Gradient-Free Neural Hamilton-Jacobi Reachability for Scalable Safety-Critical Control

Anonymous Author(s)
Affiliation
Address
email

Abstract: Hamilton-Jacobi (HJ) reachability provides a principled framework for synthesizing safety certificates and robust controllers for safety-critical robotic systems. However, applying reachability analysis to high-dimensional nonlinear systems remains challenging: classical grid-based solvers suffer from the curse of dimensionality, continuous-time neural solvers require accurate spatial value gradients, and reinforcement-learning-based approaches often suffer from weak boundary anchoring and non-stationary adversarial policy optimization. We propose a discrete-time neural reachability framework for control-disturbance-affine systems that learns backward reachable tubes (BRTs) and backward reach-avoid tubes (BRATs) through Bellman-Isaacs value propagation. Our key idea is to combine equation-driven self-supervision with structured policy learning: rather than computing explicit PDE-gradients, we exploit the bang-bang structure of optimal safety interventions to construct approximate teacher actions from gradient-free value probes, converting adversarial actor learning into supervised policy learning. To stabilize long-horizon value propagation, we leverage the learned actor to train the value function backward from the terminal boundary using a windowed temporal curriculum, where each window is used as the boundary condition for the next window. Across benchmark problems up to 80 dimensions, our method learns accurate reachability value functions while improving stability over existing learning-based solvers. We further demonstrate observation-space scalability on F1-tenth racing with over 16,000-dimensional egocentric inputs. The learned safety filter generalizes zero-shot to unseen tracks and transfers to a physical RC car, achieving real-time robust collision avoidance. [1]

Keywords: Robot Safety, Hamilton-Jacobi Reachability, Safety-Critical Control

1 Introduction

From autonomous vehicles to aerial robots, robotic systems are increasingly deployed in safety-critical settings. In these domains, failures can arise not only from nominal planning errors, but also from disturbances, model mismatch, and adversarial interactions. This requires control policies and safety certificates that are robust to worst-case disturbances. Hamilton-Jacobi (HJ) reachability provides a principled framework for this purpose: by solving a dynamic game between the controller and an adversarial disturbance, reachability analysis computes the set of states from which safety can be guaranteed [1], or a target can be reached while avoiding failure [2].

Despite its strong theoretical foundations, HJ reachability remains difficult to apply to high-dimensional nonlinear robotic systems. Classical grid-based methods [3, 4] solve the Hamilton-Jacobi-Isaacs (HJI) partial differential equation with strong numerical guarantees, but their computational cost grows exponentially with the state dimension [5]. To overcome this curse of dimensionality, recent works have explored learning-based approximations of HJ reachability. One prominent direction uses Physics-Informed Neural Networks (PINNs) to solve the continuous-time

[1] The code and project webpage are available anonymously at: https://anonymous.4open.science/w/grad_free_reach-0CDB/

Submitted to the 10th Conference on Robot Learning (CoRL 2026). Do not distribute.

HJI PDE by minimizing its residual [6, 7, 8, 9, 10]. The key appeal of this approach is that learning is self-supervised: the PDE itself provides a dense training signal, allowing the value function to be optimized without requiring ground-truth reachable sets. However, evaluating the HJI residual requires accurate spatial value gradients, $\nabla_x V$, throughout the state space. In long-horizon, high-dimensional, or nonsmooth reachability problems, these gradients can be difficult to represent and optimize reliably. While architectures such as SIRENs can improve derivative fidelity [6], they often introduce optimization instability and become increasingly difficult to scale in high-capacity settings [11, 12].

A second line of work formulates reachability as a discrete-time zero-sum reinforcement learning (RL) problem [13, 14, 15, 16]. This perspective is attractive because it naturally handles learned dynamics, complex simulators, and high-dimensional systems without requiring direct access to continuous-time PDE derivatives. However, RL-based reachability methods suffer from a fundamental moving-target problem: value estimates, control policies, and adversarial disturbance policies are all updated simultaneously through mutually dependent supervision signals. As the controller adapts to the current disturbance policy, the disturbance simultaneously adapts to the evolving controller, while both remain coupled to a changing value function estimate. Without a stable boundary-conditioned anchor, these recursive updates can accumulate bootstrap error over long horizons, leading to unstable optimization and inaccurate safety boundary propagation.

To address these challenges, we propose a discrete-time, windowed reachability learning framework for computing backward reachable tubes (BRTs) or backward reach-avoid tubes (BRATs). The method propagates safety values backward from the terminal boundary through a temporal curriculum, decomposing the long-horizon problem into sequential local learning stages. Within each temporal window, approximate optimal policies are constructed from the current value estimate through gradient-free bang-bang probing and refined through supervised imitation learning. The value function is then updated using Bellman-Isaacs supervision from both teacher-based one-step targets and student-policy rollout targets. To reduce long-horizon bootstrap drift, each temporal segment is further refined through rollout-anchored boundary correction before serving as the boundary condition for the preceding window. Collectively, this formulation enables scalable reachability analysis and stable long-horizon value propagation for high-dimensional systems.

Our approach can be viewed as combining the most useful aspects of PINN- and RL-based formulations while avoiding their primary failure modes. Akin to PINN-based methods, we rely on self-supervision from the governing optimality equation: instead of minimizing a continuous-time HJI PDE residual, we minimize the discrete-time Bellman-Isaacs error induced by the dynamic programming recursion. This preserves the physics-informed learning signal from PINN-style methods, but avoids their dependence on accurate spatial value gradients, which are difficult to approximate in high-dimensional, non-smooth problems. At the same time, our formulation inherits the flexibility of discrete-time RL-style methods, which can be applied to complex sampled (black-box) dynamics. However, rather than learning control and disturbance policies through coupled adversarial actor-critic updates, we exploit the bang-bang structure of the optimal policies to convert actor learning into a supervised learning problem. These supervised policies, together with the terminal boundary condition, provide stable anchors for Bellman target construction, making value propagation substantially more stable than unconstrained minimax policy-value optimization. Through several simulated case studies and real-world robotic experiments, we demonstrate that our method learns accurate BRT and BRAT value functions in high-dimensional adversarial systems, improves training stability over existing learning-based reachability solvers, and enables robust closed-loop safety-critical control under disturbances.

2 Problem Formulation

Consider a discrete-time nonlinear system with control-disturbance affine dynamics:

$$x_{k+1} = f(x_k, u_k, d_k) = f_0(x_k) + f_u(x_k)u_k + f_d(x_k)d_k, \tag{1}$$

where $x_k \in \mathcal{X}$, $u_k \in \mathcal{U} = \{u \in \mathbb{R}^{n_u} \mid \underline{u} \leq u \leq \overline{u}\}$, and $d_k \in \mathcal{D} = \{d \in \mathbb{R}^{n_d} \mid \underline{d} \leq d \leq \overline{d}\}$ denote the state, control, and disturbance at time step k, respectively. The disturbance d_k is treated

adversarially to capture worst-case safety margins, representing both exogenous environmental disturbances, such as wind gusts or adversarial agents, and internal epistemic uncertainties, such as unmodeled friction. This control-disturbance affine setting remains expressive enough to model a broad class of robotic systems, including autonomous vehicles, drones, and manipulators.

Let the failure set be $\mathcal{L} = \{x \mid \ell(x) \leq 0\}$ and the target set be $\mathcal{G} = \{x \mid g(x) \leq 0\}$. We study two closely related finite-horizon reachability problems. The first is an avoid-only safety problem, characterized by a backward reachable tube (BRT), denoted $\mathcal{B}_S(k)$. BRT contains all states at time step k from which failure is inevitable under worst-case disturbances. The second is a reach-avoid problem, characterized by a backward reach-avoid tube (BRAT), denoted $\mathcal{B}_L(k)$. BRAT consists of states from which the target can be reached while avoiding safety violations despite worst-case disturbances. Mathematically,

$$\begin{aligned} \mathcal{B}_S(k) &= \left\{ x : \forall u(\cdot), \exists d(\cdot), \exists \kappa \in [k, K], x_{x,k}^{u,d}(\kappa) \in \mathcal{L} \right\}, \\ \mathcal{B}_L(k) &= \left\{ x : \forall d(\cdot), \exists u(\cdot), \exists \kappa \in [k, K], x_{x,k}^{u,d}(\kappa) \in \mathcal{G}, \forall s \in [k, \kappa], x_{x,k}^{u,d}(s) \notin \mathcal{L} \right\}, \end{aligned} \tag{2}$$

where $x_{x,k}^{u,d}$ denotes the trajectory induced by policies $u(\cdot)$ and $d(\cdot)$ starting from state x at time step k. For brevity, the remainder of the main text focuses primarily on the BRT formulation. The full BRAT formulation and corresponding Bellman-Isaacs recursions are provided in Appendix A.

For a given task, we synthesize the relevant tube by learning the corresponding value function. In avoid-only tasks, we learn the BRT value function V_S^* and its associated safety-preserving policy. The value function is given as [17]:

$$V_S^*(x, k) = \max_{\mathbf{u}} \min_{\mathbf{d}} \left[\min_{\kappa \in [k, K]} \ell(x_\kappa) \right]. \tag{3}$$

Under this convention, $\mathcal{B}_S(k) = \{x : V_S^*(x_t, k) \leq 0\}$. The optimal value function satisfies the following discrete-time Bellman-Isaacs recursions [15]:

$$V_S^*(x, k) = \min \left(\ell(x), \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} V_S(f(x, u, d), k+1) \right), \tag{4}$$

with the boundary condition (BC) $b(x)$: $V_S^*(x, K) = \ell(x)$.

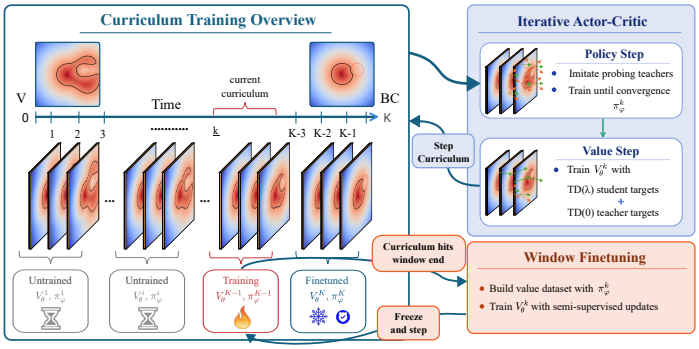
Our objective is to approximate these Bellman-Isaacs solutions with neural value functions and their corresponding optimal policies. Specifically, we aim to learn a neural value function $V_\theta(x, k)$ that approximates $V_S^*(x, k)$ thereby providing a quantitative certificate of safety or safe reachability over the state space. In parallel, we learn shared-network control and disturbance policies, $\pi_\phi^u(x, k)$ and $\pi_\phi^d(x, k)$, though our framework readily accommodates separate network architectures.

3 Method

Our method learns the neural value function associated with BRT or BRAT. Our key idea is to train backward from the terminal boundary using a windowed temporal curriculum: each window learns a local value function and corresponding control and disturbance policies through approximate teacher supervision and Bellman-error minimization, and then undergoes a finetuning phase for reducing any learning errors before being frozen as the boundary condition for the preceding window.

Temporal Partitioning via Windowed Network Architectures. Approximating long-horizon Bellman-Isaacs solutions with a single monolithic network can lead to representational overload, particularly when the value function exhibits sharp temporal variation or nonsmooth reachable-set boundaries. To reduce this burden, we partition the discrete horizon K into sequential temporal windows, each represented by a specialized network. Let $\mathcal{K} = \{0, 1, \ldots, K\}$. We define a neural value function $V_\theta : \mathcal{X} \times \mathcal{K} \to \mathbb{R}$ together with a shared policy network $\pi_\phi^{u,d} : \mathcal{X} \times \mathcal{K} \to \mathcal{U} \times \mathcal{D}$ that parameterizes both the control and disturbance policies. The value function is boundary-aware by construction at the terminal time $k = K$ and uses independent network weights within each temporal window:

$$V_\theta(x, k) = \begin{cases} b(x), & \text{if } k = K, \\ V_\theta^w(x, k), & \text{if } K - wW \leq k < K - (w-1)W, \end{cases} \tag{5}$$



Algorithm 1 Overview

Require: K, g, ℓ, W
  $\underline{k} \leftarrow K$, initialize NNs
  while $\underline{k} \geq 0$ **do**
    $\pi \leftarrow \text{Supervise}(\hat{u}_{\text{teacher}}^{\underline{k}:K})$
    $V \leftarrow \text{Update}(y_{\text{mixed}}^{\underline{k}:K})$
    if $(K - \underline{k}) \bmod W = 0$ **then**
      FinetunePhase
      Freeze current window
      Unfreeze next window
    $\underline{k} \leftarrow \underline{k} - 1$

Figure 1: Method Overview: backward temporal curriculum with actor-critic learning.

where $W \in \mathbb{Z}^+$ denotes the window length, $w \in \mathbb{Z}^+$ is the window index counted backward from the terminal time, and $b(x)$ is the terminal boundary condition.

The shared policy parameterization is decomposed into temporal windows: $\pi_\phi^{(\cdot)}(x,k) = \pi_\phi^{(\cdot),w}(x,k)$ for $K - wW \leq k < K - (w-1)W$, where $(\cdot) \in \{u, d\}$ denotes the control and disturbance policies. This windowed representation decomposes the long-horizon dynamic programming problem into shorter-horizon subproblems. Each network focuses its representational capacity on a localized temporal segment, while later-time windows, once trained and frozen, serve as fixed boundary conditions for earlier windows.

Backward Curriculum. As illustrated in Alg. 1 and Fig. **??**, training proceeds backward along the discrete time axis, starting from the terminal boundary condition. At each curriculum step, the algorithm alternates between policy training and value training, followed by a window-finetuning phase whenever a temporal window is completed. Each phase is trained to near convergence before the curriculum advances. Crucially, the policy update is performed before the value update to ensure that the updated policies are available to construct accurate temporal-difference (TD) targets for value learning. Whenever the elapsed curriculum horizon $(K - \underline{k})$ reaches a multiple of the window duration W, the algorithm triggers a finetuning phase to calibrate the value network against rollout-derived targets, effectively aligning the policies and the value function. After fine-tuning, the parameters of the completed window, including both value and policy networks, are frozen and used as the boundary condition for the preceding window. We now detail each of these phases.

3.1 Policy Training via Bang-Bang Probing

The policy training phase learns control and disturbance policies that approximate the optimal actions in the Bellman-Isaacs recursion in Eqn. 3. To mitigate the "moving target" problem prevalent in joint control-disturbance optimization (a minimax update), we construct approximate teacher actions from the current value estimate and use them to supervise the student policy networks:

$$\mathcal{L}_\pi = \frac{1}{N} \sum_{i=1}^{N} w_i \left(\left\| \pi_\phi^u(x_i, k_i) - \hat{u}^*(x_i, k_i) \right\|_2^2 + \left\| \pi_\phi^d(x_i, k_i) - \hat{d}^*(x_i, k_i) \right\|_2^2 \right), \tag{6}$$

where $x_i \in \mathcal{X}$ are sampled states, $k_i \in \mathcal{K}_{\text{curr}} = \{\underline{k}, \underline{k}+1, \ldots, K-(w-1)W-1\}$ are time indices, $(\hat{u}^*, \hat{d}^*)$ are the estimated teacher actions, and w_i weights the confidence of each teacher label.

Estimating Teacher Policies. Our teacher construction is inspired by the bang-bang structure of optimal policies in continuous-time Hamilton-Jacobi reachability [17, 18]. For continuous-time control-disturbance-affine dynamics $\dot{x} = f_c(x,u,d) = f_{c,0}(x) + f_{c,u}(x)u + f_{c,d}(x)d$, the continuous-time optimal actions are strictly bang-bang:

$$\hat{u}_j^* = \begin{cases} \overline{u}_j, & C_{u,j} > 0, \\ \underline{u}_j, & C_{u,j} \leq 0, \end{cases} \qquad \hat{d}_j^* = \begin{cases} \underline{d}_j, & C_{d,j} > 0, \\ \overline{d}_j, & C_{d,j} \leq 0, \end{cases} \tag{7}$$

where $C_{u,j}$ and $C_{d,j}$ are the j-th components of the gradient projections $f_{c,u}(x)^\top \nabla_x V$ and $f_{c,d}(x)^\top \nabla_x V$, respectively. For the opposite minimization-maximization convention (e.g., in BRAT), the endpoint assignments are reversed accordingly.

Unlike PINN-based reachability solvers, our method does not require analytic spatial gradients $\nabla_x V_\theta$ for policy extraction. Instead, we infer the relevant gradient-projection signs through discrete value comparisons of forward-simulated next states. For each control dimension j, we evaluate $\Delta V_j^u := V_\theta(x^{\overline{u}_j}, k+1) - V_\theta(x^{\underline{u}_j}, k+1) \approx \Delta t \nabla_x V_\theta(x, k+1)^\top f_{c,u}(x)\big(\overline{u}_j - \underline{u}_j\big) = \Delta t\, C_{u,j}\big(\overline{u}_j - \underline{u}_j\big)$, where $x^{\overline{u}_j}$ and $x^{\underline{u}_j}$ denote the next discrete-time states obtained by setting only the j-th control dimension to its upper or lower bound, while holding the remaining action dimensions fixed at a baseline value. Since $\overline{u}_j - \underline{u}_j > 0$, the sign of ΔV_j^u approximates the sign of $C_{u,j}$ for sufficiently small discretization step Δt. An analogous probing procedure yields $\operatorname{sign}(C_{d,j}) \approx \operatorname{sign}(\Delta V_j^d)$ for each disturbance dimension.

This procedure requires only $2(n_u + n_d)$ forward value evaluations per sampled state and provides an efficient, gradient-free approximation of the teacher actions $(\hat{u}^*, \hat{d}^*)$. To reduce the influence of ambiguous labels, such as regions where the value is locally flat or multiple actions yield nearly identical next-step values (e.g., $\nabla V_x \to \mathbf{0}$), we downweight uncertain teachers through the confidence weight w_i, computed as the inverse variance of these directional probe scores.

3.2 Value Training via Mixed Temporal-Difference Targets

A fundamental challenge in this decoupled actor-critic scheme is compounding suboptimality: if a suboptimal student policy is used to train the value network, the resulting value estimate may become biased, which in turn yields corrupted teacher actions in subsequent policy updates. To mitigate this error propagation, we train the value function using a mixture of one-step TD targets generated from teacher actions and multi-step TD targets generated from student-policy rollouts. The teacher targets keep the value update anchored to the Bellman-Isaacs recursion to preserve optimality, while the student targets improve consistency with the learned policies over longer rollouts.

For a sampled state-time pair (x_0, k_0), we first compute a one-step teacher target. Let $x_1 = f(x_0, \hat{u}^*, \hat{d}^*)$ be the next state under the teacher actions. The teacher TD target is then $y_{\text{teacher}} = \mathcal{J}\big(x_0, V_\theta(x_1, k_0+1)\big)$, where $\mathcal{J}(x, V) = \min\big(\ell(x), V\big)$ for BRT computations.

We also construct a multi-step student target by rolling out the learned policies (π_ϕ^u, π_ϕ^d) for M steps, where M is sampled from a geometrically decaying distribution. Let $x_{0:M}$ denote the resulting trajectory and let $V_\theta(x_M, k_0+M)$ be the bootstrapped terminal value. The student target is $y_{\text{student}} = \mathcal{J}\big(x_{0:M}, V_\theta(x_M, k_0+M)\big)$. For BRT, this reduces to the minimum safety margin along the rollout, clipped by the bootstrapped value at the terminal rollout state.

The value network are trained to minimize the MSE residual of V_θ against the evenly mixed targets (i.e., 50% student samples). To improve boundary-condition compliance, we oversample time indices near the current boundary, $k_0 = K - (w-1)W - 1$, when drawing $k_0 \in \mathcal{K}_{\text{curr}}$.

3.3 Finetune Phase

Although mixed TD targets reduce short-horizon suboptimality, the backward windowed curriculum remains susceptible to bootstrap drift across temporal boundaries. If a completed window is frozen with biased value estimates, those errors become the boundary condition for the preceding window and can propagate backward through the horizon. To reduce this drift, we perform a boundary-correction fine-tuning phase at the end of each temporal window before freezing its parameters.

This phase calibrates the value network against rollout-derived targets under the *learned* student policies. Specifically, we optimize a semi-supervised objective that combines local Bellman consistency with supervised anchor targets obtained from discrete-time dynamics rollouts:

$$\mathcal{L}_{\text{FT}} = \mathcal{L}_{\text{TD}}^{\text{student}} + \lambda_{\text{sup}}\mathcal{L}_{\text{sup}} = \tfrac{1}{B}\left(\textstyle\sum_{i=1}^{B}\left(V_\theta(x_i, k_i) - y_{\text{student},i}\right)^2 + \lambda_{\text{sup}}\sum_{m=1}^{B} w_m\left(V_\theta(x_m, k_m) - G_m\right)^2\right). \tag{8}$$

Here, $\mathcal{L}_{\text{TD}}^{\text{student}}$ uses only student-policy TD targets, omitting teacher targets so that the update aligns the value network with the policies being evaluated. The second term anchors the value network to empirical rollout targets G_m, sampled from a static dataset. To construct the G_m

Table 1: Quantitative comparison of open-loop safety and closed-loop control performance. Results report mean and standard deviation over 5 random seeds.

Method	Narrow Passage				Quadrotor Gate Avoidance			
	FPR % ↓	FNR % ↓	Succ % ↑	Time (h) ↓	FPR % ↓	FNR % ↓	Succ % ↑	Time (h) ↓
DeepReachMPC	$\mathbf{0.00 \pm 0.00}$	100.0 ± 0.00	2.50 ± 0.90	5.0	2.0 ± 0.8	78.40 ± 14.00	0.06 ± 0.00	**4.0**
RL Baseline	$\mathbf{0.00 \pm 0.00}$	93.40 ± 0.70	3.70 ± 0.72	5.5	30.38 ± 3.39	$\mathbf{0.00 \pm 0.00}$	0.00 ± 0.00	5.5
Ours	100.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	$\mathbf{99.65 \pm 0.31}$	**4.0**	$\mathbf{0.5 \pm 0.0}$	9.70 ± 1.00	$\mathbf{0.50 \pm 0.01}$	5.5

Table 2: Scalability stress-testing on high-dimensional Publisher-Subscriber network dynamics.

Method	40D Network Dynamics			80D Network Dynamics		
	FPR % ↓	FNR % ↓	MSE ↓	FPR % ↓	FNR % ↓	MSE ↓
DeepReachMPC	2.648 ± 0.310	$\mathbf{0.001 \pm 0.001}$	$\mathbf{0.007 \pm 0.001}$	2.115 ± 0.145	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.026 \pm 0.001}$
RL Baseline	23.306 ± 0.397	76.228 ± 0.602	1.717 ± 0.086	25.581 ± 0.941	78.696 ± 0.731	5.886 ± 0.24783
Ours	$\mathbf{0.000 \pm 0.000}$	9.134 ± 0.693	0.009 ± 0.002	$\mathbf{0.219 \pm 0.429}$	14.263 ± 1.884	0.072 ± 0.016

dataset, we forward-simulate from (x_m, k_m) using the learned student policies until reaching the upper boundary of the active window, $k_{\text{bound}} = K - (w-1)W$, where $w \geq 1$ is the current window index. The anchor target is computed from the same stage-wise reachability cost along this rollout and bootstrapped with the frozen value function at k_{bound}: $G^{\text{BRT}}(x,k) = \mathcal{J}\big(x_{k:k_{\text{bound}}}, V_\theta(x_{k_{\text{bound}}}, k_{\text{bound}})\big) = \min\left[\min_{\kappa\in[k,k_{\text{bound}}]} \ell(x_\kappa),\ V_\theta\left(x_{k_{\text{bound}}}, k_{\text{bound}}\right)\right]$. The first term evaluates whether the target is reached safely within the active window, while the second term bootstraps from the frozen boundary value. To discourage false-positive safety predictions, we apply asymmetric weights $w_m = 1 + \lambda_{\text{fp}} \mathbb{1}\{V_\theta(x_m, k_m) > 0 \text{ and } G_m < 0\}$, where λ_{fp} controls the penalty for states predicted to be safe but evaluated as unsafe by rollout. The fine-tuning phase is run with a reduced learning rate until convergence. The active window is then frozen and used as the boundary condition for the preceding window.

4 Experimental Results

We evaluate our framework across three benchmarks spanning distinct types of reachability problems: (i) **Two-Vehicle Narrow Passage** [6] (Reach-Avoid); (ii) **Quadrotor Gate Avoidance** (Avoid); and (iii) **40D/80D Publisher-Subscriber** [19] (Reach). We compare against three state-of-the-art baselines: **DeepReachMPC** [7] (a monolithic continuous-time PINN-based solver), **RARL** [14] (an RL-based solver for BRAT), and **ISAACS** [15] (an actor-critic RL baseline for BRT). Detailed training configurations and environment setups are deferred to Appendix C.

Evaluation metrics. We track the **False Positive Rate (FPR)** (predicted safe but empirical failure, indicating dangerous over-optimism) and **False Negative Rate (FNR)** (predicted unsafe but empirical success, indicating over-conservatism) of the value function. We further report **Success Rate** to assess policy optimality, and **Training Time** to evaluate computational efficiency. Each metric is evaluated over 100K samples across 5 random seeds.

Long-Horizon Narrow Passage. Two opposing vehicles are tasked with safely reaching their respective goals through a bottleneck caused by a stranded vehicle. To avoid a head-on collision, one agent must proactively yield early in the extended $T = 8.0$s horizon. This requires the reachability solver to robustly maintain the early-time optimal value function, preventing agents from adopting myopic maneuvers that greedily approach the goal at the cost of mild safety violations. As shown in Table 1, our method successfully learns these far-sighted coordination strategies by leveraging temporal partitioning to preserve short-horizon intermediate solutions required for safe traversal. Notably, because all evaluated initial states are inherently safe (i.e., ideal success is 100%) and our learned value function correctly predicts them as such, any empirical collision resulting from minor policy execution errors mathematically forces a 100% FPR. Hence, the reported FPR reflects policy execution flaws over the long horizon, rather than a severely over-optimistic value function.

13D Quadrotor Gate Avoidance. A quadrotor must safely navigate through a gate opening within an otherwise infinitely expansive y-z planar obstacle. The vehicle must also maintain a positive x-velocity while adhering to linear and angular velocity bounds across all axes. The system's high thrust-to-mass ratio of 20, combined with the non-smooth safety boundaries, introduces severe numerical instability for standard learning-based solvers, especially those utilizing explicit gradient

Table 3: Ablation of policy optimization objectives on the Narrow Passage benchmark.

Configuration	FPR % ↓	FNR % ↓	Succ % ↑	Time (mins) ↓
Policy Gradient - 1 Step	99.58 ± 0.09	0.79 ± 0.74	92.20 ± 6.11	265.5
Policy Gradient - 3 Step	$\mathbf{93.57 \pm 8.71}$	0.25 ± 0.13	90.15 ± 6.87	307.4
Policy Gradient - 5 Step	99.75 ± 0.09	0.28 ± 0.83	80.81 ± 3.85	400.1
Teacher Student	100.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	$\mathbf{99.65 \pm 0.31}$	$\mathbf{262.4}$

representations during training. Conversely, our gradient-free approach leverages the stable anchors provided by the temporal curriculum and empirical rollouts to robustly propagate the value function. Crucially, while our absolute 0.50% success rate appears low, it reflects the physical reality of the environment: the vast majority of random initial states are dynamically doomed to violate the z-velocity constraint due to gravity. We further validate the optimality of our learned policy by outperforming both the built-in sampling-based MPC solver of DeepReachMPC (0.03%) and a carefully tuned MPPI controller (0.21%).

Publisher-Subscriber Systems. This benchmark stress-tests scalability against massive state and action spaces, where the control dimension is exactly the state dimension minus one (representing a single uncontrollable publisher and $N-1$ controllable subscribers). Because the system dynamics are structurally decomposable, it admits a ground-truth solution for the MSE evaluation. As shown in Table 2, our framework—despite being formulated as a generalized solver for two-player zero-sum games—achieves a MSE on par with DeepReachMPC, which is engineered for single-player optimal control. This demonstrates that our method scales gracefully to high-dimensional domains.

Ablation on Policy Optimization. We isolate and validate our critical architectural choices using the long-horizon Narrow Passage dynamics as a benchmark. Specifically, we compare our value-grounded teacher-student objective against policy gradient (PG) alternatives, which update the policy networks using TD targets derived from policy rollouts whose lengths are sampled from a geometric distribution between 1 and m steps. As shown in Table 3, the teacher-student formulation achieves substantially faster convergence and improved consistency across random seeds. By deriving supervision directly from the extremal-action decisions induced by V_θ (Eqn. 6), our approach is significantly more robust to the sparse gradient signals inherent to reachability (where trajectory cost depends solely on the extremum state, e.g., $\arg\min_x \ell(x)$). Additionally, the teacher-student paradigm eliminates the need to tune the maximum rollout horizon (m), a hyperparameter that heavily dictates the empirical stability of PG-based training. This highlights that our framework maintains the architectural benefits of actor-critic methods while explicitly avoiding the unstable actor-training dynamics inherent to traditional RL.

Additional ablations evaluating the teacher-mixing ratio, alongside benchmarks on two-player zero-sum games (e.g., pursuer-evader problems), are deferred to Appendix E.1 due to space constraints.

4.1 Vision-Based F1Tenth Racing and Sim-to-Real Deployment

We scale our framework to synthesize a safety value function (BRT) for high-speed autonomous racing (max speed 10 m/s), where the vehicle must remain collision-free against the track boundaries with acceleration and steering rate as control inputs. We train the reachability solution jointly across 20 tracks and evaluate it on 3 held-out tracks from [20], each spanning roughly 120 m $\times$ 120 m. The model takes as input an egocentric 128×128 Bird's-Eye View (BEV) binary occupancy image concatenated with the vehicle's proprioceptive state (speed, angular velocity, slip angle, steering angle) and a disturbance scale parameter $s_d \in [0.0, 0.3]$. Crucially, the BEV spatial resolution dynamically scales with vehicle speed to guarantee sufficient look-ahead visibility for accurate safety predictions. We note that commonly used low-dimensional inputs, such as LiDAR scans, lack sufficient information for accurate safety prediction in a racing context (e.g., due to occlusion near sharp turns). Because the BEV pixel dynamics are unknown, we treat the system as a black box with underlying control-disturbance-affine dynamics. To ensure robustness against OOD observations and modeling errors, we train against adversarial disturbances including lateral drift and control authority degradation. At test time, the learned value function is equipped with a Discrete-Time Control

Barrier Function (DCBF) filter [21] to safeguard a poorly-tuned MPPI nominal controller. Across 50 evaluation trials (40s each), the filtered policy achieved zero collisions on both the in-distribution and OOD tracks under a disturbance scale $s_d = 0.1$, with an average speed of 9.25 and 9.17 m/s, respectively. Without the filter, the MPPI baseline yields 56% and 48% collision rates, respectively. We further finetune our value networks on 20 scaled-down tracks to match real-world dimensions and deploy them onto a physical Rosmaster R2 RC car on a previously unseen track. Following a low-speed warmup lap to construct a track map, we utilize Vicon motion capture for real-time localization, as the 7 Hz onboard LiDAR suffers from severe localization drift at higher speeds. All computations—including the nominal controller, BEV image generation, and safety filtering—execute on an offboard laptop with a GTX 1060 GPU and a 12-core 3.3 GHz CPU. The laptop streams linear and angular reference velocities to the car over WiFi. This closed-loop deployment introduces moderate sim-to-real challenges: a mismatch between the hardware and training control spaces, OOD track geometries, and approximately 50 ms of combined computation and communication latency. By applying a disturbance scale of 0.2, our adversarial formulation enables the safety filter to robustly absorb these modeling errors and discrepancies. Operating at speeds up to 1.5 m/s (hardware physical limit), the filtered policy completed 10 laps with zero collisions and an average speed of 1.07 m/s. In stark contrast, the unfiltered nominal policy, which blindly tracks the racing line with maximal speed, led to 3 collisions within its very first lap. More details are available in Appendix F.

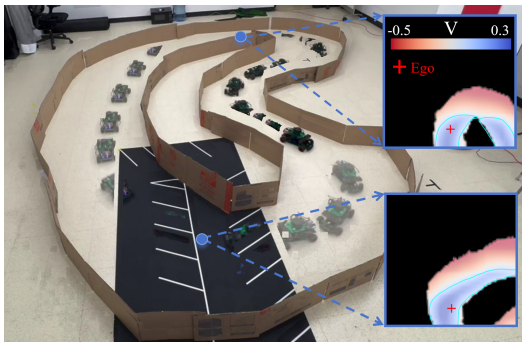


Figure 2: Real-world deployment on the Rosmaster R2 RC car, with value function slices visualized at critical states along the trajectory.

5 Conclusion

In this work, we introduced a discrete-time, gradient-free neural framework for high-dimensional Hamilton-Jacobi (HJ) reachability. Our method partitions long-horizon problems into temporal windows, adapting a teacher-student actor-critic framework to jointly approximate optimal policies and value functions. By inheriting the backward temporal curriculum from PINNs, we preserve stable boundary anchoring while maintaining the flexibility of discrete-time RL. Across diverse safety-critical benchmarks, our framework excels at solving long-horizon and numerically stiff reachability problems. Moreover, it scales to massive state, control, and observation dimensions, and achieves robust sim-to-real generalization through closed-loop physical racing.

6 Limitations and Future Work

While our discrete-time framework scales robustly to high-dimensional systems and vision-based observations, several avenues for refinement remain. First, our method is sensitive to the integration time-step (see Appendix D.2). Proper tuning is critical: excessively large steps introduce significant discretization error, while overly small steps shrink the directional value differences and induce biased policies. For simple reachability problems with closed-form dynamics, DeepReachMPC generally requires less tuning to provide strong baseline performance. Consequently, a valuable future direction is the empirical quantification of optimal Δt bounds or the development of adaptive time-stepping during the temporal curriculum.

Second, because our bang-bang action probing assumes affine dynamics, the framework provides suboptimal solutions for non-affine black-box systems, where RL-based methods theoretically approximate optimal solutions. This bang-bang nature may also struggle with tasks requiring soft and smooth control interactions. An exciting future direction is to replace single-step bang-bang teachers with multi-step episodic rollouts to relax the affine dynamics requirement—at the cost of higher variance in the learning signal—enabling the framework to solve reachability problems that demand more nuanced control, such as humanoid balancing or intricate manipulation tasks.

References

[1] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50 (7):947–957, 2005.

[2] K. Margellos and J. Lygeros. Hamilton-Jacobi Formulation for Reach–Avoid Differential Games. *IEEE Transactions on Automatic Control*, 56(8):1849–1861, 2011.

[3] I. Mitchell. A toolbox of level set methods. *http://www. cs. ubc. ca/mitchell/ToolboxLS/toolboxLS.pdf*, 2004.

[4] M. Bui, G. Giovanis, M. Chen, and A. Shriraman. Optimizeddp: An efficient, user-friendly library for optimal control and dynamic programming. *arXiv preprint arXiv:2204.05520*, 2022.

[5] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253, 2017. doi:10.1109/CDC.2017.8263977.

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

[7] Z. Feng, L. Qiu, and S. Bansal. Bridging model predictive control and deep learning for scalable reachability analysis. *Robotics: Science and Systems (RSS)*, 2025.

[8] A. Singh, Z. Feng, and S. Bansal. Exact imposition of safety boundary conditions in neural reachable tubes. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5489–5495, 2025. doi:10.1109/ICRA55743.2025.11127972.

[9] V. K. Chilakamarri, Z. Feng, and S. Bansal. Reachability analysis for black-box dynamical systems. *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[10] W. Sharpless, Z. Feng, S. Bansal, and S. Herbert. Linear supervision for nonlinear, high-dimensional neural control and differential games. *arXiv preprint arXiv:2412.02033*, 2024.

[11] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[12] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *arXiv preprint arXiv:2006.09661*, 2020.

[13] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.

[14] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac. Safety and liveness guarantees through reach-avoid reinforcement learning. *arXiv preprint arXiv:2112.12288*, 2021.

[15] K.-C. Hsu, D. P. Nguyen, and J. F. Fisac. Isaacs: Iterative soft adversarial actor-critic for safety. In *Learning for Dynamics and Control Conference*, pages 90–103. PMLR, 2023.

[16] J. Li, D. Lee, J. Lee, K. S. Dong, S. Sojoudi, and C. Tomlin. Certifiable deep learning for reachability using a new lipschitz continuous value function, 2025. URL `https://arxiv.org/abs/2408.07866`.

[17] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-Jacobi Reachability: A brief overview and recent advances. In *IEEE Conference on Decision and Control (CDC)*, 2017.

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

[19] W. Sharpless, Z. Feng, S. Bansal, and S. Herbert. Linear supervision for nonlinear, high-dimensional neural control and differential games. In N. Ozay, L. Balzano, D. Panagou, and A. Abate, editors, *Proceedings of the 7th Annual Learning for Dynamics & Control Conference*, volume 283 of *Proceedings of Machine Learning Research*, pages 365–377. PMLR, 04–06 Jun 2025. URL https://proceedings.mlr.press/v283/sharpless25a.html.

[20] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam. Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open J. Intell. Transp. Syst.*, 3:458–488, 2022. doi:10.1109/ojits.2022.3181510.

[21] A. Agrawal and K. Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *RSS*, 2017.

[22] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, 2016. doi:10.1109/ICRA.2016.7487277.

[23] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam. F1tenth: An open-source evaluation environment for continuous control and reinforcement learning. In H. J. Escalante and R. Hadsell, editors, *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, volume 123 of *Proceedings of Machine Learning Research*, pages 77–89. PMLR, 08–14 Dec 2020. URL https://proceedings.mlr.press/v123/o-kelly20a.html.

[24] A. Lin and S. Bansal. Verification of neural reachable tubes via scenario optimization and conformal prediction. *6th Annual Learning for Dynamics & Control Conference (L4DC)*, 2024.

A BRAT Derivations and Descriptions

As stated in the main text, our proposed discrete-time neural reachability framework generalizes naturally to tasks that require simultaneous goal-reaching and safety tracking. In this appendix, we present the Backward Reach-Avoid Tube (BRAT) formulation in details, including the corresponding Bellman-Isaacs recursion operators, the TD target, and the rollout reachability anchoring labels utilized during fine-tuning phase.

A.1 Reach-Avoid Value Function and Recursion

Let the failure set be characterized by the zero sub-level set $\mathcal{L} = \{x \mid \ell(x) \leq 0\}$ and the target set by $\mathcal{G} = \{x \mid g(x) \leq 0\}$. The backward reach-avoid tube $\mathcal{B}_L(k)$ represents the set of all initial states at time step k from which a controller can guarantee that the system enters the target set $\mathcal{G}$ within the finite horizon while strictly remaining outside the failure set $\mathcal{L}$ under worst-case disturbances:

$$\mathcal{B}_L(k) = \left\{ x \;\middle|\; \forall d(\cdot),\, \exists u(\cdot),\, \exists \kappa \in [k, K] \text{ s.t. } x_{x,k}^{u,d}(\kappa) \in \mathcal{G}, \text{ and } \forall s \in [k, \kappa],\, x_{x,k}^{u,d}(s) \notin \mathcal{L} \right\}. \quad (9)$$

By adopting a minimization-maximization convention over the target distance and historical safety margins, the optimal finite-horizon BRAT value function $V_L^*(x, k)$ is characterized as:

$$V_L^*(x,k) = \min_{\mathbf{u}} \max_{\mathbf{d}} \left[\min_{\kappa \in [k,K]} \left(\max \left(g(x_\kappa),\, \max_{s \in [k,\kappa]} -\ell(x_s) \right) \right) \right], \quad (10)$$

where the underlying reach-avoid tube is recovered at the zero sub-level set $\mathcal{B}_L(k) = \{x \mid V_L^*(x, k) \leq 0\}$. The solution satisfies the following discrete-time Bellman-Isaacs recursion [15]:

$$V_L(x,k) = \max \left(-\ell(x), \min \left(g(x), \min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} V_L(f(x,u,d), k+1) \right) \right), \quad (11)$$

subject to the joint terminal boundary condition (BC):

$$V_L(x, K) = \max \big(g(x), -\ell(x) \big). \quad (12)$$

A.2 Policy Extraction and Teacher Adjustments

To approximate the optimal minimax action pairs without explicit spatial gradients $\nabla_x V_\theta$, Our method simply adopts the same bang-bang probing strategy, except with mirrored signs due to the reversed min-max assignment of the control and disturbance. Specifically,

$$\hat{u}_j^* = \begin{cases} \overline{u}_j, & C_{u,j} \leq 0, \\ \underline{u}_j, & C_{u,j} > 0, \end{cases} \qquad \hat{d}_j^* = \begin{cases} \underline{d}_j, & C_{d,j} \leq 0, \\ \overline{d}_j, & C_{d,j} > 0, \end{cases} \quad (13)$$

where the sign of C's can be similarly estimated by the sign of $\Delta V_j^u := V_\theta(x^{\overline{u}_j}, k+1) - V_\theta(x^{\underline{u}_j}, k+1)$. The same procedure provides $\text{sign}(C_{d,j}) \approx \text{sign}(\Delta V_j^d)$. Although this formulation technically breaks the non-anticipative strategy assumption for the disturbance in a strict discrete-time setting, Appendix E.1 empirically shows that for small Δt, our method remains on par with numerical solvers in two-player games.

A.3 TD and Fine-Tuning Anchor Targets

Let the one-step backward reach-avoid operator be defined as:

$$\mathcal{J}_{\text{BRAT}}(x, V) = \max \left(-\ell(x), \min \left(g(x), V \right) \right). \quad (14)$$

For a sampled state-time pair (x_0, k_0), the teacher TD target is given by $y_{\text{teacher}} = \mathcal{J}_{\text{BRAT}}\big(x_0, V_\theta(x_1, k_0 + 1)\big)$, where $x_1 = f(x_0, \hat{u}^*, \hat{d}^*)$.

The M-step student target, y_{student}, evaluates the nested maximum reach-avoid cost under the learned student policies $\pi_\phi^{u,d}$, clipped by the bootstrapped value function at the final step:

$$y_{\text{student}} = \max \left[\min_{\kappa \in [k_0, k_0+M]} \left(\max \left(g(x_\kappa),\, \max_{s \in [k_0,\kappa]} -\ell(x_s) \right) \right),\; V_\theta(x_M, k_0 + M) \right]. \quad (15)$$

When the curriculum completes a temporal window and triggers the boundary-correction fine-tuning phase, we construct empirical anchor targets, G^{BRAT}. Instead of an M-step horizon, we evaluate the rollout until the current window's temporal boundary, k_{bound}, and bootstrap from the frozen downstream value function:

$$G^{\text{BRAT}} = \max\left[\min_{\kappa\in[k_m,k_{\text{bound}}]}\left(\max\left(g(x_\kappa), \max_{s\in[k_m,\kappa]} -\ell(x_s)\right)\right),\ V_{\theta_{\text{frozen}}}\left(x_{k_{\text{bound}}}, k_{\text{bound}}\right)\right]. \quad (16)$$

To penalize over-optimistic value predictions (false positives), we adapt the asymmetric penalty w_m in Eqn. (8) by mirroring the indicator conditions: $w_m = 1 + \lambda_{\text{fp}}\mathbb{1}V_\theta(x_m, k_m) \le 0$ and $G_m > 0$.

Upon convergence of this objective, the network parameters are frozen to serve as a verified boundary condition for the preceding temporal segment.

B Additional Method Details

This section presents supplementary methodological details that are omitted from the main text for brevity.

B.1 Sub-Step Safety Evaluation and Tie-Breaking

While our framework explicitly accounts for discrete-time safety, violations can still occur between discrete time steps. To mitigate this, one can roll out the transition using a finer time resolution to detect safety violations during these intermediate sub-steps. To incorporate this sub-step safety verification into the probing teacher, we determine the control actions by evaluating the sign of ΔJ_j^u instead of the standard value difference ΔV_j^u (which corresponds to the sign of $C_{u,j}$ in Eqn. 13). Here, J_j^u evaluates the worst-case safety margin during the transition, defined as the minimum between the safety function $\ell(x)$ evaluated at the intermediate transition states x_{sub}, and the network prediction at the subsequent discrete state $V_\theta(x^{u_j})$:

$$J_j^u = \min\left(\min_{x_{\text{sub}}} \ell(x_{\text{sub}}), V_\theta(x^{u_j})\right).$$

While this formulation more strictly enforces safety during intermediate transitions, there are substantial regions of the state space where ΔJ_j^u evaluates exactly to zero, because HJ reachability is inherently a sparse reward problem. For instance, ΔJ_j^u will be the safety margin at the very first sub step when the robot is already moving safely away from the failure set. In these regions, the selection rule in Eqn. 13 blindly defaults to the lower bound $\underline{u}_j$. This introduces an artificial bias into the learned optimal policies, as the ideal behavior should instead seek to transition to a next state that maximizes the overall safety value.

To account for this, we introduce a value-based tie-breaking mechanism. When the coefficient is estimated as exactly zero, we explicitly evaluate the resulting next-state values to determine the optimal action. Again, let $x^{\overline{u}_j}$ and $x^{\underline{u}_j}$ denote the next discrete-time states obtained by setting only the j-th control dimension to its upper or lower bound, respectively, while holding the remaining action dimensions fixed at their mean values. The updated probing teachers are as follows:

$$\hat{u}_j^* = \begin{cases} \overline{u}_j, & \Delta J_j^u > 0, \\ \underline{u}_j, & \Delta J_j^u < 0, \\ \arg\max_{u_j\in\{\underline{u}_j,\overline{u}_j\}} V_\theta(x^{u_j}), & \Delta J_j^u = 0, \end{cases} \qquad \hat{d}_j^* = \begin{cases} \underline{d}_j, & \Delta J_j^d > 0, \\ \overline{d}_j, & \Delta J_j^d < 0, \\ \arg\min_{d_j\in\{\underline{d}_j,\overline{d}_j\}} V_\theta(x^{d_j}), & \Delta J_j^d = 0. \end{cases} \quad (17)$$

B.2 Triple-Sided Probing

While bang-bang control is theoretically optimal for continuous-time, control-disturbance-affine systems, it can introduce chattering behavior in discrete-time implementations. For instance, a quadrotor navigating along the centerline of a narrow gate might repeatedly oscillate between maximum and minimum thrust, resulting in a chattery and inefficient control profile.

To mitigate these discretization artifacts, we provide an alternative "triple-sided" probing strategy. We introduce a midpoint baseline action, defined as $u^{\text{mid}} = (\overline{u} + \underline{u})/2$. For each control dimension j, we independently evaluate the value network at the next states resulting from applying the upper bound ($x^{\overline{u}_j}$), the lower bound ($x^{\underline{u}_j}$), and the midpoint ($x^{u_j^{\text{mid}}}$). The optimal probing control $\hat{u}_j^*$ is then selected by maximizing the value function across these three discrete candidate actions:

$$\hat{u}_j^* = \arg \max_{u_j \in \{\underline{u}_j, u_j^{\text{mid}}, \overline{u}_j\}} V_\theta(x^{u_j}).$$

B.3 Gamma Discounting

Unlike standard RL-based reachability methods, our framework does not inherently require a discount factor ($\gamma < 1$) to induce a contraction mapping for the Bellman operator. Because we evaluate the problem within a bounded temporal curriculum, the exact, undiscounted optimal value propagates backward smoothly without diverging.

However, incorporating a discount factor $\gamma \in (0, 1]$ can still be advantageous for learning BRAT solutions in practice by providing a denser signal, incentivizing earlier goal reaching. We can seamlessly incorporate this temporal discount factor into our framework, resulting in discounted safety value solutions:

$$\begin{aligned} V_S^*(x,k) &= \max_{\mathbf{u}} \min_{\mathbf{d}} \left[\min_{\kappa \in [k,K]} \gamma^{\kappa-k} \ell(x_\kappa) \right], \\ V_L^*(x,k) &= \min_{\mathbf{u}} \max_{\mathbf{d}} \left[\min_{\kappa \in [k,K]} \left(\max \left(\gamma^{\kappa-k} g(x_\kappa), \max_{s \in [k,\kappa]} -\gamma^{s-k} \ell(x_s) \right) \right) \right]. \end{aligned} \tag{18}$$

C Environment Setup and Training Configurations

C.1 Environment Definitions

C.1.1 Narrow Passage

The Narrow Passage environment models two vehicles navigating in opposite directions along a shared road segment that contains a stationary obstacle.

State and Action Space. The joint state is $x = [x_1, y_1, \theta_1, v_1, \phi_1,\ x_2, y_2, \theta_2, v_2, \phi_2] \in \mathbb{R}^{10}$, where (x_i, y_i) is the position, $\theta_i \in [-\pi, \pi]$ is the heading, $v_i \in [0.1, 7.0]$ m/s is the speed, and $\phi_i \in [-0.3\pi, 0.3\pi]$ rad is the steering angle of vehicle $i \in \{1, 2\}$. The control input is $u = [a_1, \dot{\phi}_1, a_2, \dot{\phi}_2]$ with $|a_i| \leq 2.0\,\text{m/s}^2$ and $|\dot{\phi}_i| \leq 3\pi$ rad/s.

Dynamics. Each vehicle follows the kinematic bicycle model with wheelbase $L = 2.0$ m:

$$\dot{x}_i = v_i \cos\theta_i, \qquad \dot{y}_i = v_i \sin\theta_i, \qquad \dot{\theta}_i = \frac{v_i \tan\phi_i}{L}, \qquad \dot{v}_i = a_i, \qquad \dot{\phi}_i = \dot{\phi}_i^{\text{ctrl}}. \tag{19}$$

Target and Obstacle Sets. Vehicle 1 must reach goal $g_1 = (6.0, -1.2)$ m and vehicle 2 must reach $g_2 = (-6.0, 1.2)$ m, both within distance L:

$$\ell_{\text{target}}(x) = \max(\|[x_1, y_1] - g_1\| - L,\ \|[x_2, y_2] - g_2\| - L)\,. \tag{20}$$

The failure set encodes four constraint types, all of which must be satisfied:

- **Road curbs:** $y_i \in [\text{curb}_{\text{lo}} + L/2,\ \text{curb}_{\text{hi}} - L/2]$ with $\text{curb}_{\text{lo}} = -2.8$ m, $\text{curb}_{\text{hi}} = 2.8$ m.
- **Lateral bounds:** $x_i \in [-10.0 + L/2,\ 10.0 - L/2]$ m.
- **Stranded vehicle:** $\|[x_i, y_i] - p_{\text{obs}}\| \geq L$ with $p_{\text{obs}} = (0.0, -1.8)$ m.
- **Mutual collision:** $\|[x_1, y_1] - [x_2, y_2]\| \geq L$.

The avoid function aggregates all constraints as $\ell_{\text{avoid}}(x) = -w \cdot \min_k d_k(x)$, where $\{d_k\}$ are the signed distances to each constraint boundary and $w = 10.0$.

Neural Network Inputs. The 8 non-heading state components are normalized to $[-1, 1]$; the two headings θ_1, θ_2 are encoded as $(\sin\theta_i, \cos\theta_i)$, yielding a 12-dimensional input vector $\phi(x) \in \mathbb{R}^{12}$.

C.1.2 Quadrotor Gate Avoidance

This environment is formulated as a safety (BRT) problem: a quadrotor approaching a gate along the $+x$ direction must avoid crashing into the gate frame.

State and Action Space. The state is $x = [p_x, p_y, p_z, q_w, q_x, q_y, q_z, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z] \in \mathbb{R}^{13}$, where (p_x, p_y, p_z) is position, (q_w, q_x, q_y, q_z) is the quaternion, (v_x, v_y, v_z) is linear velocity, and $(\omega_x, \omega_y, \omega_z)$ is angular velocity. State bounds are $p_x \in [-4.0, 1.0]$ m, $p_y, p_z \in [-1.5, 1.5]$ m, and $\omega_i \in [-5.0, 5.0]$ rad/s. The 4D control input is $u = [\Delta T, \dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z]$, where $\Delta T \in [-9.8, 9.8]$ N is a collective thrust perturbation around the hover thrust $T_h = 9.8$ N, and $|\dot{\omega}_{x,y}| \leq 8.0$ rad/s^2, $|\dot{\omega}_z| \leq 4.0$ rad/s^2.

Dynamics. The continuous-time equations of motion are:

$$\dot{p} = v, \tag{21}$$

$$\dot{q} = \tfrac{1}{2}\,\Omega(q)\,\omega, \tag{22}$$

$$\dot{v}_x = 2(q_w q_y + q_x q_z)\,\frac{kT_c}{m}, \quad \dot{v}_y = 2(-q_w q_x + q_y q_z)\,\frac{kT_c}{m}, \quad \dot{v}_z = -g + (1 - 2q_x^2 - 2q_y^2)\,\frac{kT_c}{m}, \tag{23}$$

$$\dot{\omega}_x = \dot{\omega}_x^{\text{ctrl}} - \tfrac{5}{9}\,\omega_y\omega_z, \quad \dot{\omega}_y = \dot{\omega}_y^{\text{ctrl}} + \tfrac{5}{9}\,\omega_x\omega_z, \quad \dot{\omega}_z = \dot{\omega}_z^{\text{ctrl}}, \tag{24}$$

where $\Omega(q)$ is the quaternion kinematic matrix, $T_c = \Delta T + T_h$, $m = 1.0$ kg, $k = C_T/m = 1.0$, and $g = 9.8$ m/s^2.

Obstacle Set. The gate is an infinite wall at $x = 0$ with a rectangular opening $|p_y| \leq 0.5$ m, $|p_z| \leq 0.5$ m and a width 0.1 m. The quadrotor body is a ball of radius $r = 0.15$ m. The unsafe set combines gate collision with velocity bound violations: $v_x \in [0.5, 4.5]$ m/s, $v_y, v_z \in [-2.0, 2.0]$ m/s, and $\omega_i \in [-4.5, 4.5]$ rad/s. A tanh-scaled signed distance is used to sharpen the gate boundary.

Neural Network Inputs. All 13 state dimensions are used; position and velocity components are linearly normalized to $[-1, 1]$ while the quaternion is kept as a raw unit vector, yielding $\phi(x) \in \mathbb{R}^{13}$.

C.1.3 Publisher-Subscriber System (ND)

This benchmark scales our method to 40D and 80D by instantiating a weakly nonlinear system with $N \in \{40, 80\}$ states.

State and Action Space. $x \in [-1, 1]^N$, control $u \in [-u_{\max}, u_{\max}]^{N-1}$ with $u_{\max} = 0.5$. There is no adversarial disturbance.

Dynamics.

$$\dot{x} = Ax + Bu + \eta(x), \tag{25}$$

where $A = -0.5\,I_N - e_0\mathbf{1}_{1:N-1}^\top$, $B = [0;\ 0.4\,I_{N-1}]$, and the nonlinear term is

$$\eta_0(x) = \mu \sin(\alpha x_0)\,x_0^2, \qquad \eta_{i>0}(x) = -\gamma\,x_0^2\,x_i, \tag{26}$$

with $\gamma = 20.0$ and $\mu = \alpha = 0$. More details are available in [10].

Target Set. A scaled ellipsoid in $\mathbb{R}^N$:

$$\ell(x) = \tfrac{1}{2}\big(\|\mathrm{diag}(\varepsilon)\,x\|^2 - R^2\big) \leq 0, \quad \varepsilon = [\sqrt{N-1},\,1,\ldots,1]^\top, \quad R = \sqrt{N-1}\cdot 0.25. \tag{27}$$

Neural Network Inputs. The state is already in $[-1, 1]^N$ and is passed directly: $\phi(x) = x \in \mathbb{R}^N$.

C.1.4 Dubins Car (3D)

The Dubins car is a planar reach-avoid problem: a unicycle system with a fixed speed must reach a target region while avoiding circular obstacles.

State and Action Space. The state is $x = [p_x, p_y, \theta] \in [-5.0, 5.0]^2 \times [-\pi, \pi]$, where (p_x, p_y) is the planar position and θ is the heading. The scalar control is the turning rate $\omega \in [-\omega_{\max}, \omega_{\max}]$ with $\omega_{\max} = 1.0$ rad/s. Forward speed is fixed at $v = 1.0$ m/s.

Dynamics.

$$\dot{p}_x = v\cos\theta, \qquad \dot{p}_y = v\sin\theta, \qquad \dot{\theta} = \omega. \tag{28}$$

Target and Obstacle Sets. The target is a disk of radius $r_{\text{target}} = 0.5$ m centred at the origin:

$$\ell_{\text{target}}(x) = \|[p_x, p_y]\| - r_{\text{target}}. \tag{29}$$

The environment contains $N_{\text{obs}} = 5$ circular obstacles with centres $c_i \in \mathbb{R}^2$ and radii r_i, given in Table 4. The signed avoid function takes the value of the most dangerous obstacle at each state:

$$\ell_{\text{avoid}}(x) = \max_{i=1}^{N_{\text{obs}}} \big(r_i - \|[p_x, p_y] - c_i\|\big). \tag{30}$$

A state is unsafe if $\ell_{\text{avoid}}(x) > 0$, i.e. the vehicle is inside at least one obstacle.

Table 4: Default obstacle parameters for the Dubins car environment.

i	c_i^x	c_i^y	r_i
1	-0.70	0.20	0.30
2	1.20	-1.50	0.35
3	1.80	1.00	0.50
4	-2.00	1.50	0.40
5	-1.50	-2.00	0.25

Neural Network Inputs. Position is linearly normalized to $[-1, 1]^2$ and the heading is encoded as $(\sin\theta, \cos\theta)$, yielding $\phi(x) \in \mathbb{R}^4$.

C.2 Network Architectures

All experiments employ a window-partitioned architecture consisting of independent value and policy networks for each temporal window. The value network estimates the reachability value function, while the policy network jointly parameterizes both control and disturbance actions through a shared latent representation. For a horizon partitioned into M windows, the framework instantiates M value networks and M policy networks, each responsible for a local temporal segment.

Windowed Value Network. Each temporal window is represented by a time-conditioned residual MLP that approximates the local value function $V(x, k)$. The architecture consists of:

1. **Input stem:**
$$[k, \phi(x)] \to \text{Linear}(1 + \phi_{\dim}, W_V)^{L_V} \to \text{ReLU},$$
where $\phi(x)$ denotes the state encoding.
2. **Residual trunk:** L residual blocks, each consisting of a two-layer MLP with a skip connection.
3. **Output head:**
$$\text{Linear}(W_V, 1),$$
producing the scalar value estimate $V_\theta(x, k)$.

Unless otherwise specified, all value networks use width $W_V = 512$, depth $L_V = 8$, and are optimized using AdamW with learning rate 3×10^{-4}.

Windowed Vector-Quantized Policy Network. Each temporal window is associated with a policy network that jointly predicts control and disturbance actions. The network maps states into a latent action embedding which is quantized onto the vertices of the bang-bang action set.

1. **Encoder:**
$$[k, \phi(x)] \to (\text{Linear} + \text{ReLU})^{L_\pi} \to \text{Linear}(W_\pi, d_u + d_d),$$
producing a continuous embedding
$$z_e \in \mathbb{R}^{d_u + d_d}.$$

2. **Vector quantization:** The embedding is projected onto a fixed (implicit) codebook containing all vertices of the admissible bang-bang action set. The nearest codebook entry
$$z_q = \arg\min_{c \in \mathcal{C}} \|z_e - c\|_2$$
is selected and returned through a straight-through estimator (STE), allowing gradients to propagate through the continuous embedding during training.

3. **Action decoding:** The quantized embedding is interpreted as the joint control-disturbance action vector and scaled to the corresponding control and disturbance limits during rollout execution.

Unless otherwise specified, all policy networks use width $W_\pi = 512$, depth $L_\pi = 5$, and are optimized using AdamW with learning rate 3×10^{-4}.

Table 5: Default architecture and training hyperparameters used across all experiments.

Hyperparameter	**Value**
Architecture	
Value network width W	512
Value network depth L	8
Policy network width W_π	512
Policy network depth L_π	5
Optimization	
Optimizer	AdamW
Learning rate	3×10^{-4}
Batch size	4096
Value Learning	
TD(λ) coefficient	0.5
Maximum rollout length M	5
Discount factor γ	1.0
Teacher mixing fraction α	0.5
Boundary-correction LR	5×10^{-5}
Anchor loss weight λ_{anchor}	5.0

C.3 Training Configuration

Unless otherwise specified, all experiments use the following training configuration.

Optimization. Value and policy networks are optimized using AdamW with a learning rate of 3×10^{-4} and batch size 4096. Policy and value updates are alternated throughout training, with each phase terminated according to a Bellman-error convergence criterion.

Rollout Targets. Value targets are generated using a TD(λ) mixture with $\lambda = 0.5$ and a maximum rollout horizon of $M = 5$ steps. All experiments use $\gamma = 1.0$.

State Sampling. Training batches combine multiple sampling strategies to improve boundary resolution. Unless otherwise specified, 30% of samples are drawn uniformly from the state space, 15% near target boundaries, 15% near obstacle boundaries, and 40% near the current reachability boundary. Boundary samples are generated using a boundary-aware replay buffer with band half-width 0.1 and capacity 4096.

Temporal Curriculum. Training proceeds backward from the terminal boundary condition using the windowed temporal curriculum described in Sec. 3. Each temporal window is initialized from the subsequent converged window and optimized independently. Before freezing a window, a boundary-correction phase is performed using rollout-derived targets generated by the learned student policies. Boundary correction uses a reduced learning rate of 5×10^{-5} and an anchor loss weight of $\lambda_{\text{anchor}} = 5.0$.

Policy Learning. Policies are trained through supervised imitation of teacher actions obtained via gradient-free bang-bang value probing. During rollout target generation, teacher and student actions are mixed with a fixed teacher injection fraction of $\alpha = 0.5$.

Table 6: Benchmark-specific configurations. d_x: state dimension; d_ϕ: NN input dimension excluding the time input (after encoding); d_u/d_d: control/disturbance dimensions.

Benchmark	**Horizon (s)**	**Windows**	$\mathbf{\Delta t}$	$\mathbf{d_x}$	$\mathbf{d_\phi}$	$\mathbf{d_u}$	$\mathbf{d_d}$
Dubins Car	4.0	4	0.02	3	4	1	0
Narrow Passage	8.0	16	0.05	10	12	4	0
Quadrotor	2.0	8	0.01	13	13	4	0
Pursuit-Evasion	8.0	8	0.05	10	13	2	2
40D Network	1.0	2	0.01	40	40	39	0
80D Network	1.0	2	0.01	80	80	79	0

D Ablations

D.1 Teacher-Mixing Fractions

To evaluate the role of teacher-student mixing, we ablate the teacher injection fraction α within each temporal window, comparing student-only ($\alpha = 0$), teacher-only ($\alpha = 1.0$), balanced mixture ($\alpha = 0.5$), and decaying ($1.0 \to 0.0$) schedules.

As shown in Table 7, the constant mixture achieves the smallest success-rate deviation from the numerical solution (0.206%) while remaining competitive on both rollout cost metrics. Teacher-only supervision produces the lowest rollout-cost discrepancy as expected, but exhibits the largest success-rate gap, suggesting that excessive reliance on teacher targets may bias policy learning. Student-only and decaying schedules achieve intermediate performance across all metrics. Overall, these results indicate that maintaining a fixed level of teacher guidance provides the best balance between policy accuracy and rollout consistency.

Overall, the results suggest that performance is relatively robust to the choice of teacher schedule, although the constant mixture consistently provides the closest match to the numerical baseline in terms of success rate.

Table 7: Ablation of the teacher mixing fraction α on the Dubins3D benchmark. Metrics report the absolute difference between our learned policy and the numerical solution over 20,000 test states. ΔSucc denotes the success-rate gap, while ΔCost Mean and ΔCost Std denote the differences in mean rollout cost and rollout-cost standard deviation, respectively. Lower values indicate closer agreement with the numerical baseline.

Curriculum Schedule (α)	**Δ Succ % $\downarrow$**	**Δ Cost Mean $\downarrow$**	**Δ Cost Std $\downarrow$**
Pure Student ($\alpha = 0.0$)	$0.2210\% \pm 0.0859\%$	0.0356 ± 0.0015	0.0772 ± 0.0021
Pure Teacher ($\alpha = 1.0$)	$0.3362\% \pm 0.0608\%$	$\mathbf{0.0274 \pm 0.0013}$	$\mathbf{0.0643 \pm 0.0025}$
Decaying Curriculum ($\alpha : 1.0 \to 0.0$)	$0.2267\% \pm 0.0607\%$	0.0299 ± 0.0070	0.0699 ± 0.0061
Constant Mixture ($\alpha = 0.5$)	$\mathbf{0.2056\% \pm 0.0856\%}$	0.0312 ± 0.0018	0.0706 ± 0.0026

D.2 Impact of Time Discretization step Δt

To analyze the sensitivity of our discrete-time framework to the underlying temporal discretization, we ablate the choice of the step size Δt during both the probing and dynamic rollout phases. Be-

cause our method relies on finite value differences to infer continuous-time optimal policy signs ($\text{sign}(C_{u,j}) \approx \text{sign}(\Delta V_j^u)$), the choice of Δt embodies a critical numerical trade-off.

An excessively large Δt may introduce significant linearization errors, distorting the local approximation of the system physics and leading to degraded teacher action labels near complex boundary manifolds. Conversely, while an infinitesimally small Δt theoretically recovers the true analytical directional derivative, it can practically bottleneck optimization by drastically expanding the number of discrete steps required to span the terminal horizon K. This temporal expansion increases the computational burden of multi-step student rollouts, increasing total training time, and renders the backward curriculum more vulnerable to bootstrapping error propagation across adjacent temporal windows. Furthermore, as Δt diminishes, the probing teacher becomes highly vulnerable to approximation errors in the learned value function.

We evaluate this sensitivity across multiple scales of Δt on the Narrow Passage benchmark problem. The resulting metrics, compiled in Table 8, track the empirical False Positive Rate (FPR), False Negative Rate (FNR), and total rollout success rate under identical network parameterizations.

As shown in Table 8, performance is relatively insensitive to the choice of Δt across a broad operating range. For this specific example, moderate discretization steps (0.1–0.15s) provide the best trade-off between computational efficiency and control performance, achieving the highest rollout success rates while substantially reducing training time. Only when the discretization becomes excessively coarse ($\Delta t = 0.3$s) does performance degrade noticeably, indicating that the local dynamics approximation becomes insufficient for accurate value propagation.

Table 8: Ablation results isolating the impact of the time discretization step size Δt on safety metrics and rollout performance over $20,000$ test states.

Discretization Step (Δt)	**FPR % ↓**	**FNR % ↓**	**Succ % ↑**	**Time (h) ↓**
$\Delta t = 0.025$s	100.0	0.0	82.01	11
$\Delta t = 0.050$s	100.0	0.0	94.29	5
$\Delta t = 0.075$s	100.0	0.0	93.49	4
$\Delta t = 0.100$s	100.0	0.0	96.31	3
$\Delta t = 0.125$s	100.0	0.0	99.99	2.5
$\Delta t = 0.150$s	100.0	0.0	99.98	1.5
$\Delta t = 0.300$s	0.0	100.0	78.75	1

E Additional Benchmarking Results

E.1 Evaluation on Two-Player Zero-Sum Games

E.1.1 4D Human-Robot Collision Avoidance

We first formulate a 4D human-robot collision avoidance scenario as a Worst-Case Backward Reachable Tube (BRT). In this game, an ego vehicle (modeled as a 4D Dubins car) attempts to avoid an adversarial human agent (modeled as a 3D Dubins vehicle). The system is modeled in a relative coordinate frame with the following dynamics:

$$\begin{aligned}
\dot{x}_{\text{rel}} &= -v_{\text{ego}} + v_{\text{human}} \cos(\theta_{\text{rel}}) + \omega_{\text{ego}} y_{\text{rel}} \\
\dot{y}_{\text{rel}} &= v_{\text{human}} \sin(\theta_{\text{rel}}) - \omega_{\text{ego}} x_{\text{rel}} \\
\dot{\theta}_{\text{rel}} &= \omega_{\text{human}} - \omega_{\text{ego}} \\
\dot{v}_{\text{ego}} &= a_{\text{ego}}
\end{aligned}$$

The 4D state vector is defined as $x = [x_{\text{rel}}, y_{\text{rel}}, \theta_{\text{rel}}, v_{\text{ego}}]^\top$, where x_{rel} and y_{rel} denote the relative position of the human with respect to the ego vehicle, θ_{rel} is the relative heading, and v_{ego} is the linear velocity of the ego vehicle. The state bounds are given as $[-3.0, 3.0]^2 \times [-\pi, \pi] \times [0.1, 1.0]$. We use Euler integration to obtain the discrete-time dynamics.

The ego vehicle acts as the control player aiming to maximize safety, with control inputs $u = [\omega_{\text{ego}}, a_{\text{ego}}]^\top$ bounded by $\omega_{\text{ego}} \in [-1.0, 1.0]$ and $a_{\text{ego}} \in [-2.0, 2.0]$. The human acts as the distur-

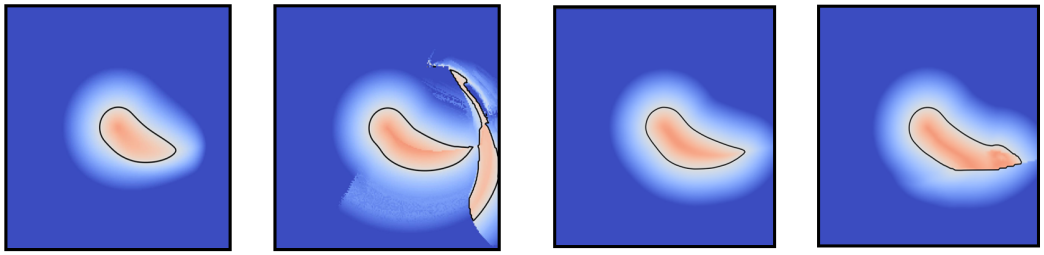
bance player aiming to cause a collision, with a higher turn rate $d = \omega_{\text{human}} \in [-2.0, 2.0]$. The human is assumed to move at a constant linear velocity $v_{\text{human}} = 1.0$.

Table 9: Quantitative safety evaluation across 20,000 random rollouts in the 4D collision avoidance environment.

Method	**False Positives (FP)** $\downarrow$	**False Negatives (FN)** $\downarrow$
OptimizedDP	1569	**10**
Ours	**206**	35

The safety set is defined by avoiding a collision ball of radius $r_{\text{goal}} = 0.5$ around the ego vehicle: $\ell(x) = \sqrt{x_{\text{rel}}^2 + y_{\text{rel}}^2} - r_{\text{goal}}$.

In this scenario, both players possess distinct advantages—the ego vehicle has greater longitudinal flexibility through variable speed, while the human has a higher maximum turn rate. This asymmetric control authority introduces sharp value gradients that cause numerical artifacts in the grid-based OptimizedDP solver [4], as illustrated by the qualitative value and cost function heatmaps in Fig. 3a. Quantitatively, the value-policy consistency is further evaluated with numbers of FP and FN among 20,000 trajectory rollouts, as shown in Table. 9. Our method yields a significantly smaller number of FP while being slightly more conservative with a higher number of FN.



(a) OptimizedDP solution (b) Our solution

Figure 3: Qualitative comparison of the 4D human-robot collision avoidance game. For both methods, we display the predicted value function (left sub-panel) alongside the reachability rollout cost evaluated from its induced policy (right sub-panel). The heatmap slices are taken at x-y plane where $\theta_{\text{rel}} = \pi/2$ and $v_{\text{ego}} = 1.0$. The grid-based OptimizedDP solution exhibits severe numerical artifacts due to the asymmetric control authority between players. In contrast, our neural framework maintains a relatively more consistent safety boundary that closely matches the empirical rollout cost.

E.1.2 10D Multi-Agent Pursuit-Evasion

We scale our evaluation to a high-dimensional reach-avoid problem featuring one evader (modeled as a 4D Dubins car) and two cooperative pursuers (modeled as 3D Dubins cars). The game takes place in a bounded 10×10 field containing a central circular obstacle with a radius of 1.

The 10D state vector concatenates the states of all three agents: $x = [x_e, y_e, \theta_e, v_e, x_{p_1}, y_{p_1}, \theta_{p_1}, x_{p_2}, y_{p_2}, \theta_{p_2}]^\top$. The spatial coordinates for all agents are bounded by the field size $x, y \in [-5.0, 5.0]$, headings by $\theta \in [-\pi, \pi]$, and the evader's velocity by $v_e \in [0.0, 2.0]$. The two pursuers act as the joint control player (minimizer) aiming to capture the evader, with control inputs $u = [\omega_{p_1}, \omega_{p_2}]^\top$ bounded by $\omega_{p_i} \in [-1.0, 1.0]$. The evader acts as the disturbance player aiming to escape, with inputs $d = [\omega_e, a_e]^\top$ bounded by $\omega_e \in [-1.2, 1.2]$ and $a_e \in [-2.0, 2.0]$. Both pursuers move at a constant linear velocity $v_{\text{const}} = 1.0$. Again, the problem is formulated with asymmetric control authority, leading to intriguing optimal strategies and numerical stiffness during training.

The continuous-time dynamics of the multi-agent system are given by

$$\begin{aligned} \dot{x}_e &= v_e \cos(\theta_e), & \dot{y}_e &= v_e \sin(\theta_e), & \dot{\theta}_e &= \omega_e, & \dot{v}_e &= a_e \\ \dot{x}_{p_i} &= v_{\text{const}} \cos(\theta_{p_i}), & \dot{y}_{p_i} &= v_{\text{const}} \sin(\theta_{p_i}), & \dot{\theta}_{p_i} &= \omega_{p_i}, & i &\in \{1, 2\} \end{aligned}$$

and we apply Euler integration to obtain the discrete-time dynamics.

The target set is successfully reached if the evader is caught (comes within a capture radius of $r_{\text{catch}} = 0.75$ of any pursuer) or if the evader crashes into an obstacle/wall:

$$g(x) = \min\left(\min_{i\in\{1,2\}}\left(\|p_e - p_{p_i}\| - r_{\text{catch}}\right), \text{SDF}(p_e)\right),$$

where $\text{SDF}(\cdot)$ denotes the signed distance function to the boundary and the central circular obstacle. The failure set is triggered if the pursuers fail the mission by colliding with the obstacles or each other:

$$\ell(x) = \min\left(\min_{i\in\{1,2\}} \text{SDF}(p_{p_i}), \|p_{p_1} - p_{p_2}\| - 2r_{\text{robot}}\right).$$

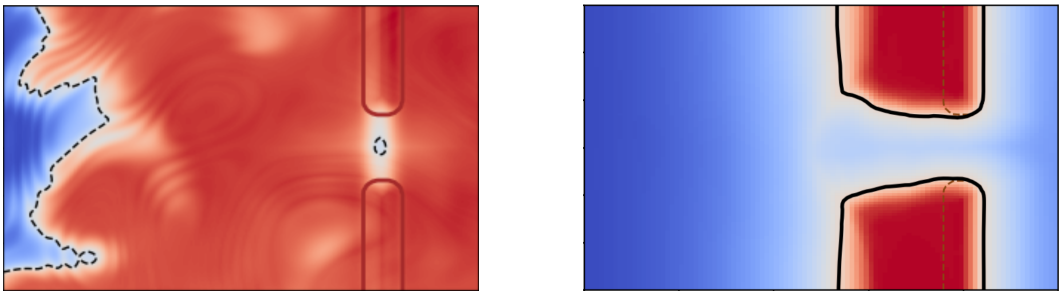
Table 10: Quantitative catching rates for the 10D pursuit-evasion game. We evaluate the performance of our method against the MPPI baseline by pairing different combinations of pursuer and evader policies.

		Evader Policy	
		MPPI	**Ours**
Pursuer Policy	**MPPI**	36%	0%
	Ours	**88%**	**42%**

To evaluate the optimality of our learned policies, we benchmark them against a Model Predictive Path Integral (MPPI) baseline [22] in a cross-play evaluation. The MPPI pursuers operate independently, each individually minimizes their cumulative distance to the evader while incorporating an SDF penalty term for safety awareness. Conversely, the MPPI evader operates by simply maximizing its distance from the pursuers while avoiding obstacles.

Table 10 presents the quantitative catching rates across all combinations of pursuer and evader policies. Our framework demonstrates dominant performance in both the evading and cooperative pursuing roles. When our learned evader is deployed against the baseline MPPI pursuers, it successfully escapes in 100% of the 50 trials. Conversely, our joint pursuer policy captures the MPPI evader in 88% of the rollouts, vastly outperforming the baseline MPPI pursuers, which only achieve a 36% success rate against their own evader. Finally, when putting our learned pursuers against our learned evader, the catch rate settles at 42%.

E.2 Qualitative Analysis of Quadrotor Gate Avoidance



(a) DeepReachMPC Solution (Sine NN) (b) Ours

Figure 4: Qualitative value function heatmaps for the Quadrotor Gate Avoidance task. The slices visualize learned value function on the x-z plane with $[p_y, q_w, q_x, q_y, q_z, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z] = [0, 1, 0, 0, 0, 2.5, 0, 0, 0, 0, 0]$. (a) The sinusoidal MLP induces severe rippling artifacts. In contrast, our discrete-time formulation yields.

A primary motivation for adopting our discrete-time formulation is the flexibility to employ standard, highly stable neural network architectures (e.g., ReLU MLPs) to represent the value function.

In contrast, PINN-based solvers rely heavily on sinusoidal MLPs to accurately represent the value function gradients required by the HJI PDE. While the sinusoidal networks excel at handling differentiable boundary conditions, they often struggle to approximate complex non-smooth safety boundaries.

As an example, the safety boundary in the Quadrotor Gate Avoidance problem is highly complex and non-smooth, defined by the SDF to the gate coupled with the velocity bounds. To demonstrate this advantage, we provide qualitative heatmap slices of the learned value functions in Fig. 4. The PINN-based baseline (Fig. 4a), constrained by its sinusoidal network, exhibits severe high-frequency oscillatory artifacts (ripples) and fails entirely to resolve the true value function near the gate. Conversely, our gradient-free framework (Fig. 4b) stably propagates the safety boundary backward over the temporal curriculum.

F Details on Hardware Setup and Experiments

This section provides additional details for the race car BRT training and safety filtering.

F.1 Vehicle Dynamics and Problem Setup

We adopt the F1Tenth hybrid dynamics from the F1Tenth simulator [23]. To train a robust safety value function capable of handling varying levels of real-world uncertainty, we augment the state space with a disturbance scale parameter, $d_s \in [0, 0.3]$.

The state is then given by $x = (p_x, p_y, \phi_s, v, \theta_{yaw}, \omega_{yaw}, \beta_{slip}, d_s)$ where $\phi_s \in [-0.4189, 0.4189]$, $v \in [0.5, 10]$, $\omega_{yaw} \in [-5, 5]$, and $\beta_{slip} \in [-0.8, 0.8]$. Here, ϕ_s denotes the steering angle, v is the longitudinal velocity, θ_{yaw} is the yaw angle, ω_{yaw} is the yawing rate, and β_{slip} denotes the slip angle. The control inputs are the rate of steering angle and the longitudinal acceleration, denoted as $u = [\dot{\phi}, a] \in [-3.2, 3.2] \times [-9.51, 9.51]$. The disturbances include lateral drift scaler $v_l \in [-1, 1]$ and control authority deduction d_ϕ and d_a, whose bounds are scaled by an extra conditional dimension d_s. The longitudinal acceleration is further capped when the current velocity v is high, and the system switches to kinematic mode when the velocity is too low. We increase the switching speed for the dynamic mode to 1.5 m/s to match the speed limit of the hardware RC car. Namely, the F1Tenth car is in its kinematic mode if the speed is less than 1.5. Otherwise, the dynamic mode will be used. We now present the continuous-time dynamics for each mode. Euler integration is employed to provide discrete-time dynamics.

The dynamic-mode dynamics are:

$$\mathbf{f} = \begin{bmatrix} v\cos(\theta_{yaw} + \beta_{slip}) - v_l d_s(|v|+1)\sin(\theta_{yaw}) \\ v\sin(\theta_{yaw} + \beta_{slip}) + v_l d_s(|v|+1)\cos(\theta_{yaw}) \\ \dot{\phi} - d_\phi d_s \\ a - d_a d_s \\ \omega_{yaw} \\ -\frac{\mu m}{vI(l_r+l_f)}\left(l_f^2 C_{Sf}(gl_r - ah) + l_r^2 C_{Sr}(gl_f + ah)\right)\omega_{yaw} \\ +\frac{\mu m}{I(l_r+l_f)}\left(l_r C_{Sr}(gl_f + ah) - l_f C_{Sf}(gl_r - ah)\right)\beta_{slip} \\ +\frac{\mu m}{I(l_r+l_f)} l_f C_{Sf}(gl_r - ah)\phi_s, \\ \left(\frac{\mu}{v^2(l_r+l_f)}\left(C_{Sr}(gl_f + ah)l_r - C_{Sf}(gl_r - ah)l_f\right) - 1\right)\omega_{yaw} \\ -\frac{\mu}{v(l_r+l_f)}\left(C_{Sr}(gl_f + ah) + C_{Sf}(gl_r - ah)\right)\beta_{slip} \\ +\frac{\mu}{v(l_r+l_f)} C_{Sf}(gl_r - ah)\phi_s \end{bmatrix}.$$

where the vehicle parameters are borrowed directly from the F1Tenth simulator:

- μ: Surface friction coefficient: 1.0489
- C_{Sf}: Cornering stiffness coefficient, front: $4.718/$ rad
- C_{Sr}: Cornering stiffness coefficient, rear: $5.4562/$ rad
- l_f: Distance from center of gravity to front axle: 0.15875 m

- l_r: Distance from center of gravity to rear axle: $0.17145\,\text{m}$
- h: Height of center of gravity: $0.074\,\text{m}$
- m: Total mass of the vehicle: $3.74\,\text{kg}$
- I: Moment of inertia of the entire vehicle about the z-axis: $0.04712\,\text{kg}\cdot\text{m}^2$

The kinematic-mode dynamics are:

$$\mathbf{f} = \begin{bmatrix} v\cos(\theta_{yaw}) - v_l d_s(|v|+1)\sin(\theta_{yaw}) \\ v\sin(\theta_{yaw}) + v_l d_s(|v|+1)\cos(\theta_{yaw}) \\ \dot{\phi} - d_\phi d_s \\ a - d_a d_s \\ \frac{v}{l_r+l_f}\tan(\phi_s) \\ \frac{a}{l_r+l_f}\tan(\phi_s) + \frac{v}{(l_r+l_f)\cos^2(\phi_s)}\dot{\phi} \\ 0 \end{bmatrix}.$$

F.2 Network Architecture and Inputs

To effectively process high-dimensional observations, our neural network architectures decouple visual feature extraction from the other input features. The network inputs consist of a 128×128 ego-centric BEV occupancy map, alongside a 5-dimensional proprioceptive state vector $(\phi, v, \omega_{yaw}, \beta_{slip}, d_s)$ and the time t, forming a $16,390$-dimensional input space.

To process the visual input, we employ a CNN encoder, which consists of five convolutional layers, each utilizing a 3×3 kernel, a stride of 2, a padding of 1, and ReLU activations. This architecture progressively downsamples the $1 \times 128 \times 128$ spatial occupancy map into a $128 \times 4 \times 4$ feature tensor. The tensor is subsequently flattened and passed through a linear projection layer to yield a dense, 256-dimensional embedding.

The 256-dimensional BEV embedding is then concatenated with the proprioceptive states and the time variable. This joint feature vector is passed into a ResNet with 8 residual blocks, each maintaining the 512-neuron width with ReLU activations, to predict the value function or optimal actions. Importantly, the entire architecture is trained end-to-end from scratch, including the BEV encoder.

F.3 Discrete-Time CBF Filter

To ensure safety during online deployment, we gaurd the task-driven MPPI nominal control, u_{nom}, using a sampling-based variant of the DCBF filter [21].

First, we define the calibrated safety value function as:

$$V_{\text{cal}}(x) = V_\theta(x, 0) - \delta \tag{31}$$

where $V_\theta(x, 0)$ is the raw output of the learned value network and δ is the calibration threshold obtained following [24]. To maintain safety, the DCBF strictly bounds the decaying rate of the safety value based on the current value. We define this required minimum safety value for the next step as:

$$V_{\text{req}}(x) = \max\left((1 - \gamma_{\text{DCBF}}\Delta t)V_{\text{cal}}(x), 0\right) \tag{32}$$

where $\gamma_{\text{DCBF}} = 1.0$ is the relaxation rate.

Since our method does not approximate the value function gradients, we utilize a sampling-based strategy to evaluate the DCBF constraint over a batch of candidate controls, $\mathcal{U}_{\text{batch}}$. The subset of safe candidate controls is formulated as:

$$\mathcal{U}_{\text{safe}} = \left\{ u \in \mathcal{U}_{\text{batch}} \mid V_{\text{cal}}\left(x_{k+1}^{u,d_\phi}\right) \geq V_{\text{req}}(x) \right\} \tag{33}$$

The filtered control output is then determined by a switching logic. If the safe control set is non-empty, the filter selects the safe candidate closest to the nominal control. If no sampled control

satisfies the safety constraint, the filter triggers the robust fallback recovery policy π^u_ϕ:

$$u_{\text{filtered}} = \begin{cases} \arg\min_{u \in \mathcal{U}_{\text{safe}}} \|u - u_{\text{nom}}\|_2, & \text{if } \mathcal{U}_{\text{safe}} \neq \emptyset \\ \pi^u_\phi(x_{\text{clamped}}, 0), & \text{otherwise} \end{cases} \tag{34}$$

where x_{clamped} restricts the state within the strict bounds of the training domain prior to evaluating the fallback policy. Theoretically, if the BRT solution is exact and all encountered disturbances remain strictly within the training bounds, the safe control set is guaranteed to be non-empty. In practice, however, neural network approximation errors or out-of-distribution observations and disturbances may occasionally render the safe set empty. The fallback recovery logic is explicitly introduced to maintain persistent system safety despite these practical imperfections.