# Reachability Barrier Networks: Learning Hamilton-Jacobi Solutions for Smooth and Flexible Control Barrier Functions

**Matthew Kim**[1] **William Sharpless**[1] **Hyun Joe Jeong**[1] **Sander Tonkens**[1]
**Somil Bansal**[2] **Sylvia Herbert**[1]
[1]University of California, San Diego [2] Stanford University
mak009@ucsd.edu

**Abstract:** Recent developments in autonomous driving and robotics underscore the necessity of safety-critical controllers. Control barrier functions (CBFs) are a popular method for appending safety guarantees to a general control framework, but they are notoriously difficult to generate beyond low dimensions. Existing methods often yield non-differentiable or inaccurate approximations that lack integrity, and thus fail to ensure safety. In this work, we use physics-informed neural networks (PINNs) to generate smooth approximations of CBFs by computing Hamilton-Jacobi (HJ) optimal control solutions. These reachability barrier networks (**RBNs**) avoid traditional dimensionality constraints and support the tuning of their conservativeness post-training through a parameterized discount term. To ensure robustness of the discounted solutions, we leverage conformal prediction methods to derive probabilistic safety guarantees for RBNs. We demonstrate that RBNs are highly accurate in low dimensions, and safer than the standard neural CBF approach in high dimensions. Namely, we showcase the RBNs in a 9D multi-vehicle collision avoidance problem where it empirically proves to be $5.5$x safer and $1.9$x less conservative than the neural CBFs, offering a promising method to synthesize CBFs for general nonlinear autonomous systems.

**Keywords:** Reachability Analysis, Optimal Control, Deep Learning, Control Barrier Functions

## 1 Introduction

Autonomous systems have revolutionized the way humans interact with technology. However, as these systems often operate in unpredictable environments, it is necessary to ensure safety and reliability. A key challenge lies in designing control mechanisms that allow autonomous systems to achieve their goals efficiently while strictly adhering to safety requirements. For example, when an autonomous vehicle detects a potential collision with another vehicle, it should be able to safely avoid the vehicle, ideally without much deviation from its original trajectory. In a different setting, a user of a robotic arm may want the arm to deliver a cup to a specific location without damaging the cup. These examples showcase why autonomous systems must be scalable and verifiably safe in order to be deployed in the real world. Toward this end, safety filters [1], a control-theoretic technique for providing safety guarantees to the system, are useful because they minimally modify the nominal actions of the robot while keeping it safe.

Control barrier functions (CBFs) [2] are a popular method for ensuring safety because they not only guarantee safety when the system is within the safe set but also minimally adjust the nominal control input when necessary to maintain safety. However, CBFs are often hand-designed, requiring an expert to tune the function for a given task specification. In addition, designing these CBFs requires tuning the exponential decay term, which governs the aggressiveness of the system's controller. Data-driven

Workshop on Safe and Robust Robot Learning for Operation in the Real World at CoRL 2025.

CBF methods [3, 4, 5] learn the control invariant set and associated control policy through supervised learning or reinforcement learning. Typically, these methods provide no formal guarantees on these learned approximations, but recent works use uncertainty quantification and other tools to provide formal or probabilistic guarantees [6, 7, 8, 9]. Moreover, as demonstrated in prior work [10, 11] and our findings, the learned CBF often fails to satisfy the necessary conditions to be a valid CBF, especially in the presence of control bounds.

Hamilton-Jacobi reachability (HJR) analysis is a method that can be used for safety analysis that rigorously evaluates the set of states that may lead to failure under worst-case conditions by solving a partial differential equation (PDE) [12, 13]. The result of the analysis is a value function, whose level sets and gradients inform the safe set and associated safe control policy. Recent work [14, 15] demonstrates that HJR can be adapted to directly compute CBF-like functions. This allows for the blending of the two approaches for applications in robotics and controls, merging the ease of use of CBFs with the rigorous guarantees and constructive nature of HJR. However, two issues remain: a) HJR relies on dynamic programming, which suffers from the curse of dimensionality, and b) the resulting CBF may not be differentiable everywhere. This differentiability issue is critical, as the gradient must exist to generate the set of safe actions at each state [16].

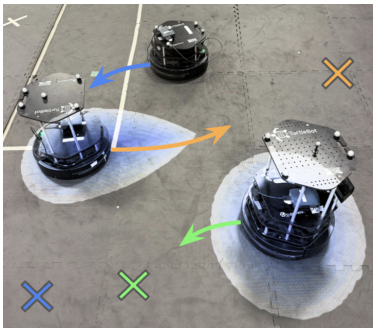

Figure 1: **Sample experiment setup.** Agents, endowed with a joint safety filter to avoid collisions, are independently attempting to reach their goals. The blue contours indicate the unsafe region for the blue agent at its heading.

We propose a resolution to these two key issues by building upon DeepReach [17], a framework that merges HJR with physics-informed neural networks (PINNs). DeepReach computes a high-confidence solution to the HJR problem using the residual PDE error as the loss function. Interestingly, the computation requirements of DeepReach scale with the complexity of the underlying solution (rather than with the number of state variables), making it suitable for high-dimensional analysis. Moreover, recent works [18, 19, 20] have shown that various inputs to a Hamilton-Jacobi solution can be parameterized, including the control input, disturbance bound, as well as the cost function, allowing for online adaptation in uncertain and changing environments. A final benefit is that the learning error can be estimated via conformal prediction, which can subsequently be used to provide probabilistic safety assurances for the system [21].

In this paper, we adapt DeepReach to compute finite-time CBFs, yielding a unifying approach to the construction of CBFs for high-dimensional nonlinear systems with probabilistic safety guarantees. Specifically, we make the following contributions:

*Smooth High-Dimensional CBFs with Bounded Control Inputs.* We introduce the Reachability Barrier Network (RBN), a learned CBF approximation using a modified version of DeepReach. Because DeepReach employs sinusoidal activation functions, we guarantee that the resulting CBF approximation is differentiable, making it readily suitable for synthesizing safe actions. Due to the underlying HJR-based backbone, the proposed approach allows us to synthesize CBF approximations for general nonlinear systems, while accounting for input constraints.

*Adaptive Conservativeness Online.* CBFs are parameterized by an exponential decay rate $\gamma$ that tunes the aggressiveness of the controller. We propose to use a $\gamma$-parameterized RBN, through a linear time (i.e. $O(N)$) self-supervised learning process, allowing for post-training adaptation of the controller.

*Probabilistic Safety Guarantees.* We provide probabilistic safety assurances to the RBN generated from conformal prediction tools. Using these assurances, we augment the learned safe set to provide the desired probabilistic guarantees. We explore the relationship between the aggressiveness of the controller (via the decay rate $\gamma$) and the resulting probabilistic guarantees.

We first compare our approach to the dynamic programming-based solution in a low-dimensional 3D Dubin's car setting. We then demonstrate the scalability of our approach in a higher-dimensional 9D

multi-vehicle collision avoidance setting and compare it to a popular neural CBF architecture from [5]. We validate our approach offline and online via conformal predictions and trajectory rollouts, respectively. Moreover, we analyze the robustness of our method compared to other baselines in a goal-seeking multi-vehicle collision avoidance hardware experiment.

## 2 Background

### 2.1 Hamilton-Jacobi Reachability

Consider a dynamical system with state $x \in \mathbb{R}^{n_s}$, time horizon $\mathbb{T} = [t, t_f]$, trajectory $\mathsf{x} : \mathbb{T} \to \mathbb{R}^{n_s}$, control input $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$, control signal $\mathsf{u} : \mathbb{T} \to \mathcal{U}$, and initial condition $\mathsf{x}(t) = x$. Let the evolution of the system be defined by $\dot{\mathsf{x}} = f(\mathsf{x}(t), \mathsf{u}(t))$, which we assume to be Lipschitz continuous with respect to $\mathsf{x}$ and $\mathsf{u}$, and continuous with respect to $t$.

Consider a failure set $\mathcal{F} \subset \mathbb{R}^{n_x}$, which the state may not enter, to be encoded by a Lipschitz continuous function $l : \mathbb{R}^{n_x} \to \mathbb{R}$ s.t. $\mathcal{F} := \{x \mid l(x) < 0\}$. We seek to solve $\mathsf{u}$ that keeps $\mathsf{x}$ out of $\mathcal{F}$ at all times in the horizon. Let the value then for any initial $(x, t)$ then be defined by [22],

$$V(x,t) = \sup_{\mathsf{u} \in \mathbb{U}} \min_{s \in [t, t_f]} l(\mathsf{x}(s)). \tag{1}$$

Note, the zero level set of this value, $\mathcal{R}(t) \triangleq \{x \mid V(x,t) \leq 0\}$, characterizes the set of initial $(x, t)$ for which there exists a safe $\mathsf{u}$ s.t. $\mathsf{x}(\tau) \notin \mathcal{F}$,

$$\mathcal{R}(t) = \{x \mid \exists \mathsf{u} \text{ s.t. } \mathsf{x}(\tau) \notin \mathcal{F}, \tau \in \mathbb{T}\}, \tag{2}$$

and is thus called the *avoid tube*. Notably, the value function is the solution to the HJ Variational Inequality [23, 24],

$$0 = \min \left\{ l(x) - V(x,t),\ \frac{\partial V}{\partial t} + H(x, \nabla_x V, t) \right\},\ \ V(x, t_f) = l(x) \tag{3}$$

where $H(x, p, t) = \max_{u \in \mathcal{U}} p \cdot f(x, u)$ is the Hamiltonian of the system. Moreover, the value function yields an optimal safe control policy for online control

### 2.2 Control Barrier Functions

Control barrier functions [2] are a powerful tool for guaranteeing the safety of dynamic systems, ensuring that a system remains within a safe region indefinitely. A control barrier function by definition satisfies,

$$\max_{u \in U} \nabla B(x) \cdot f(x, u) \geq -\alpha(B(x)), \quad \forall x \in C,\ C = \{x \in \mathbb{R}^n : B(x) \geq 0\}, \tag{4}$$

where $\alpha$ is a class $\kappa_\infty$ function, often chosen to be $\alpha(B(x)) = \gamma B(x)$ (see [1, 2] for more details). This condition ensures that any control input within the control bounds will cause a system to remain within the safe region. Online, the resulting CBF may be used as a safety filter [1],

$$u^*(x) = \arg\min_{u \in \mathcal{U}} \|u - u_{\text{nom}}(x)\|_2^2 \ \text{ s.t. (4) holds.} \tag{5}$$

Here $u_{\text{nom}} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$ represents a given nominal control law that might violate control limits and safety. For control-affine systems, (5) is a quadratic program, enabling real-time safety filtering.

The main drawback of CBF-based approaches is the challenge of constructing a valid CBF. Often, CBFs are hand-designed for specific applications and systems. Some recent constructive methods have been proposed specifically for collision avoidance applications [25] and by using data-driven approaches [26, 27, 28, 29, 30]. In many cases, these methods produce approximate CBFs that require significant parameter tuning [31] and may not have formal guarantees, particularly when faced with bounded control inputs.

### 2.3 Control Barrier Value Functions (CBVFs)

In an effort to provide a constructive method for CBFs, [14] introduced the notion of a control barrier value function (CBVF), a CBF-like function $V_\gamma : \mathbb{R}^n \times \mathbb{T} \to \mathbb{R}$, defined as

$$V_\gamma(x,t) := \sup_{\mathbf{u}\in\mathbb{U}} \min_{s\in[t,t_f]} e^{\gamma(s-t)} l(x(s)). \tag{6}$$

CBVFs can be computed via HJR by solving (3) with the class $\kappa_\infty$ function, $\alpha(V_\gamma(x,t)) = \gamma V_\gamma(x,t)$,

$$0 = \min\left\{ l(x) - V_\gamma(x,t),\ \frac{\partial V_\gamma}{\partial t} + H(x, \nabla_x V_\gamma, t) + \gamma V_\gamma(x,t) \right\},\ \ V_\gamma(x,t_f) = l(x). \tag{7}$$

This allows for the construction of CBF-like functions over a finite time horizon with user-defined control bounds, disturbance bounds, and discount rate $\gamma$. This constructive approach has been used to refine CBF approximations from hand-tuned or data-driven approaches [15, 10].

However, there are two main challenges with the CBVF formulation. First, the CBVF is not technically a CBF directly because the function may not be everywhere differentiable. This is problematic for the online CBF-QP controller (5), which relies on the gradients of the function to maintain safety. The second challenge is that this technique for the construction of CBFs suffers from the aforementioned "curse of dimensionality." To be useful for many practical autonomous systems, the scalability of CBF construction must be pushed to handle higher-dimensional and more realistic models of systems and their operating environments.

## 3 Learned Hamilton-Jacobi CBFs via RBNs

### 3.1 Offline Learning Procedure

We propose Reachability Barrier Networks (RBNs) as a unifying method for the construction of CBFs for high-dimensional nonlinear systems with probabilistic safety guarantees. Specifically, we propose modifying DeepReach [17], PINN used to solve HJ-PDEs, to compute a parameterized approximation of a CBF. The residual of (7) is employed as a loss $\mathcal{L}$ to learn the CBVF,

$$\mathcal{L}(\theta) = \mathbb{E}_{x,t,\gamma}\left[ \left\| \min\left\{ l(x) - V_\theta(x,t,\gamma),\ \frac{\partial V_\theta}{\partial t} + H(x, \nabla_x V_\theta, t) + \gamma V_\theta(x,t,\gamma) \right\} \right\| \right],$$

where $V_\theta(x,t,\gamma) = l(x) + (t_f - t)\cdot \text{NN}_\theta(x,t,\gamma)$ and $\text{NN}_\theta$ is a neural network with sinusoidal activation functions and parameters $\theta$ [32]. The learned value $V_\theta$ is by definition infinitely differentiable and grid-free, overcoming the challenges of CBFs and CBVFs. Moreover, this approach is amenable to conformal corrections with probablistic assurances as we discuss in the following section.

To allow for online adaptation of the safety filter, we input the exponential decay rate $\gamma$ as a learned parameter to the model. Akin to a CBF, when $\gamma$ is high, the resulting RBN allows for more aggressive behavior, and as $\gamma$ approaches zero, the RBN becomes conservative, forcing safe behavior. Adding $\gamma$ to the state space increases the dimension by one, but the run-time cost is negligible in high-dimensions as neural nets do not suffer from the "curse of dimensionality."

### 3.2 Probabilistic Guarantees via Conformal Prediction

After training, we use the conformal prediction-based method proposed in [33] to estimate the learning error and derive a probabilistic assurance on the learned CBF approximation. Namely, given a desired probability $\epsilon$ of safety, the method estimates an upper bound $\delta$ on the learned error such that the super-$\delta$ level of $V_\theta(x,t,\gamma)$ is safe at least with probability $1-\epsilon$,

$$\mathbb{P}\left(V(x,t,\gamma) \le 0 \mid V_\theta(x,t,\gamma) > \delta\right) < \epsilon. \tag{8}$$

The learned value function $V_\theta$ is then shifted,

$$V_\theta^\delta(x,t,\gamma) = V_\theta(x,t,\gamma) - \delta \tag{9}$$

such that the super-$\delta$ level set becomes the new safe set with the associated $1-\epsilon$ probability of validity. Note, the gradients of the value function do not change and a robust learned value is recovered.

We note the $\gamma$ parameterization may also affect the probabilistic guarantee. As $\gamma$ varies, the gradients of $V_\gamma$ [14] are affected, which can both improve or degrade the learning performance and thus result in varying levels of $\delta$-correction. This is investigated in Fig. 3 and Sec. 4. In practice, one may tune $\gamma$ or $\delta$ to adapt the RBN in real time to meet the desired probabilistic constraints.

### 3.3 Online Control Synthesis

Online, RBNs can be used directly in the CBF-QP safety filter (5), where the conservativeness of the filter can be varied via $\gamma$ in the RBNs, $V_\theta(x, t, \gamma)$.

## 4 Simulation Results

We first compare our approach to the dynamic programming-based CBVF from [14]. As this method is intractable in high dimensions, we compare the approaches in a 3D Dubin's car obstacle avoidance problem. We then scale our approach to a 9D multi-vehicle collision avoidance problem in both simulation and hardware, and compare with neural CBFs [5], a popular learning-based CBF, alongside a more standard pairwise collision avoidance approach.

### 4.1 Low Dimensional Model: 3D Dubin's Car

The baseline model used to evaluate the performance of the RBNs is a 3D Dubin's car model (note, the system becomes 4D with the parameterization of $\gamma$). The system is defined by $\dot{x} = v\cos(\theta)$, $\dot{y} = v\sin(\theta)$, $\dot{\theta} = u$, and $\dot{\gamma} = 0$, where $x$ and $y$ are the euclidean coordinates, $\theta$ is heading, $v$ is a fixed velocity, our control is the angular velocity, $u \in [-1.1, 1.1]$. We train the sine-activated $3 \times 512$ PINN over $200K$ epochs with a learning rate of $2 \times 10^{-5}$ for approximately 4.5 hours on an A40 GPU. The low-dimensional "ground truth" CBVF is based on high-fidelity dynamic programming, using the HelperOC toolbox [34]. Additional parameters for our method and the HJR solution can be found in Appendix 7.1.

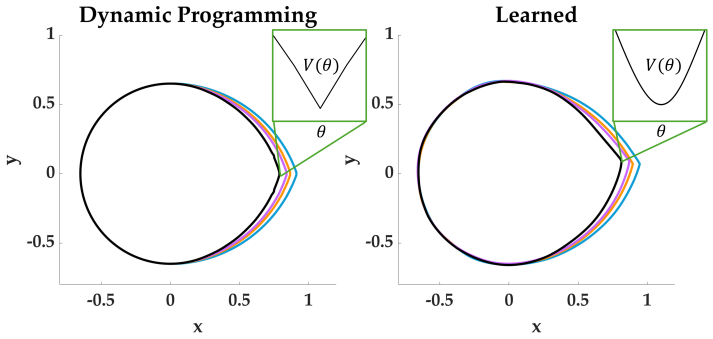


Figure 2: **Low-dimensional ground truth comparison.** The level set of the CBVF (left) and the RBN (right) are shown for $\theta = \pi$. Sets are plotted for $\gamma = \{0.0, 0.3, 0.5, 1.0\}$, shown in blue, orange, purple, and black, respectively. The insets (green) show the change in value over $\theta$, which exhibits the differentiability of the value function.

**Comparison.** For values $\gamma \in \{0.0, 0.3, 0.5, 1.0\}$, we compare the true and learned level sets and optimal policy trajectories at time $t = 1$. Performance is evaluated using: a) the intersection-over-union (IOU) between the sets ; b) falsely included points (FI) ; and c) falsely excluded points (FE). We evaluate these metrics directly on the learned CBF approximation (without conformal expansion).

The results in Table 1 indicate that the learned value function closely approximates the dynamic programming-based solution. Fig. 2 shows slices of the RBNand dynamic programming-based CBVF at different rates of $\gamma$

Table 1: **Validation metrics of the learned CBF.** Sets are scored by discretizing the space and comparing against ground truth using IOU, FI, and FE.

| $\gamma$ | IOU (%) | FI (%) | FE (%) |
|---|---|---|---|
| 0.0 | 96.85 | 2.51 | 0.64 |
| 0.3 | 97.57 | 1.90 | 0.53 |
| 0.5 | 97.77 | 1.81 | 0.42 |
| 1.0 | 97.87 | 1.89 | 0.24 |

(note, we show the super-0.4 level sets, as the super-0 level sets are by definition identical across values of $\gamma$). The inset of the learned value function plotting $V(\theta)$ shows that our approach is everywhere differentiable (unlike CBVFs), allowing for more effective safety filtering via (5).

### 4.2 High Dimensional Model: 9D Multi-Vehicle Collision Avoidance

The 9D multi-vehicle collision avoidance problem consists of three Dubin's cars seeking to avoid collision with one another. The system dynamics combine the individual models of the three Dubin's cars, with an additional parameterization of $\gamma$. Failure in simulation occurs when the distance between two vehicles falls below a collision radius of 0.25 meters. We train the sine-activated $3 \times 512$ PINN on 300K epochs with a learning rate of $2 \times 10^{-5}$.

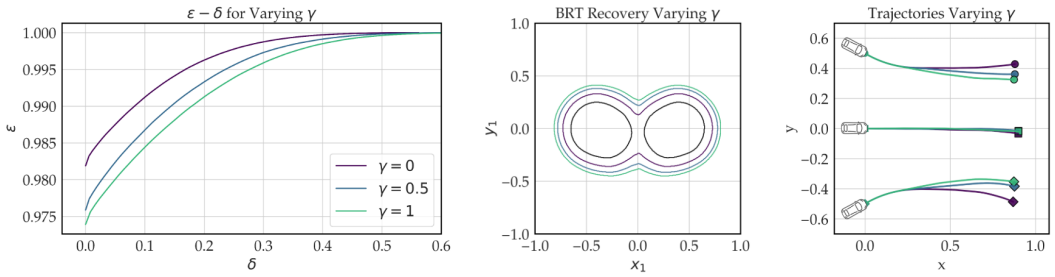


Figure 3: **Conformal expansions of the 9D learned value functions.** (Left) Relationship between $\delta$ and probabilistic guarantee of safety ($\epsilon$) for varying $\gamma$. (Middle) Comparison of conformal expansions for $\gamma = 0, 0.5, 1$. The uncalibrated zero-level set is included in black. (Right) Trajectory rollouts of the 9D multi-agent collision avoidance problem. Increasing $\gamma$ makes the control more aggressive and decreases the distance between agents.

To derive probabilistic guarantees on the learned value function, we leverage conformal prediction [33] to expand the level sets of the value function. We sample $3M$ initial conditions and simulate trajectory rollouts using the learned policy. For each trajectory, we compare the final cost, which is the minimum distance between agents over the trajectory minus the collision radius, to the value assigned at the initial condition by the learned model. This yields a distribution of discrepancies between the learned and safety controller estimates. With this distribution, a confidence $\epsilon$ with a corresponding violation $\delta$ is used to determine that all learned trajectories will have the correct safety representation (i.e. value sign) with confidence $\epsilon$. Fig. 3 (left, middle) shows that an increase in the aggressiveness of our controller (i.e. an increase in gamma) requires a larger expansion of the safe set for the same probabilistic assurance. This follows intuitively, since Fig. 3 (right) shows a reduction in minimum distance as $\gamma$ increases, equivalent to more aggressive controls. We choose $\epsilon \in [0.985, 0.995]$ to prevent outliers from influencing the recovered set to unnecessarily large proportions while also rigorously maintaining safety.

**Comparison.** We compare our RBN method to the neural CBF approach proposed by [5]. For a single training run, both methods require approximately 7 hours of training on A40 GPUs. However, unlike RBN, the neural CBF approach requires retraining the network for each desired gamma value. We use the false positive rate (FPR), and the false negative rate (FNR) to determine our model's performance. A false positive occurs when the initial condition is in the learned safe set (i.e. $V_\theta(x, t, \gamma) \geq 0$) but the rollout results in collision. The negation is true for false negatives. We also include the correctly characterized (CC) percentage, given by $1 - \text{FPR} - \text{FNR}$. Table 2 shows that our method best preserves safety for the multi-vehicle collision avoidance problem with or without a QP. While our expanded method yields the highest FNR, this is to be expected as the value function becomes more conservative and does not affect the true success of the rollouts. Fig. 4 shows a sample state where the neural CBF is unable to maintain safety for a region our method is verifiably safe.

| Setting | Metric | Neural CBF [5] | | | RBN (Ours) | | | Expanded RBN (Ours) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\gamma = 0.0$ | $\gamma = 0.5$ | $\gamma = 1.0$ | $\gamma = 0.0$ | $\gamma = 0.5$ | $\gamma = 1.0$ | $\gamma = 0.0$ | $\gamma = 0.5$ | $\gamma = 1.0$ |
| QP Rollout | FPR (%) | 28.4 | 28.2 | 30.8 | **0.8** | **1.4** | **2.2** | **0.0** | **0.0** | **0.0** |
| | FNR (%) | 1.8 | 1.6 | 1.6 | 0.0 | 0.0 | 0.0 | 6.8 | 12.0 | 16.0 |
| | **CC (%)** | 69.8 | 70.2 | 67.6 | 99.2 | 98.6 | 97.8 | 93.2 | 88.0 | 84.0 |
| Learned Policy Rollout | FPR (%) | 27.0 | 28.2 | 30.0 | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** |
| | FNR (%) | 2.2 | 2.6 | 1.4 | 0.4 | 0.4 | 0.4 | 7.4 | 10.0 | 11.2 |
| | **CC (%)** | 70.8 | 69.2 | 68.6 | 99.6 | 99.6 | 99.6 | 93.6 | 90.0 | 88.8 |

Table 2: **Simulated rollout metrics** across $\gamma$ values for Neural CBF, RBN, and Expanded RBN in both QP rollouts and rollouts using only the learned safe policy (i.e. no nominal control).

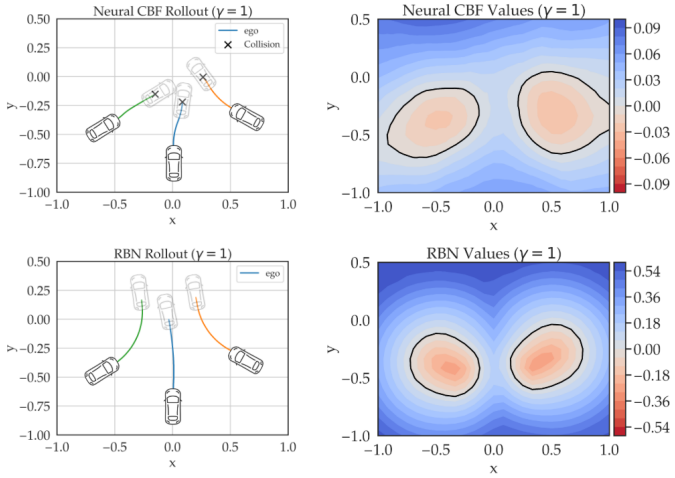


Figure 4: **Rollouts and contours.** (Left) Sample rollouts and their corresponding initial condition level sets (Right) for neural CBFs (Top) and our method (Bottom).

## 5 Hardware Results

The hardware setup is designed to represent an autonomous multi-vehicle context, and mimics the simulations. The Dubin's car model is used to drive TurtleBots, operating at a constant linear velocity of $v = 0.4$ m/s, with control inputs $u \in [-1.1, 1.1]$ rad/s. We use a motion capture system to assume perfect state estimation; a detailed explanation can be found in Appendix 7.2. During each 60-second trial, we solve the CBF-QP (5) to generate safe control inputs that let each agent pursue its assigned goal independently while jointly avoiding collisions—defined as any time the distance between two agents falls below 0.4 m. The goals are defined by targets with a radius of 0.3 m. When an agent reaches its goal, a new goal is assigned, independent of the other agents. If a collision occurs, all agents are instructed to stop and turn away from each other until they are safe to avoid chained collisions. A simple proportional controller nominally controls the angular velocity, directing each agent towards its goal region. However, in some cases, the safety constraints produce deadlock scenarios in which agents are unable to make progress and ultimately collide. We address this issue using the resolution strategy described in Appendix 7.2.

As solving the full 9D multi-vehicle collision avoidance game via dynamic programming is infeasible, we first compare our method to a two-player dynamic-programming solution. Online, the two-player solution is used in a decentralized, pairwise manner: each agent computes its own CBF relative to its nearest neighbor, where the ego agent assumes the other agent acts cooperatively. While this two-player baseline cannot fully capture higher-order multi-agent interactions, it serves as the strongest available solution that can be computed exactly. This comparison thus highlights both the practical limitations imposed by the curse of dimensionality and the benefits of scalable, learning-based approaches such as RBNs. Our principal comparison is to neural CBFs [5], as both methods are trained to address the same 9D multi-vehicle collision avoidance problem. All training hyperparameters can be found in Appendix 7.1.

To evaluate the safety and efficiency of each approach, we compare the success of reaching goals and the number of collisions incurred as the primary metrics. We focus our evaluation on in-distribution collisions, as the learned methods are only trained within a prescribed domain: specifically, $x \in [-1, 1]$, $y \in [-1, 1]$, and $\theta \in [-\pi, \pi]$. However, we also report the total number of collisions to assess how well learning-based approaches generalize beyond their training distribution.

| Controller | $\gamma = 0.0$ | | | $\gamma = 0.5$ | | | $\gamma = 1.0$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | Collisions | In-Dist | Goals | Collisions | In-Dist | Goals | Collisions | In-Dist | Goals |
| RBN(ours) | 0.2 | **0.0** | 12.6 | 0.4 | **0.2** | **29.2** | 1.8 | 0.8 | **30.6** |
| Two-player (coop) | **0.0** | **0.0** | 5.4 | **0.2** | **0.2** | 20.6 | 2.2 | 2.2 | 24.4 |
| Neural CBF | 2.6 | 1.4 | **14.0** | 4.2 | 3.0 | 19.6 | 6.0 | 4.4 | 16.0 |
| Nominal | 7.4 | 7.4 | 20.8 | 7.4 | 7.4 | 20.8 | 7.4 | 7.4 | 20.8 |

Table 3: **Hardware comparison** of total collisions, in-distribution collisions, and goals reached for RBN (ours), neural CBF, and HJR-based controllers across different values of $\gamma$. All metrics are reported per minute and are averaged over 5 minutes of hardware simulation.

**Comparison to pairwise collision avoidance.** Our in-distribution collision rate, as shown in Table 3, is comparable to that of the cooperative two-player solution when $\gamma = 0.0$ or $0.5$. However, since increasing $\gamma$ reduces conservativeness, more three-agent interactions occur when $\gamma = 1.0$, resulting in 2.8x the in-distribution collision rate for the two-player cooperative solution compared to our method. Furthermore, our method achieves 1.4x the goal completion rate of the two-player cooperative solution at $\gamma = 0.5$, and 1.3x at $\gamma = 1.0$. We hypothesize that increasing the number of agents will correlate with a higher relative score for our method compared to the HJR solution since the pairwise solution are unable to safely handle multi-agent interactions.

**Comparison to neural CBFs.** We attribute the subpar performance of neural CBFs in part to difficulties in constructing an effective boundary region between safe and unsafe sets. The neural CBFs require predefining safe and unsafe regions, whereas our method only requires instantiating the boundary function. This allows our method to be less complex to implement and train. Ultimately, the RBNs on average incur 5.5x fewer collisions and reach 1.9x as many goals as the neural CBFs.

Overall, our method performs safely even when using highly aggressive controls at $\gamma = 1.0$, whereas increasing the aggressiveness for other methods compromises their safety. Our controller also exhibits the lowest deviation from the nominal objective, consistently achieving the highest goal completion rates. This trend is also illustrated in Fig. 5, where, under induced three-agent collisions, our controller most effectively avoids collisions.

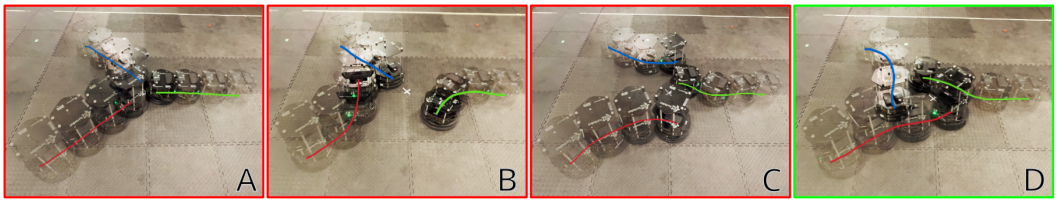


Figure 5: **Comparison of the controllers in hardware** (the robots become less transparent as time progresses). A) Nominal controller. B) neural CBF controller. C) HJ cooperative pairwise controller. D) RBNs (our method). Our method is the only one capable of safely navigating this scenario.

## 6 Conclusion

In this paper, we introduce RBNs, a method for computing a $\gamma$-parameterized control barrier function, and discuss the insights of the probabilistic guarantees for varying $\gamma$. We build intuition through a low-dimensional navigation example and demonstrate the scalability of our work through a high-dimensional multi-vehicle collision avoidance example. We demonstrate the efficacy of our method through hardware experiments, where we achieve high success rates relative to other methods.

While RBNs scale better than dynamic programming, training still demands substantial compute, motivating exploration of sampling-efficient or warm-starting methods [35]. To address out-of-distribution scenarios that challenge neural networks, future work will incorporate uncertainty and risk quantification [36]. Our approach also assumes known system models and constraint sets, which can be relaxed through reduced-order modeling [37]. Finally, the current reliance on perfect state estimation via motion capture will be extended to realistic sensing (e.g., LiDAR, RGB cameras) and robust disturbance-aware estimation [38].

## References


[1] K.-C. Hsu, H. Hu, and J. F. Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

[2] 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)*, pages 3420–3431. IEEE, 2019.

[3] S. Liu, C. Liu, and J. Dolan. Safe control under input limits with neural control barrier functions. In *Conference on Robot Learning*, pages 1970–1980. PMLR, 2023.

[4] W. Lavanakul, J. Choi, K. Sreenath, and C. Tomlin. Safety filters for black-box dynamical systems by learning discriminating hyperplanes. In *6th Annual Learning for Dynamics & Control Conference*, pages 1278–1291. PMLR, 2024.

[5] C. Dawson, Z. Qin, S. Gao, and C. Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1724–1735. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/dawson22a.html.

[6] T. Kim, R. I. Kee, and D. Panagou. Learning to refine input constrained control barrier functions via uncertainty-aware online parameter adaptation. *arXiv preprint arXiv:2409.14616*, 2024.

[7] H. Hu, Y. Yang, T. Wei, and C. Liu. Verification of neural control barrier functions with symbolic derivative bounds propagation, 2024. URL https://arxiv.org/abs/2410.16281.

[8] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724, 2020. doi:10.1109/CDC42340.2020.9303785.

[9] W. Xiao, R. Hasani, X. Li, and D. Rus. Barriernet: A safety-guaranteed layer for neural networks, 2021. URL https://arxiv.org/abs/2111.11277.

[10] S. Tonkens, A. Toofanian, Z. Qin, S. Gao, and S. Herbert. Patching approximately safe value functions leveraging local hamilton-jacobi reachability analysis. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pages 3577–3584. IEEE, 2024.

[11] Z. Qin, T.-W. Weng, and S. Gao. Quantifying safety of learning-based self-driving control using almost-barrier functions. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12903–12910, 2022. doi:10.1109/IROS47612.2022.9982058.

[12] S. Bansal, M. Chen, J. F. Fisac, and C. J. Tomlin. Safe Sequential Path Planning of Multi-Vehicle Systems Under Presence of Disturbances and Imperfect Information. *Proc. American Control Conference*, 2017.

[13] J. Borquez, K. Chakraborty, H. Wang, and S. Bansal. On safety and liveness filtering using hamilton-jacobi reachability analysis. *IEEE Transactions on Robotics*, 2024.

[14] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert. Robust control barrier-value functions for safety-critical control, 2021.

[15] S. Tonkens and S. Herbert. Refining control barrier functions through hamilton-jacobi reachability, 2022.

[16] 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)*, pages 3420–3431, 2019. doi:10.23919/ECC.2019.8796030.

[17] S. Bansal and C. Tomlin. DeepReach: A deep learning approach to high-dimensional reachability. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[18] K. Nakamura and S. Bansal. Online update of safety assurances using confidence-based predictions. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12765–12771. IEEE, 2023.

[19] J. Borquez, K. Nakamura, and S. Bansal. Parameter-conditioned reachable sets for updating safety assurances online. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10553–10559. IEEE, 2023.

[20] H. J. Jeong, R. Chen, and A. Bajcsy. Robots that suggest safe alternatives, 2025. URL https://arxiv.org/abs/2409.09883.

[21] A. Lin and S. Bansal. Generating formal safety assurances for high-dimensional reachability. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10525–10531. IEEE, 2023.

[22] I. Mitchell, A. Bayen, and C. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Automatic Control*, 50(7):947–957, 2005. doi:10.1109/TAC.2005.851439.

[23] L. C. Evans and P. E. Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. *Indiana Univ. Math. J.*, 33(5):773–797, 1984.

[24] J. Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40(6):917–927, 2004.

[25] E. Squires, P. Pierpaoli, and M. Egerstedt. Constructive barrier certificates with applications to fixed-wing aircraft collision avoidance. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1656–1661. IEEE, 2018. ISBN 1538676982.

[26] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020. ISBN 1728174473.

[27] C. Wang, Y. Li, Y. Meng, S. L. Smith, and J. Liu. Learning control barrier functions with high relative degree for safety-critical control. *arXiv preprint arXiv:2011.10721*, 2020.

[28] A. Taylor, A. Singletary, Y. Yue, and A. Ames. Learning for safety-critical control with control barrier functions. In *Learning for Dynamics and Control*, pages 708–717. PMLR, 2020. ISBN 2640-3498.

[29] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela. Synthesis of control barrier functions using a supervised machine learning approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7139–7145. IEEE, 2020. ISBN 1728162122.

[30] C. Dawson, Z. Qin, S. Gao, and C. Fan. Safe Nonlinear Control Using Robust Neural Lyapunov-Barrier Functions. *arXiv preprint arXiv:2109.06697*, 2021.

[31] Q. Nguyen and K. Sreenath. Optimal robust time-varying safety-critical control with application to dynamic walking on moving stepping stones. In *Dynamic Systems and Control Conference*, volume 50701, page V002T28A005. American Society of Mechanical Engineers, 2016.

[32] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33: 7462–7473, 2020.

[33] A. Lin and S. Bansal. Verification of neural reachable tubes via scenario optimization and conformal prediction. In *6th Annual Learning for Dynamics & Control Conference*, pages 719–731. PMLR, 2024.

[34] I. M. Mitchell et al. A toolbox of level set methods. *UBC Department of Computer Science Technical Report TR-2007-11*, 1:6, 2007.

[35] W. Sharpless, Z. Feng, S. Bansal, and S. Herbert. Linear supervision for nonlinear, high-dimensional neural control and differential games, 2025. URL https://arxiv.org/abs/2412.02033.

[36] A. Majumdar and M. Pavone. How should a robot assess risk? towards an axiomatic theory of risk in robotics. In N. M. Amato, G. Hager, S. Thomas, and M. Torres-Torriti, editors, *Robotics Research*, pages 75–84, Cham, 2020. Springer International Publishing. ISBN 978-3-030-28619-4.

[37] T. G. Molnar and A. D. Ames. Safety-critical control with bounded inputs via reduced order models. In *2023 American Control Conference (ACC)*, pages 1414–1421, 2023. doi:10.23919/ACC55779.2023.10155871.

[38] A. Lin, S. Peng, and S. Bansal. One filter to deploy them all: Robust safety for quadrupedal navigation in unknown environments. *arXiv preprint arXiv:2412.09989*, 2024.

# 7 Appendix

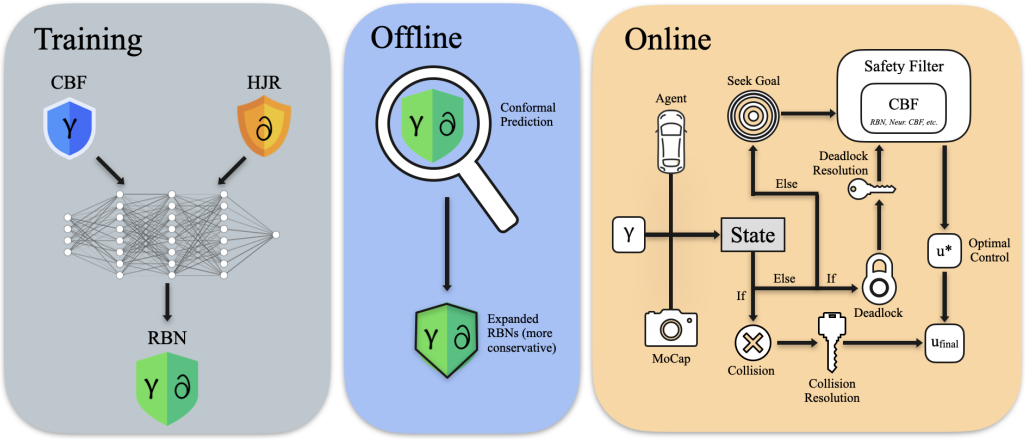


Figure 6: **Method overview.** During training, we integrate CBFs with HJR to obtain a $\gamma$-parameterized RBN. We subsequentally obtain probablistic guarantees post-training. Online, we use RBN as a safety filter, where we account for both collision resolution and deadlock detection.

Detailed explanations of the training process and conformal prediction are provided in Sections 3.1 and 3.2, respectively. For the full online control pipeline illustrated in Figure 6, we begin by obtaining the agent states from the motion capture system. The parameter $\gamma$ is appended to the state; this can be done at any point prior to passing the values to the safety filter.

We first check for collisions. If any are detected, we zero the agents' velocities and apply turning controls to separate them. Once agents are no longer in collision (i.e., at least $0.4$m apart), we restore their velocities (e.g., $0.4$m/s) and resume the nominal policy.

Following collision resolution, we apply either the nominal goal-seeking policy or deadlock-resolution control, depending on whether a deadlock is detected. These controls are then passed through the RBN-based safety filter, which enforces safety via a CBF-QP. Both the nominal and deadlock policies are described in Appendix 7.2.

## 7.1 Training Details

We list the parameters used to train the HJ pairwise solution, the neural CBF solutions, and our RBN solution, along with the domain parameters for the Dubin's systems. We keep parameters consistent across methods to ensure fairness during evaluation.

| Parameter | Value |
|---|---|
| X | $[-1, 1]$ m |
| Y | $[-1, 1]$ m |
| $\Theta$ | $[-\pi, \pi]$ rad |
| $v$ | 0.6 m/s |
| $\dot{\theta}$ $(u)$ | $[-1.1, 1.1]$ rad/s |
| Collision Radius | 0.4 m |

Table 4: System Parameters

We restrict the state domains so that the learned methods can be trained with high fidelity in a reasonable amount of time. We also limit our angular and linear velocities so that they do not exceed the maximum physical capabilities of the TurtleBots.

*HJ Pairwise*. We compute backward reachable tubes using dynamic programming over a discretized state space. For the two-player setting, we use the `Air3D` dynamics model, which captures the

| Hyperparameter | Value |
|---|---|
| Grid Size | $80 \times 80 \times 60$ |
| Solver Accuracy | High |

(a) Two-Player HJR: Grid

| Hyperparameter | Value |
|---|---|
| Hidden Layers | 3 |
| Hidden Layer Size | 512 |
| Epochs | 51 |
| Learning Rate | $5 \times 10^{-4}$ |
| CBF $\lambda$ ($\gamma$) | $[0.0, 0.5, 1.0]$ |
| Relaxation Penalty | 200 |
| Controller Period | 0.01 s |
| Learn Shape Epochs | 21 |
| Scale Parameter | 10.0 |
| Use ReLU | True |
| Trajectories / Episode | 50 |
| Trajectory Length | 300 |
| Fixed Samples | 40,000 |
| Max Points | 50,000 |
| Validation Split | 0.1 |
| Batch Size | 64 |
| Boundary Quota | 0.5 |
| Unsafe Quota | 0.4 |
| Safe Quota | 0.0 |

(b) Neural CBFs

| Hyperparameter | Value |
|---|---|
| $\gamma$ | $[0, 1]$ |
| Min With | Target |
| Sample Count | 65,000 |
| Pretrain | True |
| Pretrain Epochs | 20,000 |
| Time Range | $[0.0, 1.0]$ |
| Counter Start | 0 |
| Counter End | -1 |
| Source Samples | 1,000 |
| Target Samples | 0 |
| Activation | Sine |
| Model Mode | MLP |
| Hidden Layers | 3 |
| Hidden Layer Size | 512 |
| DeepReach Model | Exact |
| Batch Size | 1 |
| Learning Rate | $2 \times 10^{-5}$ |
| Epochs | 300,000 |
| Gradient Clipping | 0.0 |
| Adjust Relative Gradients | True |
| Dirichlet Loss Divisor | 1.0 |

(c) RBNs (DeepReach)

Table 5: Training and implementation parameters for each method.

relative state between two evaders in polar coordinates relative to the ego agent. The system is described by the state $z = (x, y, \psi)$, where $(x, y)$ is the relative position between two evaders, and $\psi$ is the relative heading. The avoid set, which represents the non-ego evader, is a circular region of radius 0.4m centered at the origin. We solve the HJ PDE on a 3D grid of size $80 \times 80 \times 60$ with periodic boundary conditions in the angular dimension. We use high-accuracy solver settings and compute the reachable tube over a time horizon of 1.0 seconds.

*Neural CBFs.* All Neural CBF hyperparameters were selected empirically after evaluation on validation rollouts. We intentionally keep the size of the network and the sampling points to be the same as our RBN method. Another key design choice was how we defined safe and unsafe regions in the state space, which implicitly defines a boundary region where transitions occur.

As mentioned in [5], a non-zero boundary is required to enable smooth transitions between the safe and unsafe sets, which helps avoid gradient discontinuities and training instability. we experimented with various settings. We chose to keep the boundary function consistent with what we use in our RBN method (i.e. min distance function). Specifically, we define the *unsafe region* as the set of states where the minimum distance between agents is less than the collision radius (0.4 m), and the *safe region* as the set where the minimum inter-agent distance exceeds the collision radius by at least 0.2 meters. The boundary region is thus the band between these two thresholds.

*RBNs*. We adapt the hyperparameters from [17]. We first pretrain the boundary condition for 20k epochs, ensuring that the terminal condition is properly learned. Then, we uniformly sample 65,000 points across the state space per epoch. We sequentially increase the time-horizon samples uniformly across epochs. To keep the PDE and boundary loss consistent, we adjust relative gradients between the two losses.

### 7.2 Hardware Implementation Details

*Deadlock*. We find that deadlock occurs in our long time-horizon multi-vehicle collision avoidance experiment for all of the CBF methods we test. The nominal control of deadlocked agents drive them towards unsafe regions, preventing them from achieving their goals. More importantly, deadlocked agents eventually exit the state distribution on which they were evaluated, leading to collisions. To accommodate, we replace the original goal-seeking nominal control of deadlocked agents with instructions to turn away from each other. We classify deadlock when two agents maintain similar headings:

$$\mathcal{D}_{N,M} = |\theta_N - \theta_M| < \delta \tag{10}$$

where $\mathcal{D}$ is the deadlock check, $N$ and $M$ is a unique pair of agents, and $\delta$ is the deadlock threshold. If $\mathcal{D}$ is true for $T$ consecutive timesteps, we consider the system to be in a deadlock. In practice, we empirically determined $T = 100$ (1 second) to consistently detect deadlock while minimally interfering with the nominal and safe controllers. As seen in Fig. 7, once deadlock is detected, we apply a corresponding deadlock resolution controller, defined as:

$$u_{\mathcal{D},N} = -k \cdot \text{sgn}\left(\cos(\theta_N)(y_M - y_N) - \sin(\theta_N)(x_M - x_N)\right) \tag{11}$$

$$u_{\mathcal{D},M} = -k \cdot \text{sgn}\left(\cos(\theta_M)(y_N - y_M) - \sin(\theta_M)(x_N - x_M)\right) \tag{12}$$

Here, $u_{\mathcal{D}}$ is shown for each agent, sgn represents a sign function, and $k$ is the magnitude of the controller. To determine the direction each agent should turn towards to resolve deadlock, we take the sign of the cross product between the heading vector and the vector between the agents. We set $k = 3$ during all experiments. We apply $u_{\mathcal{D}}$ repeatedly for 100 timesteps to ensure that the robots escape deadlock and do not fall back into deadlock again. Note that $u_{\mathcal{D}}$ is passed through the safety filter to also ensure safety during deadlock resolution. A visualization of deadlock resolution is shown in 7.

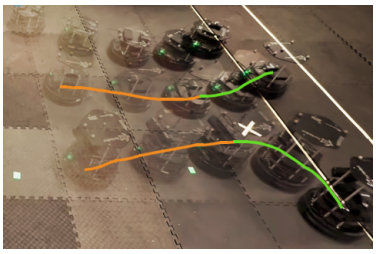

Figure 7: **Deadlock** occurrence (orange) and resolution (green).

*Nominal policy*. For our nominal policy, we correct the angle error between each agent and their respective goals. This is denoted as:

$$\theta_i^* = \arctan\left(y_i^* - y_i,\ x_i^* - x_i\right) \tag{13}$$

$$\omega_i = \arctan\left(\sin(\theta_i^* - \theta_i),\ \cos(\theta_i^* - \theta_i)\right) \tag{14}$$

where $i$ denotes each agent, $\theta_i^*$ denotes the target angle for each agent, and $\omega_i$ represents the nominal control (e.g. correction term).

*Sim2Real*. Although our simulations are based on the Dubin's car model, the physical implementation on TurtleBots is non-trivial because the Dubin's car assumes instantaneous control input actuation, which is not the case when it comes to TurtleBots. To account for delays in control actuation, we implement a closed-loop PID controller that aligns each TurtleBot's physical motion with a simulated reference trajectory generated by our policy. Specifically, the PID controller adjusts linear and angular velocities such that the robot closely follows the true Dubin's trajectory computed at each timestep.