TAMING THE LEARNING DYNAMICS OF NEURAL BARRIERS USING PROJECTED ANCHOR SETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural control barrier functions (NCBFs) provide expressive safety certificates for complex dynamical systems controlled by neural policies, but training them remains brittle. Learning an NCBF requires jointly shaping a high-dimensional safety boundary and finding controls that preserve it, coupling barrier fitting, worst-case state search, and control selection in a difficult min–max–min optimization problem. We propose a new training formulation that returns to the original Nagumo boundary conditions for forward invariance. Our method exploits the fact that safety violations are often observable in a low-dimensional space even when the system state space is high-dimensional. As a data-driven analog of high-order CBF reasoning, multistep recovery searches backtrack along failed rollouts to identify states where changing the actions can still avert failure. These full-state anchors pin down a candidate zero level set, while their recovery trajectories supervise barrier fitting and policy fine-tuning. The resulting loop jointly optimizes the barrier and policy using its own safety feedback, without expert safety demonstrations or a separate task reward. We specify conditions under which the learned boundary and policy can ensure forward invariance.

1 INTRODUCTION

Safe control is essential to deploying learning-based robotics. Control barrier functions (CBFs) certify safety by defining a safe set and constraining control actions to prevent trajectories from leaving the set (Ames et al., 2019). Neural CBFs extend this idea to complex systems by learning the barrier from sampled data (Dawson et al., 2023). However, replacing analytical construction of CBFs with the function approximation of neural networks may lead to low empirical loss without properly satisfying the CBF conditions over the state space. Existing methods mitigate this difficulty by using additional supervision, such as safe policies, demonstrations, or reinforcement-learning signals (So et al., 2024; Robey et al., 2020; Choi et al., 2020). This raises our central question: *can we learn neural control barrier functions without an external oracle for safe control?*

The core difficulty is that neural-barrier training couples optimization over the barrier, states, and admissible controls: changing the barrier changes both the safe set boundary and the controls required to preserve it. Rather than resolving this nested search through external supervision, we exploit an intrinsic source of feedback. While recoverability depends on the full state, physical safety is often observable through a fixed, lower-dimensional geometric projection. Projected rollout outcomes can therefore provide a stable reference for identifying where the evolving full-state barrier should lie and how its local boundary should be oriented.

We propose *Neural Barrier Functions with Projected Anchor Sets (NBF-PAS)*, a self-supervised framework that uses low-dimensional safety geometry to extract full-state boundary supervision from recovery behavior. Rather than assuming access to a safe-control oracle that provides safe actions at queried states, NBF-PAS requires only queryable rollouts: starting from failed or near-critical trajectories, it backtracks to selected full states and searches alternative actions for a successful recovery. These recovery witnesses supervise both the location of the safe set boundary and its local orientation, supporting iterative barrier refinement and policy improvement. Supervision comes from the system's own recovery behavior, without externally supplied safe controls, expert demonstrations, or an analytically specified barrier.

We evaluate NBF-PAS on Dubins, Segway, quadrotor, and Hopper systems with nominal controllers, comparing learned boundary geometry and closed-loop collision rates against direct neural-barrier

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

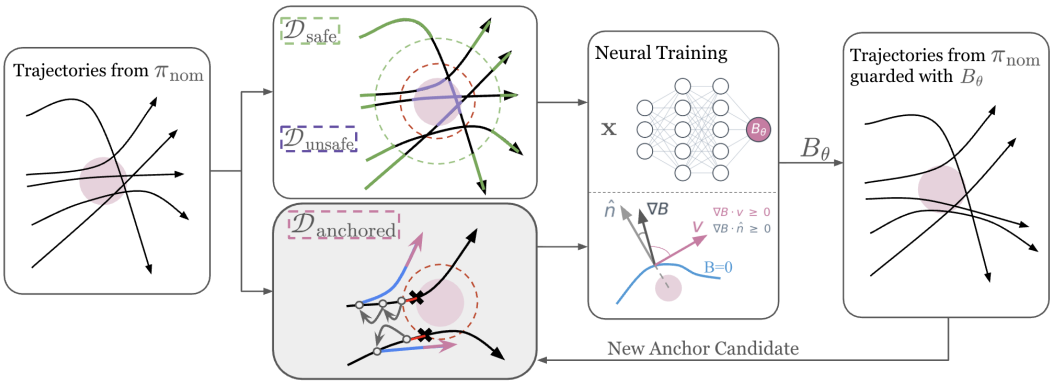


Figure 1: **NBF-PAS Pipeline.** Given a set of trajectories rolled out with nominal controller π_{nom}, we partition it into safe set $\mathcal{D}_{\text{safe}}$ (colored in green), unsafe set $\mathcal{D}_{\text{unsafe}}$ (colored in purple). Using our backtracking recovery search (grey block), we additionally produce the anchored set (trajectories colored in pink). We feed this dataset into the neural network training and aim at aligning the barrier normal ∇B_θ. After the learning converges, we roll out using π_{nom} guarded by B_θ to collect new anchored set candidates. The loop ends when termination criteria is satisfied.

training. We also assess falls and goal-reaching behavior, including barrier-guided policy retraining versus pure reinforcement learning on Hopper. The results show that projected recovery feedback can improve boundary geometry and guide neural policies toward collision avoidance, while highlighting that collision reduction alone does not ensure task performance.

We make the following contributions. First, we formulate neural CBF learning without a safe-control oracle (Section 3). Second, we introduce projected anchors and recovery-based refinement (Sections 4.2–4.3). Third, we relate recovery evidence to forward invariance while distinguishing finite-horizon success from certification (Section 4.1; Appendix C). Fourth, we evaluate NBF-PAS on four systems spanning boundary geometry, closed-loop safety, and task recovery (Section 5).

2 Related Work

Learning barriers and controllers. Neural certificates offer flexible representations of safety sets, but introduce a gap between optimization on data and verification over a continuous domain (Dawson et al., 2023). Robey et al. (2020) learn CBFs from safe expert trajectories and derive guarantees under coverage and regularity assumptions. Dai et al. (2022) enlarge a conservative handcrafted barrier using data collected with an MPC safety mechanism. Taylor et al. (2020) learn the effect of model uncertainty on a given barrier derivative, while Choi et al. (2020) use reinforcement learning to compensate for model uncertainty in CLF/CBF control. These approaches differ in what is learned and what supplies the initial safety structure. Our method uses recovery searches to generate its own boundary and action supervision; regression onto these internally generated actions is still a form of policy distillation, but requires no external demonstrations.

Policy values and reachability. So et al. (2024) identifies difficulties in direct neural-CBF training under input constraints and construct policy neural CBFs from maximum-over-time safety costs. Their framework also includes policy iteration, so joint improvement of safety representations and policies is not itself our novelty. The distinction is the training target: we fit a boundary localized by recovery anchors rather than regress a policy's safety value throughout the state space. Control barrier-value functions likewise connect reachability and CBF-based control (Choi et al., 2021). Our recovery score is a local, finite-horizon reachability surrogate. It retains a control search, but concentrates that search on informative failures and stores its successful witnesses instead of solving a global value-function equation.

High-order and geometric safety constraints. High-order CBFs introduce recursively differentiated constraints so that control inputs can enter a safety condition with high relative degree (Xiao & Belta, 2022). Learned high-relative-degree barriers and uncertainty-aware constraints have also been studied (Wang et al., 2021; Khojasteh et al., 2020). Our connection is operational: multistep

interventions reveal delayed control effects without symbolic Lie derivatives. The recovery horizon is not an estimate of formal relative degree; it also depends on input bounds, sampling time, and the current state. Collision-cone CBFs exploit relative position and velocity to construct obstacle-avoidance constraints (Tayal et al., 2024; Goswami et al., 2024), and reduced-order models provide another route to tractable safety design (Cohen et al., 2024). We similarly exploit observable geometry, but keep the barrier on the full state and learn its dynamic margin from recovery trajectories rather than prescribe a universal workspace boundary.

3 Why Direct Neural-Barrier Training Is Difficult

3.1 Basic Definitions

Consider the deterministic controlled system

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \qquad \mathbf{x} \in \mathcal{D} \subseteq \mathbb{R}^n, \quad \mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^m, \tag{1}$$

where $\mathcal{D}$ is open, $\mathcal{U}$ is compact, and $f : \mathcal{D} \times \mathcal{U} \to \mathbb{R}^n$ is continuous in $(\mathbf{x}, \mathbf{u})$ and locally Lipschitz in $\mathbf{x}$ uniformly over $\mathbf{u}$. An admissible control is a measurable signal $\mathbf{u} : \mathbb{R}_{\geq 0} \to \mathcal{U}$; each initial state and admissible control induce a unique maximal trajectory. Let $\mathcal{X}_{\text{free}} \subseteq \mathcal{D}$ denote the physically collision-free region. A continuously differentiable barrier $B : \mathcal{D} \to \mathbb{R}$ represents a candidate safe set $\mathcal{C} = \{\mathbf{x} \in \mathcal{D} : B(\mathbf{x}) \geq 0\}$, which must satisfy $\mathcal{C} \subseteq \mathcal{X}_{\text{free}}$ to certify physical safety. The barrier depends on the full state because states with the same position but different velocities or orientations may require different safety interventions. A control barrier function (CBF) requires an extended class-$\mathcal{K}$ function α such that

$$\max_{\mathbf{u} \in \mathcal{U}} \left[\nabla B(\mathbf{x})^\top f(\mathbf{x}, \mathbf{u}) + \alpha(B(\mathbf{x})) \right] \geq 0, \qquad \mathbf{x} \in \Omega, \tag{2}$$

on a domain $\Omega \supseteq \mathcal{C}$. An admissible control satisfying this condition enforces $\dot{B}(\mathbf{x}) \geq -\alpha(B(\mathbf{x}))$, preventing trajectories initialized in $\mathcal{C}$ from crossing into $B(\mathbf{x}) < 0$.

In this work, B_θ is represented by a neural network. We therefore distinguish a learned *candidate barrier* from a certified CBF: fitting the CBF conditions on sampled states alone does not establish equation 2 throughout the desired certification domain.

3.2 Standard Training Methods and Their Difficulties

Standard neural-barrier training combines sign losses on sampled safe and unsafe states with a feasibility loss that penalizes violations of the CBF condition. The sign losses shape the candidate safe set, while the feasibility loss seeks an admissible control that preserves safety at each sampled state. Neural barriers are thus trained from finitely sampled states, whereas the defining CBF conditions must hold over a continuous state domain. Consequently, low training loss does not necessarily identify the correct invariant boundary or its local orientation. These ambiguities motivate the projected anchor construction introduced in the next section.

Nested Feasibility. For a candidate barrier B_θ, define the CBF residual

$$r_\theta(\mathbf{x}, \mathbf{u}) = \nabla B_\theta(\mathbf{x})^\top f(\mathbf{x}, \mathbf{u}) + \alpha(B_\theta(\mathbf{x})). \tag{3}$$

A representative neural-barrier objective can be written schematically as

$$\min_\theta \mathcal{L}_{\text{shape}}(\theta) + \lambda \sup_{\mathbf{x} \in \Omega_\theta} \min_{\mathbf{u} \in \mathcal{U}} [-r_\theta(\mathbf{x}, \mathbf{u})]_+^2 . \tag{4}$$

Here, $\mathcal{L}_{\text{shape}}$ collects the safe and unsafe sign losses, $\lambda > 0$ weights the feasibility penalty, and $[z]_+ = \max\{z, 0\}$. The inner optimization asks whether an admissible control can satisfy the CBF condition, while the outer search identifies states where the current barrier is infeasible. In practice, both are approximated using finitely sampled states and controls. Moreover, changing θ changes the learned boundary, its normal direction, and the controls required to preserve it, making direct barrier synthesis strongly coupled and nonconvex.

Geometric Safety Does Not Determine Dynamic Viability. A standard training scheme requires safe and unsafe labels to constrain the sign of the learned barrier and exclude physical collision states (more discussion in Appendix A). However, these labels alone do not identify the controlled-invariant boundary. Recoverability depends on the full state, including velocity, orientation, and control authority, so states with similar projected obstacle geometry may require substantially different interventions. The desired barrier boundary is therefore a dynamical frontier inside the physically admissible region rather than simply the obstacle boundary.

Boundary Placement and Direction Are Underconstrained. Even when the learned barrier correctly separates sampled safe and unsafe states, many scalar fields can satisfy the same sign constraints while having different zero-level-set shapes and gradients. Sampled derivative losses therefore may produce a locally feasible fit without recovering the boundary placement or normal direction required by the actual dynamics. This motivates explicitly anchoring both *where* the boundary should lie and *how* its gradient should be oriented. NBF-PAS obtains this information from dynamically feasible recovery witnesses identified through the projected safety geometry.

4 Learning with Projected Anchor Set

Our proposed method Neural Barrier Function-Projected Anchor Set (NBF-PAS) focuses on addressing two main questions: ***where** to **anchor** a feasible barrier function* and ***how** to **anchor** the proper gradient direction on the boundary*.

4.1 Nagumo's Theorem and Forward Invariance

A closed set $\mathcal{C}$ is forward control invariant if, for every $\mathbf{x}_0 \in \mathcal{C}$, there exists an admissible control such that the resulting trajectory remains in $\mathcal{C}$ for all future time. The admissible velocities at $\mathbf{x}$ form the set $\mathcal{V}(\mathbf{x}) = \{f(\mathbf{x}, \mathbf{u}) : \mathbf{u} \in \mathcal{U}\}$. For a closed set $\mathcal{C}$, the Bouligand tangent cone is

$$T_{\mathcal{C}}(\mathbf{x}) = \left\{ v \in \mathcal{V} : \liminf_{h \downarrow 0} \frac{\operatorname{dist}(\mathbf{x} + hv, \mathcal{C})}{h} = 0 \right\}. \tag{5}$$

Intuitively, $T_{\mathcal{C}}(\mathbf{x})$ contains the instantaneous directions that stay inside $\mathcal{C}$ or move tangent to its boundary to first order. Under the standard regularity conditions of Nagumo's theorem, forward control invariance requires $\mathcal{V}(\mathbf{x}) \cap T_{\mathcal{C}}(\mathbf{x}) \neq \emptyset$. If $\mathcal{C} = \{\mathbf{x} : B(\mathbf{x}) \geq 0\}$ with $B \in C^1$ and $\nabla B(\mathbf{x}) \neq 0$ on $\partial\mathcal{C}$, then $T_{\mathcal{C}}(\mathbf{x}) = \left\{ v : \nabla B(\mathbf{x})^\top v \geq 0 \right\}$. Hence, this condition recovers equation 2 evaluated at the boundary:

$$\max_{\mathbf{u} \in \mathcal{U}} \nabla B(\mathbf{x})^\top f(\mathbf{x}, \mathbf{u}) \geq 0, \qquad \mathbf{x} \in \partial\mathcal{C}. \tag{6}$$

Geometrically, $\nabla B(\mathbf{x})$ points toward increasing barrier values, i.e., into the candidate safe set. Thus, the Nagumo boundary condition depends not only on where the zero level set is placed, but also on whether its local normal is compatible with an admissible system motion. This motivates using feasible recovery trajectories to provide supervision for both boundary placement and orientation.

4.2 Design Rationale for Projected Anchor Sets

Nagumo's condition identifies two requirements for learning a safety boundary: locating states where intervention is needed, and finding admissible velocities that remain tangent to or point into the candidate safe set. Projected anchor sets connect these requirements to observable recovery behavior. The projection supplies a geometric criterion for searching for recovery, while full-state anchors retain the dynamic information needed to fit the barrier.

Projected safety and anchoring boundary location. Although the system evolves in the full state space, physical safety is often determined by a lower-dimensional geometric representation. Let $\kappa : \mathcal{D} \to \mathbb{R}^d$, $d \leq n$, denote a safety projection, and let $g : \mathbb{R}^d \to \mathbb{R}$ be a signed geometric safety margin. The corresponding collision-free and collision sets in the full state space are

$$\mathcal{X}_{\text{free}} = \{\mathbf{x} \in \mathcal{D} : g(\kappa(\mathbf{x})) \geq 0\}, \qquad \mathcal{X}_{\text{col}} = \{\mathbf{x} \in \mathcal{D} : g(\kappa(\mathbf{x})) < 0\}. \tag{7}$$

For an obstacle centered at c_{obs} with radius r_{obs}, for example, $g(\kappa(\mathbf{x})) = \|\kappa(\mathbf{x}) - c_{\text{obs}}\|_2 - r_{\text{obs}}$. At a regular boundary point of $\mathcal{X}_{\text{free}}$, Nagumo's condition requires the system motion to lie within the tangent cone of the collision-free set. In $\mathcal{X}_{\text{free}}$, this condition is associated with the local normal toward increasing safety margin. For the circular and spherical obstacles, this direction is the outward obstacle normal. We therefore use projected motion relative to this normal as the geometric evidence for recovery, while retaining the full state because recoverability may also depend on velocity, orientation, and other unprojected variables. For sampled rollouts, we write

$$\mathbf{x}_{t+1} = F_\Delta(\mathbf{x}_t, \mathbf{u}_t), \qquad \mathbf{x}^+ = F_\Delta(\mathbf{x}, \mathbf{u}) \tag{8}$$

for the transition over a sampling interval Δ under a held input. We approximate the projection derivative with finite projected displacement $\kappa(\mathbf{x}^+) - \kappa(\mathbf{x})$ and its alignment with the outward

obstacle normal, which motivates the recovery score in equation 16 defined in the appendix F. A successful recovery supplies both full-state boundary candidates and admissible transition directions. Anchoring $B_\theta(\mathbf{x})$ near zero at these candidates constrains boundary placement, while recovery directions guide its local orientation. Under sufficient smoothness, the connection to equation 6 follows from the full-state expansion

$$B_\theta(F_\Delta(\mathbf{x},\mathbf{u})) - B_\theta(\mathbf{x}) = \Delta \nabla B_\theta(\mathbf{x})^\top f(\mathbf{x},\mathbf{u}) + O(\Delta^2). \tag{9}$$

If an anchor lies on the zero level set and its successor remains on the nonnegative side of the barrier, then the corresponding finite transition is consistent with the Nagumo boundary condition for sufficiently small sampling intervals. This provides a local geometric interpretation of the recovery supervision. However, projected outward motion alone does not establish full-state tangency or forward invariance; certification still requires the CBF condition in equation 6 to hold over the learned boundary.

Backtracking and high-order control effects. For a geometric constraint with high relative degree, the control appears only after repeated time differentiation. In model-based settings, high-order CBFs address this by recursively constructing derivative constraints that account for the intermediate dynamics (Xiao & Belta, 2022). A control that eventually prevents collision may therefore leave the system moving toward the obstacle initially. In the model-free setting that we consider, we backtrack on trajectories to move recovery-search seed earlier along a failed trajectory, testing alternative controls over multiple optimized forward steps. Successful recovery transitions then supply full-state anchor candidates with the resulting dynamic margins. This is a data-driven counterpart to high-order CBF reasoning: rollouts expose the effects of the intermediate dynamics without explicitly constructing their Lie derivatives. These considerations motivate the algorithm below: collect failure at near-obstacle seeds, backtrack to search for feasible recoveries, retain full-state anchors with projected geometric coverage, and use their locations and recovery directions to train and refine the barrier.

4.3 Algorithms

We implement the design in Section 4.2 through sampling, recovery search, and barrier refinement. Appendix F gives the full construction and training procedure.

Sampling and recovery. With the nominal controller π_{nom}, we collect N trajectories, split equally into collision trajectories $\mathcal{T}_{\mathrm{col}}$ and regular trajectories $\mathcal{T}_{\mathrm{reg}}$, and terminate collision rollouts at first contact. Recovery seeds are the transitions into $\mathcal{X}_{\mathrm{col}}$ and regular transitions whose successors satisfy $d_{\mathrm{unsafe}} \le \|\kappa(\mathbf{x}^+) - c_{\mathrm{obs}}\|_2 \le d_{\mathrm{unsafe}} + \epsilon$. The union of these seeds is $\mathcal{B}_{\mathrm{col}} \cup \mathcal{B}_{\mathrm{reg}}$. We also collect safe and unsafe samples using $\|\kappa(\mathbf{x}) - c_{\mathrm{obs}}\|_2 \ge d_{\mathrm{safe}}$ and $\|\kappa(\mathbf{x}) - c_{\mathrm{obs}}\|_2 < d_{\mathrm{unsafe}}$, respectively.

For a proposed transition, let $d = \kappa(F_\Delta(\mathbf{x},\mathbf{u})) - \kappa(\mathbf{x})$ and $r = \kappa(\mathbf{x}) - c_{\mathrm{obs}}$. The recovery score is the cosine similarity $\rho(\mathbf{x},\mathbf{u}) = d^\top r/(\|d\|_2\|r\|_2)$, expanded in equation 16. Write $R(\mathbf{x}) = (\mathbf{u}^\star, F_\Delta(\mathbf{x},\mathbf{u}^\star))$, where $\mathbf{u}^\star \in \arg\max_{\mathbf{u}\in\mathcal{U}} \rho(\mathbf{x},\mathbf{u})$. Algorithm 1a backtracks from each seed and advances until recovery meets the threshold η. Each accepted record stores $(\mathbf{x}, \mathbf{u}^\star, \mathbf{x}^+, \kappa(\mathbf{x}^{++}) - \kappa(\mathbf{x}^+))$, retaining the full state and a feasible projected direction.

Anchor set thinning. For each anchor A_i at state a_i, let $\delta_i = \kappa(a_i) - c_{\mathrm{obs}}$, $d_i = \|\delta_i\|_2$, and σ_i be its recovery score. Algorithm 1b partitions directions δ_i/d_i into M regions and keeps at most K anchors per region, ordered by increasing d_i and then decreasing σ_i. This limits dataset growth while retaining projected coverage; Figure 10 in the appendix illustrates the construction.

Barrier training with iterative refinement. Let $[z]_+ = \max\{z, 0\}$. The sign losses encourage positive values on $\mathcal{D}_{\mathrm{safe}}$ and negative values on $\mathcal{D}_{\mathrm{unsafe}}$:

$$\mathcal{L}_{\mathrm{safe}}(\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{\mathrm{safe}}}\left[\left[\epsilon_1 - B_\theta(\mathbf{x})\right]_+^2\right], \qquad \mathcal{L}_{\mathrm{unsafe}}(\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{\mathrm{unsafe}}}\left[\left[B_\theta(\mathbf{x}) + \epsilon_2\right]_+^2\right].$$

For an anchor transition $(\mathbf{x}, z)$ with $z = \mathbf{x}^+$, define the outward obstacle normal $\hat{\mathbf{n}}(z) = (\kappa(z) - c_{\mathrm{obs}})/\|\kappa(z) - c_{\mathrm{obs}}\|_2$ and transition direction $\mathbf{v}(\mathbf{x}, z) = (\kappa(z) - \kappa(\mathbf{x}))/\|\kappa(z) - \kappa(\mathbf{x})\|_2$. Let $\mathbf{n}_\theta(z) = \nabla_\kappa B_\theta(z)/\|\nabla_\kappa B_\theta(z)\|_2$ denote the normalized projected barrier normal. The anchor loss places z on the zero level set and aligns its normal with the obstacle and recovery directions:

$$\mathcal{L}_{\mathrm{anchor}}(\theta) = \mathbb{E}_{(\mathbf{x},z)\sim\mathcal{D}_{\mathrm{anchor}}}\left[B_\theta(z)^2 + \left[-\hat{\mathbf{n}}(z)^\top \mathbf{n}_\theta(z)\right]_+^2 + \left[-\mathbf{v}(\mathbf{x},z)^\top \mathbf{n}_\theta(z)\right]_+^2\right].$$

Algorithm 1 Projected anchor construction: recovery search and geometric thinning

(a) Recovery search

Require: Seeds, trajectories, stride Δ, threshold η
$\mathcal{D}_{\text{anchor}} \leftarrow \emptyset$
for each seed (τ, t_s) **do**
  for $k = t_s - \Delta, t_s - 2\Delta, \ldots$ **do**
    $\mathbf{x} \leftarrow \mathbf{x}_k$
    while $\mathbf{x}$ remains feasible **do**
      $(\mathbf{u}^\star, \mathbf{x}^+) \leftarrow R(\mathbf{x})$
      if $\rho(\mathbf{x}, \mathbf{u}^\star) \geq \eta$ **then**
        $(\mathbf{u}^{+\star}, \mathbf{x}^{++}) \leftarrow R(\mathbf{x}^+)$
        Add the recovery record to $\mathcal{D}_{\text{anchor}}$
        break
      end if
      $\mathbf{x} \leftarrow \mathbf{x}^+$
    end while
    if recovery found **then break**
  end for
end for
return $\mathcal{D}_{\text{anchor}}$

(b) Anchor thinning

Require: Anchors $\mathcal{D}_{\text{anchor}}$, region count M, cap K
Partition directions into M regions
$\mathcal{C}_m \leftarrow \emptyset$ for $m = 1, \ldots, M$
$\widehat{\mathcal{D}_{\text{anchor}}} \leftarrow \emptyset$
for each anchor $A_i \in \mathcal{D}_{\text{anchor}}$ **do**
  $\delta_i \leftarrow \kappa(a_i) - c_{\text{obs}}$
  $d_i \leftarrow \|\delta_i\|_2$
  $m \leftarrow$ region containing δ_i / d_i
  $\mathcal{C}_m \leftarrow \mathcal{C}_m \cup \{A_i\}$
end for
for $m = 1, \ldots, M$ **do**
  Sort $\mathcal{C}_m$ by $(d_i, -\sigma_i)$
  $k_m \leftarrow \min\{K, |\mathcal{C}_m|\}$
  $\mathcal{S}_m \leftarrow$ first k_m records in $\mathcal{C}_m$
  $\widehat{\mathcal{D}_{\text{anchor}}} \leftarrow \widehat{\mathcal{D}_{\text{anchor}}} \cup \mathcal{S}_m$
end for
return $\widehat{\mathcal{D}_{\text{anchor}}}$

We minimize a weighted sum of $\mathcal{L}_{\text{safe}}$, $\mathcal{L}_{\text{unsafe}}$, and $\mathcal{L}_{\text{anchor}}$. At iteration j, we roll out the nominal controller guarded by $B_\theta^{(j)}$ and select transitions satisfying $0 < B_\theta^{(j)}(\mathbf{x}_{t+1}) \leq \tau_h$. For those with $\rho(\mathbf{x}_t, \mathbf{u}_t) < \eta$, we repeat recovery search, add the successful witnesses to $\mathcal{D}_{\text{anchor}}$, thin the updated set, and retrain with the same losses.

5 Experiments

Our experiments primarily examine how projected recovery feedback is incorporated into neural-barrier learning for high-dimensional system and whether the resulting barrier improves the safety of a given nominal policy. Across four systems, we track how this process anchors the zero level set in full state space from geometric information obtained in lower dimension, and whether it reduces collisions without preventing progress toward the task goal. On Hopper, we additionally compare against a policy trained for collision avoidance directly through reinforcement learning.

5.1 Experiment Setup

We assume a queryable black-box simulator that returns the successor state $\mathbf{x}^+$ and collision status for each state–action pair $(\mathbf{x}, \mathbf{u})$. Disturbances are excluded, and the nominal policy π_{nom} is treated as a black box without assumed collision avoidance.

We evaluate four goal-reaching systems of increasing complexity (Table 3), using κ for collision checking and recovery search while the barrier and policy operate on the full state. The 3D **Dubins** vehicle has constant forward speed and bounded turn rate, avoids a circular obstacle, and uses planar position $\kappa(\mathbf{x}) = (x, y)$. The 4D linearized **Segway** has bounded horizontal actuation and a circular obstacle, with head-position projection $\kappa(\mathbf{x}) = (p + L\sin\phi, L\cos\phi)$. The 12D rigid-body **quadrotor** has bounded thrust and body torques, avoids a spherical obstacle, and uses $\kappa(\mathbf{x}) = (p_x, p_y, p_z)$. The 12D MuJoCo **Hopper-v5** has bounded joint torques and a circular obstacle in the sagittal (x, z) plane; its projection is the closest point on the body geometry, $\kappa(\mathbf{x}) \in \arg\min_{\mathbf{p} \in \mathcal{G}(\mathbf{q})} \|\mathbf{p} - c_{\text{obs}}\|_2$. We report falls and unhealthy terminations for Segway and Hopper. Detailed dynamics, state definitions, control bounds, environment parameters, and network setting appear in the appendix B.

5.1.1 Nominal Controllers and Baseline

Nominal Controllers. All four systems' nominal policies are trained using PPO before barrier learning begins. Segway and quadrotor use policy networks with two hidden layers of 128 units, while Hopper uses two hidden layers of 64 units.

Baseline. We compare against direct neural-CBF training using the same safe/unsafe data, network architecture, and training budget. The baseline, instead of handling the projected anchor set, min-

Table 1: Closed-loop collision and fall rates. For Dubins and quadrotor, R_{norm} is the fraction of near-boundary states ($|B| \leq \tau_h$) where the unit barrier normal decreases true obstacle clearance. For Segway and Hopper, we report fall rate instead.

System	Method	Collision rate	R_{norm}	System	Method	Collision rate	Fall rate
Dubins	Nominal	15%	–	Segway	Nominal	100.00%	0.00%
Dubins	Baseline	0.6%	9.9%	Segway	Baseline	0.00%	100.0%
Dubins	NBF-PAS	0%	3.6%	Segway	NBF-PAS	0.00%	3.10%
Quadrotor	Nominal	29.5%	–	Hopper	Nominal	75.10%	7.70%
Quadrotor	Baseline	1.75%	22.3%	Hopper	Baseline	1.20%	98.70%
Quadrotor	NBF-PAS	0.00%	0.0%	Hopper	NBF-PAS	0.00%	100.00%

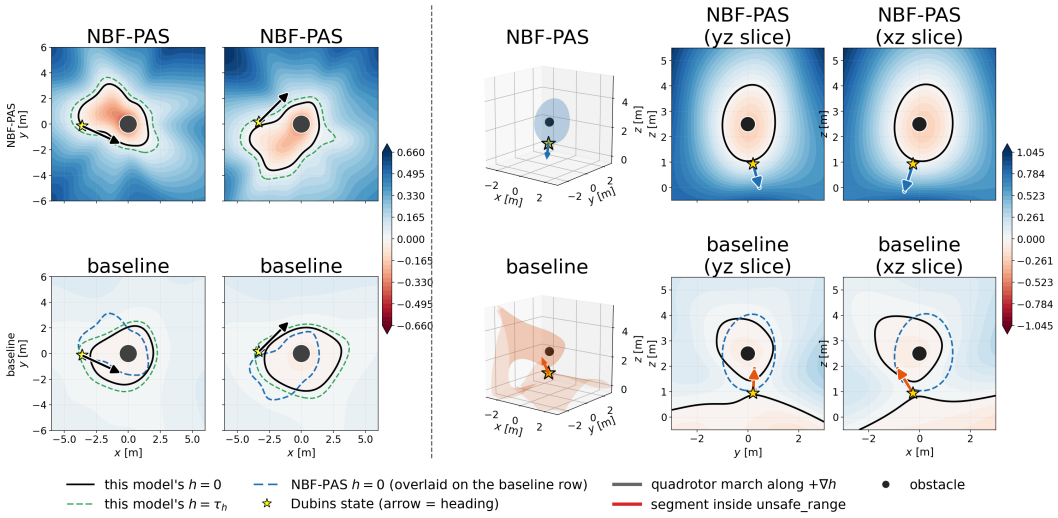


Figure 2: **Barrier geometry comparison between NBF-PAS and the baseline.** Left: Dubins barrier slices at two headings with zero level sets and heading directions. Right: quadrotor zero-level-set geometry and normalized barrier normals at a near-boundary state.

imizes a sampled CBF-feasibility loss over intermediate states and iteratively adds near-boundary violations during retraining. Full objectives and update rules are given in Appendix B.2.

Safety Filters. At deployment, we activate the barrier filter only near the learned boundary and at states sufficiently supported by the training data; data support is estimated using Gaussian Process (GP) posterior variance, with details in Appendix B.3.

5.2 Main Results

Table 1 reports normal regression for Dubins and quadrotor, where obstacle clearance provides a meaningful local measure of safety, and fall rate for the balance-critical Segway and Hopper, where safety interventions can instead induce loss of balance. Normal regression is the fraction of near-boundary states ($|B| \leq \tau_h$) at which moving along the unit barrier normal decreases true obstacle clearance. Although the Dubins and quadrotor baselines achieve low collision rates, their substantially higher normal regression rates, together with Figure 2, reveal less reliable learned boundary geometry. For Dubins, the baseline remains nearly concentric with the obstacle and fails to capture the upstream shift induced by the nonholonomic dynamics, whereas NBF-PAS shifts the unsafe region opposite the direction of travel. For the quadrotor, the baseline normal provides less reliable safety-direction guidance, while NBF-PAS aligns more consistently with increasing clearance. At the examined state, the position-restricted Hessian is also positive definite for NBF-PAS but indefinite for the baseline, further indicating different local boundary geometry. Figures 3

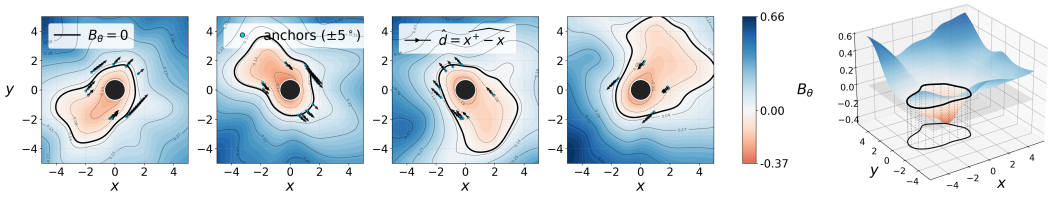


Figure 3: **NBF-PAS Dubins barrier and projected anchors.** Barrier slices show anchors and recovery directions aligned with the learned boundary; the right panel visualizes the corresponding barrier surface at $\theta = 0$.

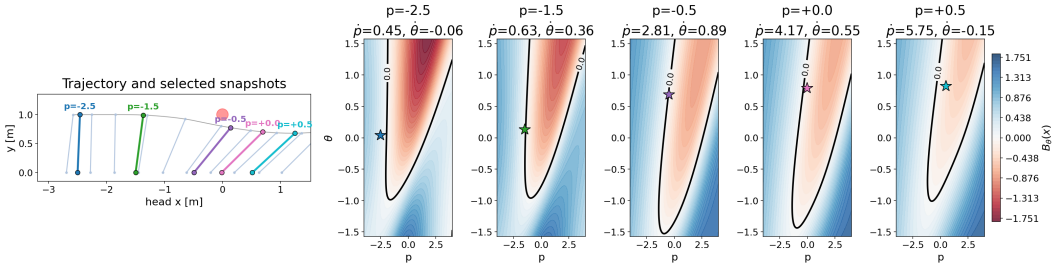


Figure 4: CBF level set $B_\theta(p, \theta)$ along a successful crossing, with $(\dot{p}, \dot{\theta})$ fixed to the segway's actual velocity at each snapshot (left). The black curve is B_θ=0; stars mark the true (p, θ) state. The safe region contracts and shifts as the segway accelerates.

and 4 further visualize how the learned boundary and recovery directions vary with the system state and dynamics.

5.2.1 Recovery of Reach Behaviors

Barrier interventions can drive a nominal RL policy outside its training distribution, while collisions are avoided, and reduce task performance even when collisions are avoided, such as the high fall rates reported in Table 1. We therefore examine how the original reaching behavior can be recovered while preserving the learned safety intervention.

Segway reach recovery. For Segway, the nominal controller can be used directly with the safety filter: the CBF intervenes minimally in regions with sufficient data support and is disabled once GP uncertainty exceeds the trust threshold (Fig. 5). Additional ablations on barrier-network capacity and the resulting safe-set conservatism are provided in Appendix D.

Comparison with Pure Reinforcement Learning (RL) on Hopper. For the higher-dimensional Hopper system, direct CBF intervention can induce a larger distribution shift and substantially degrade the nominal policy's reaching behavior. We therefore retrain the control policy, warm-started from the nominal policy, so that it can recover the task behavior while adapting to the safety constraints imposed by the learned barrier.

We compare this procedure against pure RL using the same initialization and training budget. Pure RL must discover collision-avoiding behavior through the environment reward, whereas NBF-PAS additionally uses the barrier-based auxiliary loss described in Appendix B.4. This loss is evaluated at currently visited safety-critical states and provides a local direction for modifying the policy toward actions that preserve the barrier, thereby guiding policy search rather than relying only on delayed collision penalties. As shown in Fig. 6 and Table 2, this guidance rapidly suppresses collisions while the policy continues to improve progress and goal-reaching performance. Pure RL, in contrast, retains a much higher collision rate and recovers the task more slowly.

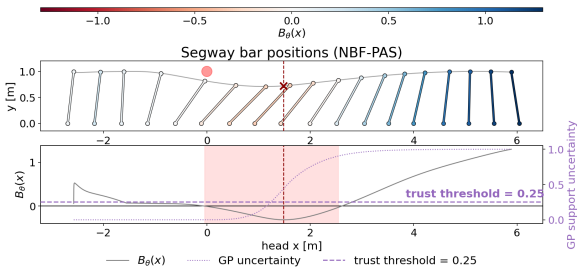


Figure 5: Segway rollout with the nominal controller guarded by NBF-PAS.

Method	Collision rate	Goal success	Fall rate
Nominal	75.1%	17.2%	7.7%
NBF-PAS (pre)	0.0%	0.0%	100.0%
NBF-PAS (post)	0.7%	89.3%	9.3%
Pure RL	34.7%	49.7%	15.7%

Table 2: Hopper policy-recovery performance.

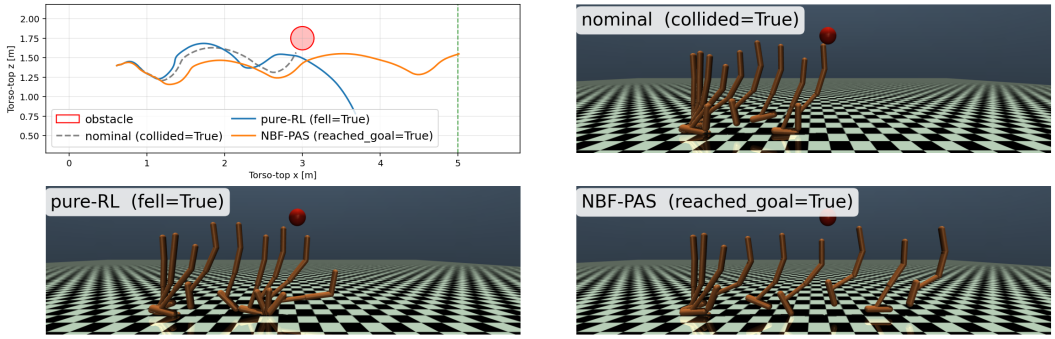


Figure 6: **Trajectory comparision among three modes**

6 Conclusion

NBF-PAS reorganizes neural-barrier learning around states where interventions change safety outcomes. Low-dimensional safety geometry provides an interpretable recovery objective, while full-state anchors retain the dynamic information needed to shape the learned safe set. Across four systems, NBF-PAS improves learned boundary geometry and closed-loop collision avoidance, and barrier-guided policy recovery further restores task performance under safety intervention. These results show that successful recoveries can provide useful boundary evidence and policy-improvement targets without relying on an external safe-action teacher. At the same time, the method transfers rather than removes the difficulty of safety learning. It requires counterfactual rollout access, sufficiently rich control proposals, adequate recovery horizons, and coverage of dynamically distinct full states. Search may also become expensive in high-dimensional action spaces, while projection or model bias can produce misleading recovery evidence. Finally, finite recovery and empirical loss reduction do not by themselves establish invariance, maximality of the learned set, convergence of alternating optimization, or preservation of task performance. These remain distinct claims that require separate verification or empirical evidence.

AI use statement

In this work, we used generative AI tools to aid and polish writing and to assist with retrieval and discovery of related work. The authors conducted the literature review independently and used AI tools as an additional check for potentially relevant work that may have been missed. All suggested references were manually verified against the original sources. AI tools were not used to draft and revise portions of the manuscript. We did not use generative AI tools to generate synthetic datasets or prove mathematical claims. The authors take responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

References

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European*

control conference (ECC), pp. 3420–3431. Ieee, 2019.

Jason Choi, Fernando Castaneda, Claire J Tomlin, and Koushil Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. *arXiv preprint arXiv:2004.07584*, 2020.

Jason J. Choi, Donggun Lee, Koushil Sreenath, Claire J. Tomlin, and Sylvia L. Herbert. Robust control barrier-value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 6814–6821, 2021. doi: 10.1109/CDC45484.2021.9683085.

Max H Cohen, Tamas G Molnar, and Aaron D Ames. Safety-critical control for autonomous systems: Control barrier functions via reduced-order models. *Annual Reviews in Control*, 57:100947, 2024.

Bolun Dai, Prashanth Krishnamurthy, and Farshad Khorrami. Learning a better control barrier function. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 945–950. IEEE, 2022.

Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023.

Bhavya Giri Goswami, Manan Tayal, Karthik Rajgopal, Pushpak Jagtap, and Shishir Kolathaya. Collision cone control barrier functions: Experimental validation on ugvs for kinematic obstacle avoidance. In *2024 American Control Conference (ACC)*, pp. 325–331. IEEE, 2024.

Mohammad Javad Khojasteh, Vikas Dhiman, Massimo Franceschetti, and Nikolay Atanasov. Probabilistic safety constraints for learned high relative degree system dynamics. In *Learning for Dynamics and Control*, pp. 781–792. PMLR, 2020.

Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3717–3724. Ieee, 2020.

Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11532–11539, 2024. doi: 10.1109/ICRA57147.2024.10610418.

Manan Tayal, Rajpal Singh, Jishnu Keshavan, and Shishir Kolathaya. Control barrier functions in dynamic uavs for kinematic obstacle avoidance: A collision cone approach. In *2024 American Control Conference (ACC)*, pp. 3722–3727. IEEE, 2024.

Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. Learning for safety-critical control with control barrier functions. In *Learning for dynamics and control*, pp. 708–717. PMLR, 2020.

Chuanzheng Wang, Yiming Meng, Yinan Li, Stephen L Smith, and Jun Liu. Learning control barrier functions with high relative degree for safety-critical control. In *2021 European Control Conference (ECC)*, pp. 1459–1464. IEEE, 2021.

Wei Xiao and Calin Belta. High-order control barrier functions. *IEEE Transactions on Automatic Control*, 67(7):3655–3662, 2022. doi: 10.1109/TAC.2021.3105491.

A WHY UNSAFE SUPERVISION IS NEEDED IN NEURAL BARRIER LEARNING

For an analytical CBF, the unsafe set need not appear explicitly in the CBF condition: once a valid safe set

$$\mathcal{C}_\theta = \{\mathbf{x} : B_\theta(\mathbf{x}) \geq 0\}$$

is constructed and shown to satisfy $\mathcal{C}_\theta \subseteq \mathcal{X}_{\text{free}}$, forward invariance of $\mathcal{C}_\theta$ is sufficient for safety. In neural barrier learning, however, B_θ is a flexible function fitted from finite data. Derivative constraints alone do not determine where its zero level set should lie or which side should contain physically collided states. We therefore use unsafe samples to impose the sign constraint

$$B_\theta(\mathbf{x}) < 0, \qquad \mathbf{x} \in \mathcal{D}_{\text{unsafe}},$$

which helps enforce the required containment $\mathcal{C}_\theta \subseteq \mathcal{X}_{\text{free}}$.

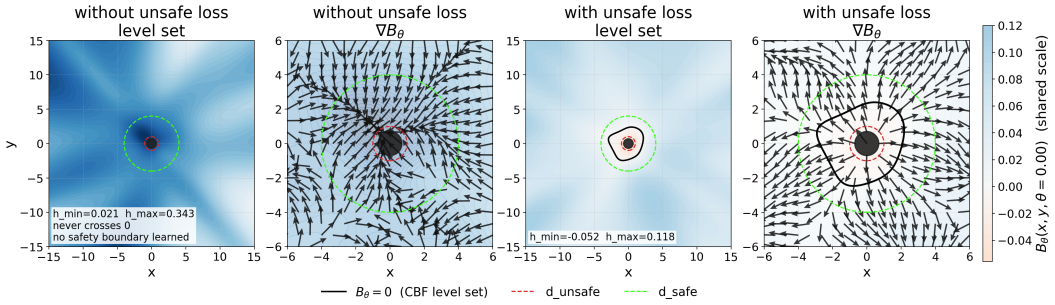


Figure 7: **Effect of unsafe sign supervision.** Shown is the CBF for a Dubins car avoiding an obstacle (black), sliced at $\theta = 0$ without (left) and with (right) unsafe loss.

Figure 7 illustrates this role. Adding unsafe supervision produces a boundary that excludes the obstacle, but the resulting zero level set can still have incorrect geometry or gradient orientation. Thus, unsafe labels provide the basic physical separation needed to train the neural representation, while the projected recovery anchors provide the additional dynamical supervision used by NBF-PAS.

B EXPERIMENT SETUP DETAILS

B.0.1 EXPERIMENT SYSTEMS SPECIFICS

Dubins. We model the Dubins car as a planar nonholonomic system with state $\mathbf{x} = (x, y, \theta)$, where (x, y) denotes the position and θ denotes the heading. The vehicle moves at a constant forward speed $v = 4$, and the controllable action input is the constrained angular velocity $\mathbf{u} = \omega$ where $|\omega| \leq 1.5$. Since the Dubins vehicle is represented as a point for collision checking, the safety projection extracts its planar position: $\kappa : \mathbb{R}^3 \to \mathbb{R}^2, \kappa(\mathbf{x}) = (x, y)$.

Segway. We model the Segway as a linearized cart-pole system with state $\mathbf{x} = (p, \phi, \dot{p}, \dot{\phi})$, where p is the base position, ϕ is the body tilt angle, $\dot{p}$ is the base velocity, and $\dot{\phi}$ is the angular velocity. The control input is a one-dimensional bounded horizontal force $\mathbf{u} = F \in \mathcal{U}$ applied at the base. For this system, we consider a collision-avoidance environment, where the body of the Segway must never hit or intersect at $(0, 1)$ with $r_{\text{obs}} = 0.1$. The safety projection extracts its head position: $\kappa : \mathbb{R}^4 \to \mathbb{R}^2, \kappa(\mathbf{x}) = (p + L \sin\phi, L\cos\phi)$, where L is the body length.

Quadrotor. We model the quadrotor as a 12-dimensional rigid-body system with state $\mathbf{x} = (p_x, p_y, p_z, v_x, v_y, v_z, \phi, \theta, \psi, \omega_x, \omega_y, \omega_z)$, where $\mathbf{p} = (p_x, p_y, p_z)$ and $\mathbf{v} = (v_x, v_y, v_z)$ are the position and linear velocity, (ϕ, θ, ψ) are the roll, pitch, and yaw angles, and $(\omega_x, \omega_y, \omega_z)$ are the body angular velocities. The bounded control input is $\mathbf{u} = (T, \tau_x, \tau_y, \tau_z) \in \mathcal{U}$, consisting of the total thrust and three body torques. We consider a spherical obstacle centered at $(0, 0, 2.5)$ with radius $r_{\text{obs}} = 0.3$. The safety projection extracts its 3D position: $\kappa : \mathbb{R}^{12} \to \mathbb{R}^3, \kappa(\mathbf{x}) = (x, y, z)$.

Table 3: **Experimental systems.** The safety projection detects collisions and evaluates recovery, while the barrier and policy operate on the full state.

Property	Dubins	Segway	Quadrotor	Hopper
State dimension	3	4	12	12
Action dimension	1	1	4	3
Sampling interval Δt	0.05	0.01	0.1	0.008
Dynamics	Analytic unicycle	Analytic	Analytic	MuJoCo
Nominal policy	PPO	PPO	PPO	PPO
Obstacle (c_{obs}, r)	$((0,0), 0.7)$	$((0,1), 0.1)$	$((0,0,2.5), 0.3)$	$((3.0, 1.75), 0.15)$
Safe/unsafe sampling range	$4.0/1.0$	$2.0/0.1$	$1.6/0.8$	$2.0/0.3$
Safety projection κ	(x, y)	Head position	(x, y, z)	Closest geom

Table 4: **Projected recovery parameters.** The backtracking stride specifies the spacing between tested states on the original trajectory.

Parameter	Dubins	Segway	Quadrotor	Hopper
Collected trajectories	200	1000	10000	3000
Rollout horizon	–	800	550	300
Acceptance threshold $\sigma_{\min}$	-0.85	-0.40	-0.40	-0.40
Backtracking stride	3	5	1	5
Maximum backtracks	10	12	15	15
Recovery budget $H_{\max}$	60	100	60	60
Action samples	200	200	150	50
Angular bins	180	1200	500	1300
Maximum anchors per bin	2	3	2	2
Safe/unsafe/anchor targets	$500/1000/500$	$500/1000/500$	$1000/2000/1000$	$500/1000/500$

Hopper. We model Hopper using the nonlinear contact dynamics of the MuJoCo `Hopper-v5` environment. Its state is $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{12}$, where $\mathbf{q} \in \mathbb{R}^6$ contains the horizontal position, torso height, body angle, and three joint angles. The bounded control input $\mathbf{u} \in [-1, 1]^3$ specifies the three joint torques. We place an obstacle at $(3.0, 1.75)$ in the sagittal (x, z) plane with radius $r_{\text{obs}} = 0.15$. We separately classify a rollout as unhealthy when the torso height falls below 0.7 or the body angle leaves $[-0.2, 0.2]$. Let $\mathcal{G}_j(\mathbf{q}) \subset \mathbb{R}^2$ denote the region occupied by the jth Hopper body geometry in the sagittal (x, z) plane, and define $\mathcal{G}(\mathbf{q}) = \bigcup_{j=1}^{N_{\text{geom}}} \mathcal{G}_j(\mathbf{q})$. The safety projection extracts the point on the Hopper body geometry closest to the obstacle center:$\kappa : \mathbb{R}^{12} \to \mathbb{R}^2, \kappa(\mathbf{x}) \in \arg,\min_{\mathbf{p} \in \mathcal{G}(\mathbf{q})} |\mathbf{p} - c_{\text{obs}}|_2, \mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$. Thus, $\kappa(\mathbf{x})$ is explicitly the (x, z) position of the closest Hopper geom.

B.1 Network Setup

All barriers use an MLP with three hidden layers of 512 units, $\tanh$ activations, and a scalar output, initialized with Xavier-uniform weights and zero biases. We train with Adam at learning rate 10^{-3} and batch size 1024 for Dubins, Segway, and quadrotor, or 512 for Hopper. Initial training stops at 300 epochs or upon reaching a loss threshold, followed by at most 200 refinement iterations.

B.2 Baseline

For each system, we collect offline dataset by rolling out with the nominal controller π_{nom}. States $\|\kappa(\mathbf{x}) - c_{\text{obs}}\| > d_{\text{safe}}$ are put in the safe data bin $\mathcal{D}_{\text{safe}}$, $\|\kappa(\mathbf{x}) - c_{\text{obs}}\| \leq d_{\text{unsafe}}$ in unsafe $\mathcal{D}_{\text{unsafe}}$, and in-betwen states are put in the lie $\mathcal{D}_{\text{lie}}$. The network is trained by solving the min-max-min problem by minimizing $\mathcal{L}_{\text{cbf}}$

$$\mathcal{L}_{\text{cbf}} = w_1 \mathcal{L}_{\text{safe}} + w_2 \mathcal{L}_{\text{unsafe}} + w_3 \mathcal{L}_{\text{lie}}$$

where

$$\mathcal{L}_{\text{lie}} = \mathbb{E}_{x\in\mathcal{D}_{\text{lie}}}\left[\ell\left(\sup_{\mathbf{u}\in\mathcal{U}}\left[\nabla B_\theta(\mathbf{x})^\top F(\mathbf{x},\mathbf{u}) + \alpha(B_\theta(\mathbf{x}))\right]\right)\right]$$

After initial barrier $B_\theta^{(0)}$ is learn, if there are still collisions, we update the barrier by sampling action wrt the $B_\theta^{(0)}$ and evaluate if the state transition $(\mathbf{x}_t, \mathbf{u}, \mathbf{x}_{t+1})$ near the 0 level set satisfies the CBF safety constraint. If not, we sample the best action $\mathbf{u}_t^\star$ and add the transition $(\mathbf{x}_t, \mathbf{u}_t^\star, \mathbf{x}_{t+1}^\star)$ to $\mathcal{D}_{\text{lie}}$. If an action that satisfies the barrier condition is found, we add this transiton to the set; otherwise, we add $\mathbf{x}_t$ to $\mathcal{D}_{\text{unsafe}}$.

B.3 Data-Supported Safety Filtering.

Before each action, we estimate data support using the posterior variance of a target-free GP fit to the standardized safe, unsafe, and anchor states Z:

$$\mathbf{u}(\mathbf{x}) = 1 - k_x^\top (K_{ZZ} + \epsilon I)^{-1} k_x \in [0,1],$$

where $u(x) \approx 0$ indicates strong data support and $u(x) \approx 1$ indicates extrapolation. The safety filter is activated only near the learned boundary and in supported regions,

$$\text{FILTER}(\mathbf{x}) = \big[B_\theta(\mathbf{x}) - \ell \le \alpha\big] \wedge \big[\mathbf{u}(\mathbf{x}) \le \tau_u\big], \qquad \tau_u = 0.5.$$

When activated, we apply

$$\mathbf{u}^\star = \arg\min_u \|u - u_{\text{nom}}\| \quad \text{s.t.} \quad m(\mathbf{x},\mathbf{u}) \ge 0,$$

and fall back to $\arg\max_u m(x,u)$ if no feasible candidate exists. Otherwise, the nominal action is executed unchanged.

B.4 Pure RL and NBF-PAS Setup Details

Both pure RL and NBF-PAS use the same nominal-policy initialization, environment, obstacle configuration, interaction budget, evaluation distribution, and reward design. They also share the same nominal-action imitation loss, which limits deviation from the original controller. The key difference is therefore the auxiliary safety signal used during policy retraining.

$$\mathcal{L}_{\text{nom}} = \mathbb{E}_{x\sim\mathcal{D}_{\text{rollout}}}\left[\|\mu_\phi(x) - \mu_{\text{nom}}(x)\|_2^2\right].$$

The recovery of the nominal controller has one more auxillary term that penalizes the barrier value decrease when the state is in activation margin and has sufficient data support:

$$\mathcal{L}_{\text{change}} = \mathbb{E}_{x_0\sim\mathcal{D}_{\text{active}}}\left[\text{ReLU}\left(-\Delta B(x_0) - \Delta t\,\gamma\big(B_\theta(x_0) - \ell\big)\right)^2\right],$$
$$\Delta B(x_0) = B_\theta(x_0 + \Delta t\,(f(x_0) + g(x_0)\mu_\phi(x_0))) - B_\theta(x_0).$$

C Boundary Guarantees and Their Assumptions

Section 4.1 gives the Nagumo boundary condition used to motivate our recovery-based supervision. Here we clarify the assumptions under which that condition implies forward invariance and what additional requirements are needed to infer it from finitely sampled boundary states.

Controlled boundary condition. Let

$$\mathcal{C} = \{\mathbf{x}\in\mathcal{D} : B(\mathbf{x}) \ge 0\}.$$

Under the standard regularity assumptions of Nagumo's theorem, including a closed candidate set, sufficiently regular dynamics, and admissible velocity sets satisfying the usual compactness and viability conditions, the boundary condition

$$\mathcal{V}(\mathbf{x}) \cap T_{\mathcal{C}}(\mathbf{x}) \ne \emptyset, \qquad \mathbf{x}\in\partial\mathcal{C},$$

implies controlled forward invariance. For a regular level-set boundary, Section **??** reduces this condition to equation 6. Thus, the theoretical requirement remains a full-state condition over the entire learned boundary; recovery witnesses provide sampled evidence toward this condition but do not establish it by themselves.

Fixed-policy boundary condition. For a learned policy μ_ϕ, define the closed-loop boundary derivative

$$q_\phi(\mathbf{x}) = \nabla B_\theta(\mathbf{x})^\top f(\mathbf{x}, \mu_\phi(\mathbf{x})). \tag{10}$$

If the closed-loop vector field $f(\mathbf{x}, \mu_\phi(\mathbf{x}))$ is locally Lipschitz and

$$q_\phi(\mathbf{x}) \geq 0, \qquad \forall \mathbf{x} \in \partial\mathcal{C}_\theta, \tag{11}$$

then the closed-loop velocity lies in $T_{\mathcal{C}_\theta}(\mathbf{x})$ at every regular boundary point. By the same Nagumo condition described in Section 4.1, $\mathcal{C}_\theta$ is forward invariant under μ_ϕ, provided solutions remain well defined for all future time. If additionally $\mathcal{C}_\theta \subseteq \mathcal{X}_{\text{free}}$, this implies physical safety.

C.1 From Sampled Boundary Checks to Certification

The practical method evaluates only finitely many recovery-supported states, whereas equation 11 must hold over the entire boundary. A sufficient coverage argument can be stated as follows. Assume q_ϕ is L_q-Lipschitz on $\partial\mathcal{C}_\theta$, and let exact boundary points $z_1, \ldots, z_N$ form an ε-net of $\partial\mathcal{C}_\theta$. Suppose estimates $\widehat{q}_\phi(z_i)$ satisfy

$$|\widehat{q}_\phi(z_i) - q_\phi(z_i)| \leq e_q, \qquad \widehat{q}_\phi(z_i) \geq L_q\varepsilon + e_q. \tag{12}$$

For any $\mathbf{x} \in \partial\mathcal{C}_\theta$, choose z_i such that $\|\mathbf{x} - z_i\|_2 \leq \varepsilon$. Then

$$q_\phi(\mathbf{x}) \geq q_\phi(z_i) - L_q\varepsilon \tag{13}$$

$$\geq \widehat{q}_\phi(z_i) - e_q - L_q\varepsilon \geq 0. \tag{14}$$

Hence equation 11 holds over the complete boundary, and under the regularity and containment assumptions above, the Nagumo condition implies forward invariance.

This argument is conditional and does not by itself certify the learned barrier. In particular, the recovered anchors need not lie exactly on $\partial\mathcal{C}_\theta$ or form an ε-net of the boundary. Coverage must also hold in the full state space, rather than only in the projected geometry $\kappa(\partial\mathcal{C}_\theta)$. Errors in boundary localization, dynamics, or finite-step estimates must therefore be included in the effective coverage and estimation bounds. These requirements are why we describe the practical procedure as barrier refinement rather than formal certification.

D Ablation on Network Capacity and Conservatism

We first examine how barrier-network capacity affects the initially learned safe set. We use Dubins and Segway for this ablation because their initial barriers can be evaluated directly without the iterative retraining loop, making the effect of architecture easier to isolate. We vary depth $d \in \{2, 5\}$ and width $w \in \{256, 1024\}$ with three seeds per architecture.

Figure 8 shows that increasing capacity substantially improves the Segway training fit, but a better fit does not necessarily yield a safer closed-loop barrier. The only architecture cells that fail certification are among the better-fitting models, including one model that collides in 81% of guarded rollouts. At the same time, all Segway models classify the sampled safe and unsafe states accurately, with balanced accuracy above 97.7%. Thus, the difference is not captured by training loss or sampled boundary classification alone.

Figure 9 reveals the main effect of increasing capacity. Better-fitting Segway models claim a systematically larger portion of the state space as safe. The safe-set volume increases from about 15% for the most under-fit architecture to about 33% for the best-fitting models, with a significant association between training loss and claimed safe-set volume ($\rho = -0.748$, $p = 0.005$). The same effect is visible near the obstacle. At 0.6 m clearance, the under-fit model accepts only about 6% of states, whereas the best-fitting model accepts about 32%. The under-fit barrier is therefore protected by a large conservative buffer, while improved fitting progressively reclaims states as safe and reduces this margin.

This explains why lower training loss need not monotonically improve guarded rollout safety. Additional capacity allows the barrier to fit the available supervision more closely and recover a larger safe set, which is desirable, but also leaves less conservative margin to absorb residual modeling and closed-loop errors. The ablation therefore motivates evaluating neural barriers not only by training fit, but also by the geometry and closed-loop behavior of the safe set they induce.

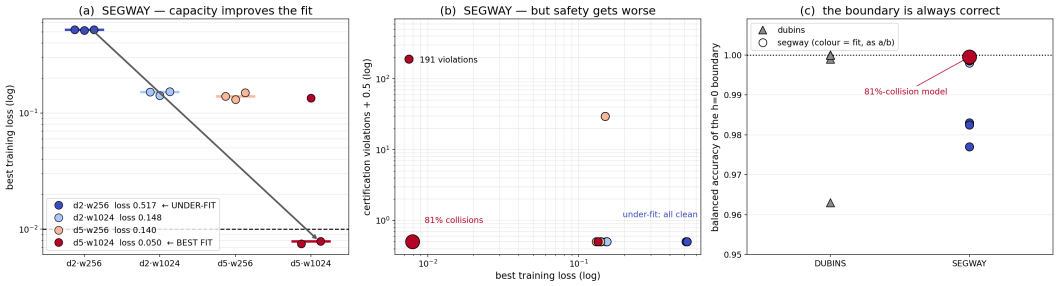


Figure 8: **Effect of barrier-network capacity on fit and safety.** Depth $d \in \{2, 5\}$ and width $w \in \{256, 1024\}$ are evaluated with three seeds per architecture. **(a)** Increasing Segway network capacity substantially reduces training loss. **(b)** Closed-loop certification does not improve monotonically with fit: the architecture cells exhibiting violations are among the better-fitting models, including one model with an 81% guarded-rollout collision rate. **(c)** All models nevertheless retain high balanced accuracy on the sampled safe/unsafe labels, showing that sample-wise classification quality alone does not capture the closed-loop difference.

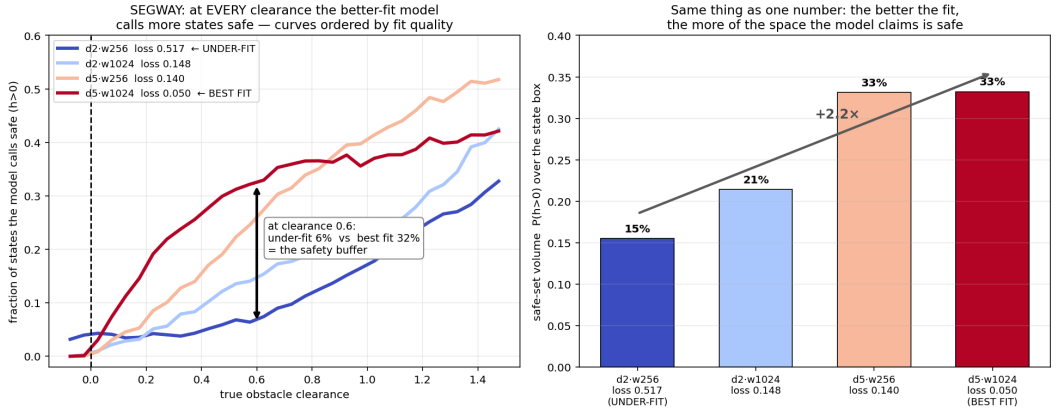


Figure 9: **Higher-capacity Segway barriers recover a larger safe set. Left:** Fraction of states classified as safe as a function of true obstacle clearance. Near the obstacle, better-fitting models admit substantially more states. At $0.6\,\mathrm{m}$ clearance, the accepted fraction grows from about 6% for the under-fit model to about 32% for the best-fitting model. **Right:** The overall claimed safe-set volume increases from about 15% to 33% as training fit improves. This reduction in conservatism explains why better fit can expose closed-loop errors that are masked by an under-fit barrier's larger safety buffer.

E Recovery Search and Data Management Details

A concrete derivative-free search. Choose a finite horizon schedule, a proposal count M, a number of refinement rounds, and an elite fraction before each search. At a given head, generate sequences around the current-policy rollout and retained recovery sequences, clip or parameterize them into $\mathcal{U}$, and evaluate equation 16. Retain the highest-scoring fraction, update the proposal distribution, and repeat within the budget. A physically validated successful proposal is retained with its complete state/action sequence, horizon, minimum clearance, terminal clearance, model version, and origin trajectory. Search budgets and success thresholds must be reported alongside any future empirical evaluation; no particular numerical settings are asserted here as validated.

Backtracking and nonmonotonicity. For a collision at index t_c, scan $t_c - 1, t_c - 2, \ldots, 0$, or use a coarse stride followed by sequential refinement. A successful state that precedes a failed tested state yields a candidate anchor bracket. A nominal trajectory can enter and leave recoverable regions, so this local bracket is not a globally ordered boundary. Retain multiple brackets when separate

hazards or different recovery modes are discovered. If recovery succeeds at every tested head before contact, retain the witnesses but do not infer a bracket from the physically colliding endpoint alone.

Replay and conflict resolution. Maintain distinct buffers for physically colliding samples, buffered recovery-supported interiors, anchors, witness transitions, and unresolved seeds. Exclude anchors from positive-margin interior labels. Before fitting, compare contradictory records at the same or nearby full states: preserve direct physical collision evidence, invalidate stale model-generated witnesses when their model changes, and re-run ambiguous recovery searches. If a previously excluded head now has a valid witness, remove its conservative exclusion loss. Negative labels from physical collision and provisional downstream exclusion are never conflated.

Boundary geometry without an ill-defined projected gradient. A general many-to-one observation $p(x)$ does not define a unique quantity $\nabla_p B_\theta(x)$. If a coordinate chart $x = \chi(y, z)$ is available, a conditional workspace derivative is well-defined as $\nabla_y B_\theta(\chi(y, z)) = D_y\chi(y, z)^\top \nabla_x B_\theta(x)$ at fixed hidden coordinates z. Without such a chart, directly aligning a "projected barrier normal" with an obstacle normal is ambiguous. Our losses instead evaluate the full-state field at observed transition states. For short transitions from an exact anchor a,

$$B_\theta(F_\Delta(a, u)) = \Delta \nabla B_\theta(a)^\top f(a, u) + O(\Delta^2), \tag{15}$$

under sufficient smoothness. Thus successor nonnegativity is related to a full-state tangent condition in the small-step limit, while for finite steps the exact successor and intermediate trajectory checks remain the appropriate objects.

Evaluation boundary. All buffers used to discover anchors or select stopping times belong to training or validation. A final held-out assessment should fix the barrier, policy, initial-state distribution, rollout horizon, and event definition before collecting trajectories. Collision frequency, intervention size, recovery-search success, and candidate-set exits measure different properties. No finite rollout statistic alone proves invariance over every state or over infinite time; the universal conditions in Section 4.1 require separate evidence.

F Detailed Algorithms and Training Procedure

We implement the design in Section 4.2 through the following sampling, recovery, and training steps.

Initial Sampling. We construct a *projected anchor set* from dynamically feasible recovery witnesses in addition to the safe and unsafe dataset with the black-box controller in hand. Given a nominal controller π_{nom}, we collect N trajectories $\tau^{(i)} = \{(\mathbf{x}_t^{(i)}, \mathbf{u}_t)\}_{t=1}^T$ with initial and goal states sampled from some distribution. These trajectories are partitioned into two subsets, $\mathcal{T}_{\text{col}}$ and $\mathcal{T}_{\text{reg}}$, each containing $N/2$ trajectories. $\mathcal{T}_{\text{col}}$ enters collision mode, and $\mathcal{T}_{\text{reg}}$ enters regular mode. By collision mode, each $\tau \in \mathcal{T}_{\text{col}}$ terminates at the first collided state.

Projected Anchor Set. Each anchor corresponds to a full state $\mathbf{x} \in \mathcal{D}$ from which a feasible recovery witness is found, while $\kappa(\mathbf{x})$ specifies its location in the projected safety geometry. We call the resulting collection a projected anchor set because the anchors are discovered and geometrically characterized through κ, while the anchors themselves retain the full-state information used to train $B_\theta : \mathcal{D} \to \mathbb{R}$.

We use transitions $(\mathbf{x}_t, \mathbf{x}_{t+1})$ from both trajectory subsets as seeds for the recovery search. For collision trajectories, we retain the transition leading to the first collided state,

$$\mathcal{B}_{\text{col}} = \{(\mathbf{x}_t, \mathbf{x}_{t+1}) : \tau \in \mathcal{T}_{\text{col}},\ \mathbf{x}_{t+1} \in \mathcal{X}_{\text{col}}\},$$

where $\mathcal{X}_{\text{col}}$ denotes the collision set. For regular trajectories, we retain transitions whose successor state lies within a projected ring around the obstacle,

$$\mathcal{B}_{\text{reg}} = \{(\mathbf{x}_t, \mathbf{x}_{t+1}) : \tau \in \mathcal{T}_{\text{reg}},\ d_{\text{unsafe}} \le \|\kappa(\mathbf{x}_{t+1}) - c_{\text{obs}}\|_2 \le d_{\text{unsafe}} + \epsilon\},$$

where $\epsilon > 0$ specifies the ring width. The combined seed pool $\mathcal{B}_{\text{col}} \cup \mathcal{B}_{\text{reg}}$ is then used to search backward for dynamically feasible recovery witnesses.

Algorithm 2 Recovery Search for Projected Anchor Set

Require: Seed transitions $\mathcal{B}_{\text{col}} \cup \mathcal{B}_{\text{reg}}$, trajectories $\mathcal{T}$, dynamics F, backtracking step Δ, score threshold η

$\mathcal{D}_{\text{anchor}} \leftarrow \emptyset$
for each seed $(\tau, t_s) \in \mathcal{B}_{\text{col}} \cup \mathcal{B}_{\text{reg}}$ **do**
  for $k = t_s - \Delta, t_s - 2\Delta, \ldots$ **do**
    $\mathbf{x} \leftarrow \mathbf{x}_k$
    while $\mathbf{x}$ remains feasible **do**
      $\mathbf{u}^\star \leftarrow \arg\max_{\mathbf{u}\in\mathcal{U}} \rho(\mathbf{x}, \mathbf{u}), \quad \mathbf{x}^+ \leftarrow F(\mathbf{x}, \mathbf{u}^\star)$
      if $\rho(\mathbf{x}, \mathbf{u}^\star) \geq \eta$ **then**
        $\mathbf{u}^{+\star} \leftarrow \arg\max_{\mathbf{u}\in\mathcal{U}} \rho(\mathbf{x}^+, \mathbf{u}), \quad \mathbf{x}^{++} \leftarrow F(\mathbf{x}^+, \mathbf{u}^{+\star})$
        $\mathcal{D}_{\text{anchor}} \leftarrow \mathcal{D}_{\text{anchor}} \cup \{(\mathbf{x}, \mathbf{u}^\star, \mathbf{x}^+, \kappa(\mathbf{x}^{++}) - \kappa(\mathbf{x}^+)\}$
        break
      end if
      $\mathbf{x} \leftarrow \mathbf{x}^+$
    end while
    if a recovery witness was found **then**
      break
    end if
  end for
end for
return $\mathcal{D}_{\text{anchor}}$

Here, $\rho(\cdot)$ is a measurement of the cosine similarity score

$$\rho(\mathbf{x}, \mathbf{u}) = \frac{\kappa(F(\mathbf{x}, \mathbf{u})) - \kappa(\mathbf{x})}{\|\kappa(F(\mathbf{x}, \mathbf{u})) - \kappa(\mathbf{x})\|_2}^\top \frac{\kappa(\mathbf{x}) - c_{\text{obs}}}{\|\kappa(\mathbf{x}) - c_{\text{obs}}\|_2}, \tag{16}$$

and the velocity vector $\kappa(F(\mathbf{x}, \mathbf{u})) - \kappa(\mathbf{x})$ describes a feasible recovery direction given the state $\mathbf{x}$.

Projected anchor thinning. To maintain geometric coverage without allowing the anchor set to grow unbounded, we thin the accepted anchors according to their locations in the projected safety space. For each anchor $A_i \in \mathcal{D}_{\text{anchor}}$ with representative state a_i, define

$$\delta_i = \kappa(a_i) - c_{\text{obs}}, \qquad d_i = \|\delta_i\|_2.$$

We partition the projected directions δ_i/d_i into M geometric regions and retain at most K anchors from each region. Within each region, anchors are ranked first by increasing distance d_i and then by decreasing recovery score σ_i. Thus, anchors closer to the obstacle are preferred, with stronger recovery witnesses used to break ties.

Algorithm 3 Projected Anchor Thinning

Require: Anchor set $\mathcal{D}_{\text{anchor}}$, projection κ, obstacle center c_{obs}, number of regions M, per-region cap K

Partition the projected anchor directions into regions $\{\mathcal{B}_m\}_{m=1}^M$
$\widehat{\mathcal{D}_{\text{anchor}}} \leftarrow \emptyset$
for $m = 1, \ldots, M$ **do**
  $\mathcal{C}_m \leftarrow \{A_i \in \mathcal{D}_{\text{anchor}} : \kappa(a_i) \in \mathcal{B}_m\}$
  Order $\mathcal{C}_m$ by $(d_i, -\sigma_i)$, where $d_i = \|\kappa(a_i) - c_{\text{obs}}\|_2$
  Retain the first $\min\{K, |\mathcal{C}_m|\}$ anchors
  Add the retained anchors to $\widehat{\mathcal{D}_{\text{anchor}}}$
end for
return $\widehat{\mathcal{D}_{\text{anchor}}}$

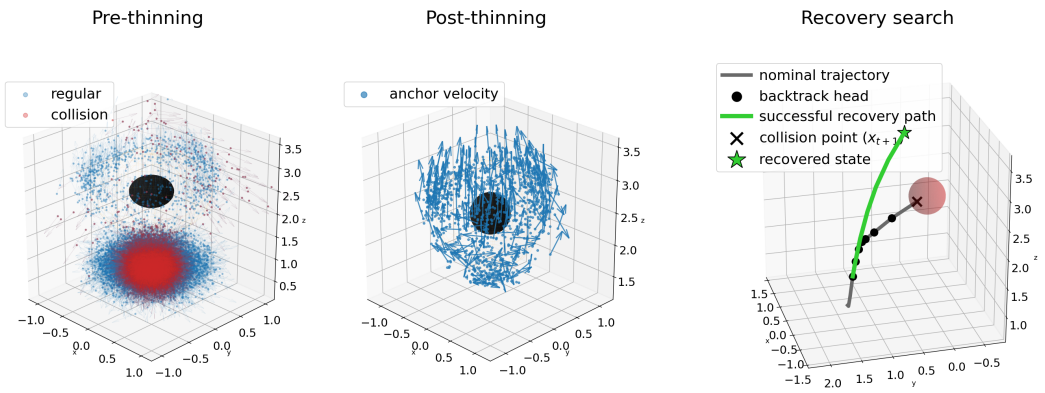


Figure 10: **Example of radial thinning and recovery search**. **Left**: the anchor dataset before thinning. **Mid**: the anchor set coverage and velocities after radial thinning. **Right**: a recovery search from collided trajectory.

Safe and Unsafe Dataset Construction. From the collected trajectories, we classify states according to their projected distance to the obstacle. A state is added to the safe dataset $\mathcal{D}_{\text{safe}}$ if

$$\|\kappa(\mathbf{x}) - c_{\text{obs}}\|_2 \geq d_{\text{safe}},$$

and to the unsafe dataset $\mathcal{D}_{\text{unsafe}}$ if

$$\|\kappa(\mathbf{x}) - c_{\text{obs}}\|_2 < d_{\text{unsafe}}.$$

Neural Training Setup. We train the neural CBF $B_\theta(\mathbf{x})$ using the safe, unsafe, and projected anchor datasets constructed above. The safe and unsafe losses enforce the sign of the CBF:

$$\mathcal{L}_{\text{safe}}(\theta) = \mathbb{E}_{\mathbf{x}_{\text{safe}}}\left[\max\left(-B_\theta(\mathbf{x}_{\text{safe}};\theta) + \epsilon_1, 0\right)^2\right]$$
$$\mathcal{L}_{\text{unsafe}}(\theta) = \mathbb{E}_{\mathbf{x}_{\text{unsafe}}}\left[\max\left(B_\theta(\mathbf{x}_{\text{unsafe}};\theta) + \epsilon_2, 0\right)^2\right]$$

In this case, states in $\mathcal{D}_{\text{safe}}$ are encourated to have positive CBF values, while states in $\mathcal{D}_{\text{unsafe}}$ are encouraged to have negative CBF values. For each projected anchor transition $(\mathbf{x}_t, \mathbf{x}_{t+1}) \in \mathcal{D}_{\text{anchor}}$, we define the outward normal direction from the obstacle as

$$\hat{\mathbf{n}}(\mathbf{x}_{t+1}) = \frac{\kappa(\mathbf{x}_{t+1}) - c_{\text{obs}}}{\|\kappa(\mathbf{x}_{t+1}) - c_{\text{obs}}\|_2},$$

and the transition velocity direction as

$$\mathbf{v}(\mathbf{x}_t, \mathbf{x}_{t+1}) = \frac{\kappa(\mathbf{x}_{t+1}) - \kappa(\mathbf{x}_t)}{\|\kappa(\mathbf{x}_{t+1}) - \kappa(\mathbf{x}_t)\|_2}.$$

Let

$$\mathbf{n}_\theta(\mathbf{x}_{t+1}) = \frac{\nabla_\kappa B_\theta(\mathbf{x}_{t+1})}{\|\nabla_\kappa B_\theta(\mathbf{x}_{t+1})\|_2}$$

denote the normalized CBF normal in the projected space. We anchor our zero level set on $\mathbf{x}_{t+1}$ in $\mathcal{D}_{\text{anchor}}$ and align the normalized ∇B_θ with both the outward obstacle direction $\hat{\mathbf{n}}$ and the recovered transition direction $\mathbf{v}$:

$$\begin{aligned}\mathcal{L}_{\text{anchor}}(\theta) = \mathop{\mathbb{E}}_{(\mathbf{x}_t,\mathbf{x}_{t+1})\sim\mathcal{D}_{\text{anchor}}} [&B_\theta(\mathbf{x}_{t+1})^2 \\ &+ \max\left(-\hat{\mathbf{n}}(\mathbf{x}_{t+1})^\top \mathbf{n}_\theta(\mathbf{x}_{t+1}), 0\right)^2 \\ &+ \max\left(-\mathbf{v}(\mathbf{x}_t, \mathbf{x}_{t+1})^\top \mathbf{n}_\theta(\mathbf{x}_{t+1}), 0\right)^2].\end{aligned}$$

During training, we aim at minimizing the weighted sum of the loss above.

Certification-in-the-loop Retraining. After the initial CBF is trained, we iteratively certify and refine the learned safety boundary. At retraining iteration j, we use the current CBF $B_\theta^{(j)}(\cdot)$ together with the nominal controller u_{nom} to generate a new set of closed-loop trajectories, denoted by $\mathcal{T}_{\text{cert}}^{(j)}$. From these trajectories, we select transition pairs whose successor states lie near the current zero level set:

$$\mathcal{B}_{\text{cert}}^{(j)} = \left\{ (\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) : 0 < B_\theta^{(j)}(\mathbf{x}_{t+1}) \leq \tau_h \right\},$$

where $\tau_h > 0$ specifies the width of the certification band. For each seed transition, we evaluate the cone score $\rho(\mathbf{x}_t, \mathbf{u}_t)$ equation 16. A transition violates the current CBF condition if $\rho(\mathbf{x}_t, \mathbf{u}_t) < \eta$.

For each violated seed, we perform a geometric backtracking procedure Algorithm 2. We keep a pool of such recovery witness, and add them back to our $\mathcal{D}_{\text{anchor}}$. After radial thinning 3, we retrain the CBF using the same training objective. This certify-augment-retrain procedure is neural barrier is good.