGetComponent(Df, β))
    else if β_ℓ = β_h then
        DfNorm ← SymbolicNorm(SymbolicGetComponent(Df, ξ))
    else
        DfNorm ← SymbolicNorm(Df)
    end
    gridSize ← 1
    refine ← True
    while refine do
        gridSize ← gridSize/10
        refine ← False
        for (ξ', β') in GridIterator(gridSize,ξ_ℓ,ξ_h, β_ℓ, β_h) do
            if s · f(ξ', β') < 0 then
                return False
            else if s · f(ξ', β') < √2· gridSize · DfNorm(ξ', β') then
                refine ← True
                break
            end
        end
    end
    return True
end
```

Experimental Details from Section 6

Experimental Details

Integrating ShieldNN with PPO

We train a Proximal Policy Optimization (PPO) [36] neural network in order to perform the desired RL task. To speed up policy learning as in [37], we encode the front camera feed into a latent vector using the encoder part of a trained β-Variational Auto-Encoder (β-VAE). As shown in Fig. 6b, the encoder takes 160x80 RGB images generated by the simulated vehicle's front facing camera and outputs a latent vector that encodes the state of the surroundings. The inputs to the PPO Network are: The latent vector $[z_1, ..., z_{dim}]$, the vehicle's inertial measurements (current steering angle δ_f^c, speed v and acceleration a) and the relative angle ξ and distance r between the vehicle and the nearest obstacle. The latter two measurements are estimated using an obstacle detection module that takes the vehicle's LIDAR data as input. In our experiments, we assume we have a perfect obstacle detection estimator and we implement it by collecting the ground truth position and orientation measurements of the vehicle and the obstacles from CARLA then calculating ξ and r. The PPO network outputs the new control actions: Throttle ζ and steering angle δ_f. We omit using the brakes as part of the control input vector, as it is not necessary for this task. However, the RL agent will still be able to slow down the vehicle by setting the throttle value to 0 due to the simulated wheel friction force in CARLA. The throttle control action ζ gets passed directly to CARLA, while the steering angle control action gets filtered by ShieldNN. The filter also takes ξ and r as input and generates a new safe steering angle $\hat{\delta_f}$. To train the VAE, we first collect 10,000 images by driving the vehicle manually in CARLA along the desired route with obstacles spawned at random locations and observing different scenes from different orientations. We train the VAE encoder with cross validation and early-stopping. Then, after convergence, we visually inspect the reconstructed image to test the accuracy of the VAE encoder.

Reward Function

The reward function used for training all three agents is as follows:

$$R(s) = W_s * R_{Speed} + W_c * R_{Centering} + W_a * R_{Angle} + W_d * R_{DistToObst}$$

, where

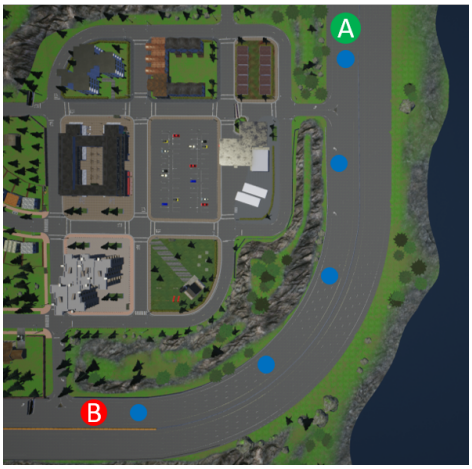
$$R_{Speed} = \begin{cases} \frac{v}{v_{min}}, & v < v_{min} \\ 1 - \frac{v_{target}}{v_{max}}, & v > v_{target} \\ 1 & otherwise \end{cases}$$

$$R_{Centering} = \max\left(1 - \frac{d_c}{l_{max}}, 0\right)$$

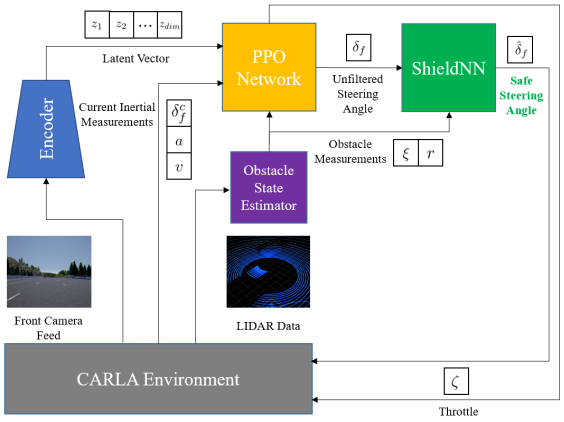
$$R_{Angle} = \max\left(1 - \left|\frac{a_c}{\frac{\pi}{9}}\right|, 0\right)$$

$$R_{DistToObst} = \max\left(\min(\frac{r}{r_{max}}, 1), 0\right)$$

v is the current vehicle's speed, d_c is the current lateral distance between the center of the vehicle and the track, a_c is the current angle between the heading of the vehicle and the tangent of the track's curvature, r is the distance between the center of the vehicle and the nearest obstacle, $r = 0$ if obstacle doesn't exist, $l_{max} = 10m, r_{max} = 20m, v_{min} = 35KPH, v_{max} = 45KPH, v_{target} = 40KPH$.



(a) RL Task: Goal is to drive the vehicle from point A to point B without hitting random obstacles (the blue circles) spawned along the route



(b) Integration Framework of ShieldNN with PPO inside a CARLA simulator Environment.

Figure 6: Environment Setup and Integration Framework

Experimental Results

Part 3-B: Robustness of ShieldNN in New Track

This experiment is essentially an evaluation of the ability (or not) of RL agents equipped with ShieldNN to generalize to novel environments. To evaluate performance in this setting, a transfer learning task is implemented where the pretrained Agents 2 and 3 are then *retrained* for 500 episodes in the new environment (compare to 6000 training episodes for the original experiments). The new environment is a city road surrounded by buildings, as opposed to the urban highway environment used in the original training. This modification substantially shifts the distribution of the camera feed input.

Fig. 7 shows the results for Configurations 4 and 6; recall from Table 1 in the main text that these configurations represent when (re)training is conducted with ShieldNN OFF and ON, respectively. Note that configuration 2 – where the original training was done in an environment with no obstacles – could not successfully complete the track during re-training for 500 episodes nor in testing, and is thus not included in the figure. Observe that in both configurations, the agent is still able to avoid obstacles for the 200 number of test episodes. Furthermore, configuration 6 (both retraining and testing with ShieldNN ON), the agent appears to behave more conservatively with respect to obstacle avoidance.

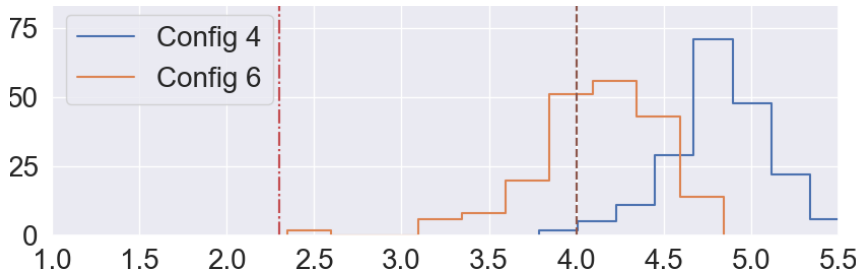


Figure 7: Results of Experiment 3-B, distributions of performance metrics within a completely novel environment relative to training.