

## 7 Limitations

**Computational scalability.** While RBNs scale better than dynamic programming, training still requires significant compute (e.g., 7 hours on an A40 GPU for the 9D system). For very high-dimensional systems, training cost could become significant. We aim to explore sampling-efficient or warmstarting approaches [35] to address this problem. Note, the neural CBF method also suffers from the same compute requirements.

**In-distribution safety.** The probabilistic guarantee generated by conformal prediction weakens as  $\gamma$  increases; we aim to address this limitation through online adaptation of RBN via a tradeoff between aggressive behavior arising from  $\gamma$  variance and safety assurances (i.e. corresponding probabilistic guarantees).

**Probabilistic assurances rely on the exchangeability assumption.** If the system encounters scenarios not well covered by the training or calibration data, safety may not be assured. Future works will include leveraging uncertainty/risk quantification methods [36] to handle out-of-distribution scenarios.

**Modeling Assumptions.** The approach assumes the system model is known (e.g., Dubin’s car) and the constraint set is also known a priori. We are excited to address this in future work via reducing order modeling [37], where disturbances in the system dynamics act as the model mismatch error.

**State Estimation Assumption.** Our current method assumes perfect state estimation through a motion capture arena. On the contrary, robots in the real world rely on imperfect state estimation. In future works, we will explore various sensors (e.g. LiDAR, RGB Camera), as well as accounting for state estimation errors via disturbance modeling [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](https://doi.org/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.

## 8 Appendix

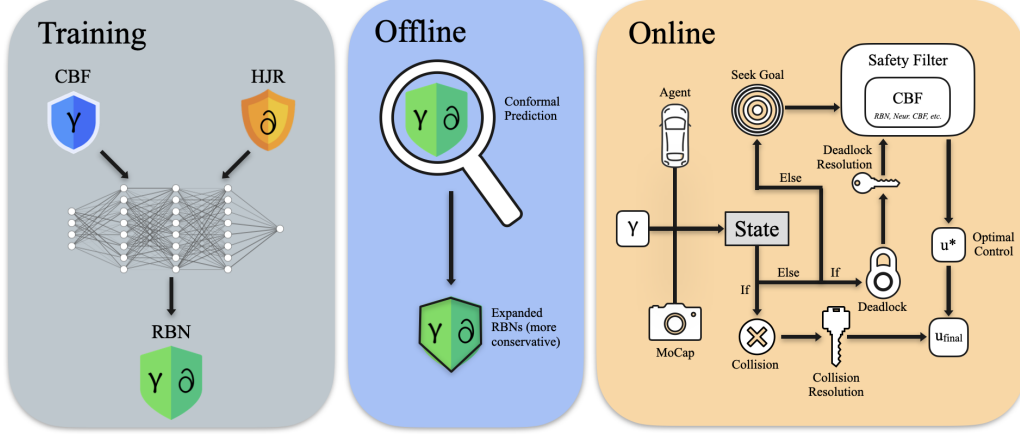


Figure 6: **Method overview.** During training, we integrate CBFs with HJR to obtain a  $\gamma$ -parameterized RBN. We subsequently obtain probabilistic guarantees post-training. Online, we use RBN as a safety filter, where we account for both collision resolution and deadlock detection.

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 0.4m apart), we restore their velocities (e.g., 0.4m/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 8.2.

### 8.1 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	Hyperparameter	Value	Hyperparameter	Value
Grid Size	$80 \times 80 \times 60$	Hidden Layers	3	$\gamma$	$[0, 1]$
Solver Accuracy	High	Hidden Layer Size	512	Min With	Target
(a) Two-Player HJR: Grid		Epochs	51	Sample Count	65,000
		Learning Rate	$5 \times 10^{-4}$	Pretrain	True
		CBF $\lambda$ ( $\gamma$ )	$[0.0, 0.5, 1.0]$	Pretrain Epochs	20,000
		Relaxation Penalty	200	Time Range	$[0.0, 1.0]$
		Controller Period	0.01 s	Counter Start	0
		Learn Shape Epochs	21	Counter End	-1
		Scale Parameter	10.0	Source Samples	1,000
		Use ReLU	True	Target Samples	0
		Trajectories / Episode	50	Activation	Sine
		Trajectory Length	300	Model Mode	MLP
		Fixed Samples	40,000	Hidden Layers	3
		Max Points	50,000	Hidden Layer Size	512
		Validation Split	0.1	DeepReach Model	Exact
		Batch Size	64	Batch Size	1
		Boundary Quota	0.5	Learning Rate	$2 \times 10^{-5}$
		Unsafe Quota	0.4	Epochs	300,000
		Safe Quota	0.0	Gradient Clipping	0.0
(b) Neural CBFs		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.

## 8.2 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:

$$\mathcal{D}_{N,M} = |\theta_N - \theta_M| < \delta \quad (11)$$

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:

$$u_{\mathcal{D},N} = -k \cdot \text{sgn}(\cos(\theta_N)(y_M - y_N) - \sin(\theta_N)(x_M - x_N)) \quad (12)$$

$$u_{\mathcal{D},M} = -k \cdot \text{sgn}(\cos(\theta_M)(y_N - y_M) - \sin(\theta_M)(x_N - x_M)) \quad (13)$$

Here,  $u_{\mathcal{D}}$  is shown for each agent,  $\text{sgn}$  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.

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

$$\theta_i^* = \arctan(y_i^* - y_i, x_i^* - x_i) \quad (14)$$

$$\omega_i = \arctan(\sin(\theta_i^* - \theta_i), \cos(\theta_i^* - \theta_i)) \quad (15)$$

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.

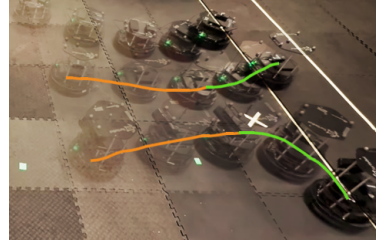


Figure 7: **Deadlock** occurrence (orange) and resolution (green).