CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Closed-loop Real-time Safety Critic as Guardrail in Robotics

Master's Thesis

Master's thesis in Data Science and AI

Xingyu Chen
Shipeng Nan

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Closed-loop Real-time Safety Critic as Guardrail in Robotics

Master's Thesis

Xingyu Chen
Shipeng Nan

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Closed-loop Real-time Safety Critic as Guardrail in Robotics
Master's Thesis
Xingyu Chen and Shipeng Nan

Xingyu Chen, Data Science and AI
Shipeng Nan, Data Science and AI

Supervisor: Ze Zhang, Computer Science and Engineering
Examiner: Ahmed Ali-Eldin Hassan, Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Closed-loop Real-time Safety Critic as Guardrail in Robotics
Master's Thesis
Xingyu Chen and Shipeng Nan
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Robots are increasingly used in manufacturing, logistics, service, and manipulation settings, where physical safety is a basic premise rather than an optional supplement. While robot policies may be obtained through reinforcement learning, imitation learning, optimal control, etc., the development normally involves training, simulation, testing, or rollout data. Apart from learning task behavior, the interactive data can also be used to extract knowledge about danger. To address this issue, this thesis adapts the idea of safe reinforcement learning to a deployment setting in which the task policy is already available and should remain frozen. Instead of retraining the nominal policy with a new constrained objective, the work asks whether a learned safety module can act as a last-step guardrail: before each action is executed, it evaluates the nominal action, accepts it when risk is low, applies a small correction when risk is moderate, or switches to a recovery action when collision risk is high.

Specifically, the proposed method is a closed-loop robot safety guardrail built around a real-time dual-head safety critic. Mixed-behavior trajectories are first used to construct hard-collision labels and continuous soft-risk labels; the critic is then pre-trained to estimate hard-constraint violation risk and cumulative soft-risk cost. During online execution, directional gating, candidate triggering, random shooting, gradient-projection soft correction, and emergency Recovery Actor takeover are combined to produce hierarchical and minimally invasive action correction.

The framework is evaluated on multiple task suites, including a PointMaze navigation problem, a Franka Panda reach-obstacle task, and a Meta-World pick-place-wall manipulation, where the proposed guardrail consistently reduces collision risk while preserving task performance. To start with, collision rates prominently drop in all scenarios. For Panda reach-obstacle, the average minimum clearance also improves. On Meta-World, while the collision instances dramatically decrease, the overall task success also rises. Additional ablation results show that soft correction and Recovery address different risk stages and work best when combined.

Keywords: safety in robotics, safe reinforcement learning, collision avoidance.

Acknowledgements

Xingyu Chen

I would like to express my sincere gratitude to my supervisor for the valuable guidance and suggestions throughout this thesis, especially regarding the experimental design and analysis. This guidance helped me gradually move from having only a limited understanding of reinforcement learning to becoming more confident in working with this topic.

I am also deeply grateful to my friends, who accompanied me during my time in Gothenburg and made this period warmer, more meaningful, and more memorable. I would also like to thank my badminton teammates for the encouragement, energy, and many enjoyable moments outside thesis work, which helped me maintain balance during this journey.

Finally, I would like to express my heartfelt thanks to my family for their unconditional support, patience, and encouragement. Their understanding and trust have always been a source of strength for me, especially during the challenging moments of this thesis.

Xingyu Chen, Gothenburg, 2026-06-21

Shipeng Nan

I would like to thank my supervisor, whose insight and feedback were central to shaping this work and kept it moving in the right direction. I am also thankful to my examiner for taking the time to review this thesis and for the comments that helped sharpen it.

To my friends—thank you for the company, the conversations, and the small everyday moments that made my time in Gothenburg something I will look back on fondly. I am grateful, too, to my family, whose steady support from afar gave me the footing to see this through.

Finally, I extend my gratitude to everyone who has helped and cared for me along the way. Our paths crossed for a reason, and I hope we meet again. Life is not always easy; I hope we all find the courage to do what we truly want, and to stay happy, always.

Shipeng Nan, Gothenburg, 2026-06-21

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background and Motivation

With the increasing trend of robotics systems operating outside safety fences and scripted policies, adaptive control and learning-based policies, which rely on real-time perception and decision-making, pose a significant challenge to safety guarantees. Due to flexible environmental layouts, unexpected scene objects, and even human intervention, contact with the environment is not harmless: warehouse robots move near shelves and people, robot arms pass close to fixtures, and service robots operate around unstructured obstacles [1]. High-level planning interfaces, including large-language-model-based planners, add another source of task commands, although this thesis does not study the planner itself. The low-level safety question remains the same: before an action is executed, can an independent module detect that it is likely to cause contact and replace it with a safer one?

Robotic policies trained under reinforcement learning (RL) typically optimize for cumulative task reward. Such policies often achieve high task success in simulation, but they may output unsafe actions when deployed in the real world or in perturbed environments, due to limited out-of-distribution generalization and simulation-to-real environment mismatch [2], [3], [4]. These unqualified actions may trigger physical collision constraints, damage mechanical components or sensors, and pose a risk to human safety. From a cybersecurity perspective, robotic systems also face adversarial threats: compromised controllers, manipulated sensor readings, malicious command streams, and communication-level denial-of-service attacks can all affect physical behavior and operator safety [5], [6]. Deep reinforcement learning agents can also be compromised by training-time trojans or data poisoning [7]. In such scenarios, the last-step assessment is especially valuable: even if the nominal policy or the higher-level instruction system has been compromised, a low-level safety module can still identify high-risk actions to warn bystanders or correct them before execution.

1.2 Problem Statement

This thesis studies the following deployment setting. A trained nominal policy π_{nom} is given, primarily optimized for task success and frozen during guardrail training and evaluation. At each decision step k, the guardrail module observes the current state $\boldsymbol{s}_k$ and nominal action $\boldsymbol{a}_{\text{nom},k}$, and must decide before execution whether to keep, correct, or take over that action. The goal is to reduce episode-level collision rate, hard-constraint frame ratio(the fraction of per-step decisions within an episode in which the hard constraint is violated), and average collision count, and to improve collision-free success (CF-Success), while preserving task performance as much as possible.

The setting is difficult for several reasons. The safety module must work outside the nominal policy, meaning it cannot rely on retraining or policy gradients. It must also distinguish actual collision from earlier pre-collision states, because these two cases call for different interventions. Offline data are needed to expose the critic to dangerous states before online training, and the final decision rule must remain light enough for real-time use. The central research question is therefore: *how can a real-time, lightweight safety guardrail be designed outside a frozen nominal policy, so that it reduces collision risk, anticipates pre-collision risk, while preserving task performance?*

1.3 Proposed Approach

The proposed guardrail RL framework follows a three-stage offline-to-online pipeline.

Phase 1: Offline safety-critic pre-training. A mixed behavior policy is used to collect offline data. Each transition is annotated with a binary hard-constraint label $c_{\text{hard}} \in \{0, 1\}$ and a non-negative continuous soft-constraint label $c_{\text{soft}} \in \mathbb{R}_{\geq 0}$. A dual-head safety critic is trained with a variant of implicit Q-learning [8], estimating hard-constraint violation risk and cumulative soft-constraint cost separately.

Phase 2: Nominal task-policy training. Since a nominal policy is still required, it is separately trained or loaded depending on the specific data and environment interface of each task. PointMaze and Meta-World use behavior cloning from expert data, while Panda reach-obstacle uses online Soft Actor-Critic with Hindsight Experience Replay [9], [10]. The nominal policy is trained for task success and then kept frozen throughout guardrail training and evaluation.

Phase 3: Online guardrail learning. The pre-trained safety critic and frozen nominal policy are loaded into the online environment. At each control step, the nominal policy first proposes an action. The guardrail then estimates the action's hard-collision risk and soft pre-collision cost. Low-risk actions are executed unchanged. Moderate-risk actions are locally corrected by sampling and gradient projection, and the corrected action is accepted only if it reduces safety risk while remaining close to the nominal action. If the critic detects severe hard risk, the Recovery Actor temporarily takes over. The safety critic and Recovery Actor continue to update on online trajectories to adapt to the executed distribution.

1.4 Contributions and Results Overview

The main contributions of this thesis are:

1. **Formalizing last-step guardrails under a frozen nominal policy.** The safety module is formulated as an action-level post-processor independent of the nominal policy, explicitly separating task-policy learning from deployment-time safety intervention.

2. **Dual-layer safety constraints and a dual-head safety critic.** Actual collisions are modeled as hard constraints, while continuous distance, approach velocity, and discounted future risk form soft constraints. A dual-head critic separately learns hard violation probability and cumulative soft cost.

3. **An offline-to-online guardrail training pipeline.** Offline data are used to pre-train the safety critic, while the online phase updates the critic and Recovery Actor under the executed guardrail policy.

4. **A hierarchical intervention mechanism.** Directional gating, candidate triggering, random shooting, gradient projection, and sparse emergency Recovery are combined so that moderate-risk states receive small corrections, while extremely high-risk states trigger takeover.

5. **Cross-task empirical evaluation.** The experiments cover PointMaze, Panda reach-obstacle, and Meta-World pick-place-wall, spanning 2D navigation, 3D robot reaching with obstacles, and object manipulation.

In the PointMaze Medium / Large baseline comparison, the proposed method reduces collision rate from 0.587 / 0.653 to 0.290 / 0.413, while task success remains at 0.943 / 0.813. In the Panda reach-obstacle paired evaluation, the guardrail reduces collision

rate from 0.70 to 0.00, improves average minimum clearance from 3.9 mm to 13.2 mm, and keeps task success at 1.00. In the Meta-World pick-place-wall $K = 150$ paired protocol, the hard-frame ratio drops from 0.268 to 0.005, task success rises from 0.900 to 0.953, and the number of collision-free instances increases from 9/150 to 130/150. These numbers are the main empirical evidence for using a last-step guardrail outside a frozen nominal policy.

1.5 Thesis Organization

Chapter 2 reviews related work on safe and offline safe reinforcement learning, runtime safety filters and classical collision avoidance, and recovery policies and deployment-oriented guardrails.

Chapter 3 presents the preliminaries and formalizes the problem: it covers reinforcement learning and actor-critic methods, constrained and offline reinforcement learning, and then gives the precise definitions of the dual-layer safety constraints together with the dual-criterion safe RL objective.

Chapter 4 details the general algorithmic design of the guardrail RL framework, with each part specifying its concrete instantiation in PointMaze, Panda reach-obstacle, and Meta-World pick-place-wall.

Chapter 5 reports the experimental results, including maze navigation, Panda robot reach-obstacle, robot-arm manipulation, generalization tests, ablation studies, and cross-environment discussion.

Chapter 6 concludes the thesis, discusses its limitations, and outlines directions for future research.

2

Related Work

2.1 Related Work

2.1.1 Safe and Offline Safe Reinforcement Learning

Safe reinforcement learning is commonly defined as learning policies that maximize task return while satisfying safety or constraint requirements [1], [11]. A central formal route is the constrained Markov decision process, where safety is represented as a cost signal and optimized jointly with reward. CPO uses trust-region updates to approximately enforce such constraints during policy learning [12]. More recent infrastructure and benchmark work, such as Safety-Gymnasium and OmniSafe, have made these constrained reinforcement learning settings easier to compare across tasks and algorithms [13], [14]. These methods provide clear training objectives, but they usually assume that the task policy itself can be retrained or jointly optimized. This assumption is different from the deployment setting in this thesis, where the nominal policy is frozen, and the safety module must act as an external action-level post-processor.

Offline safe reinforcement learning is closely related because robot collisions are expensive or damaging to collect online. Constraints Penalized Q-Learning penalizes both constraint risk and distribution shift in offline Q-learning backups [15]. COptiDICE addresses offline constrained reinforcement learning through stationary distribution correction [16], while Constrained Decision Transformer formulates offline safe reinforcement learning as sequence modeling conditioned on a constraint budget [17]. More recent constraint-conditioned IQL extends this direction with value-based offline safe-policy learning [18]. These methods use offline data to obtain a new safe policy. In contrast, this thesis uses offline data to pre-train a safety critic that evaluates and corrects actions from an already-trained nominal policy, without replacing that policy.

2.1.2 Runtime Safety Filters and Robot Collision Avoidance

Runtime safety methods intervene at execution time rather than only during training. Shielding synthesizes a runtime monitor from formal specifications and blocks unsafe actions selected by a learning agent [19]; later work has extended shielding to partial observability [20]. Control barrier functions provide another important runtime-safety mechanism by enforcing forward invariance of safe sets, often through a quadratic program that minimally modifies the nominal control input [21]. These approaches are close in spirit to this thesis because they preserve a separation between task control and safety intervention. Their limitation in the present setting is that they often require explicit system models, formal specifications, differentiable constraint functions, or locally valid barrier conditions. The guardrail in this thesis instead learns local hard- and soft-risk critics from data and uses those critics to search for lower-risk actions.

Classical robot collision avoidance provides the geometric intuition behind the soft-risk features used here. Artificial potential fields combine an attractive term that pulls the robot toward the goal with a repulsive term that pushes it away from obstacles [22]. The dynamic window approach (DWA) performs local forward simulation in velocity space and selects commands that balance goal progress, speed, and collision clearance [22]. Recent gradient-field extensions to DWA add obstacle-distance gradient terms to improve anticipation in complex environments [23]. These methods are real-time and interpretable, but they depend on hand-designed local scoring functions. This thesis borrows their use of distance, approach direction, and local clearance, then learns a dual-head safety critic over these features so that corrections can be applied to actions proposed by learned robot policies.

2.1.3 Recovery Policies and Deployment-Oriented Guardrails

Recovery policies maintain a separate controller that takes over near unsafe regions. Recovery reinforcement learning (Recovery RL) trains such a controller and switches control when the estimated risk becomes too high [24]. This line of work is closely related to the Recovery Actor in this thesis. However, the proposed guardrail uses Recovery only as a sparse emergency mechanism: moderate-risk states are first handled through soft correction, while full takeover is reserved for states with high hard-risk estimates and very small obstacle clearance. This layered design aims to reduce collisions without unnecessarily discarding the task direction produced by the nominal policy.

The experiments use established robotic learning benchmarks only as evaluation testbeds. PointMaze follows the fixed-dataset navigation setting of D4RL [25], Meta-World provides standardized manipulation tasks [26], and the Panda reach-obstacle task follows a goal-conditioned control setting commonly associated with HER [10]. These benchmarks cover navigation, reaching, and manipulation; the benchmarks themselves are not contributions of this thesis.

The frozen-policy assumption is motivated by deployment settings where the nominal controller may be a legacy module, a third-party policy, or the downstream executor of a high-level instruction system. Since robot controller and communication vulnerabilities can lead to unsafe physical behavior [5], safety mechanisms that operate at runtime without retraining the nominal policy are practically useful. The proposed guardrail fits this setting by inspecting, correcting, or taking over actions immediately before execution while remaining external to the nominal controller.

2.2 Limitations of Existing Approaches

Existing safe reinforcement learning methods often incorporate constraints directly into policy optimization, for example, through constrained Markov decision processes (CMDPs), constrained policy optimization (CPO), and related benchmarked safe reinforcement learning settings [12], [13]. These methods are suitable when a safe policy can be trained from scratch, but they usually assume that the task policy itself can be retrained or jointly optimized. In practical robotic systems, the nominal policy may be a third-party model, a legacy controller, a scripted expert, a behavior-cloned policy, or the downstream executor of a high-level planning module. The safety module may have no access to the policy's training process or gradients. This thesis, therefore, does not aim to relearn a complete task policy. Instead, it adds an independent safety layer immediately before nominal action execution.

Another class of methods filters or corrects actions at runtime. Shielding, control barrier functions, and safety-filter methods all instantiate the idea of runtime safety filtering [19], [21]. These methods are close in spirit to the present goal, but they often rely on formal specifications, explicit system models, differentiable constraint functions, or linearized safety boundaries. In complex robotic tasks, collision risk can depend on nonlinear geometry, local velocity direction, and historical risk. Constructing a globally valid analytic barrier or hand-written constraint is therefore difficult. This thesis instead uses a learned safety critic: low-dimensional geometric safety features are used to learn a local action-risk surface, and actions are corrected

on that learned surface.

Recovery-policy methods train a separate recovery controller and switch control when the estimated risk exceeds a threshold [24]. This is useful as an emergency defense, but a single threshold switch can be either too late or overly intrusive. If it fires only at very high risk, the system may miss the window for small early corrections. If it fires too frequently, it may sacrifice task direction. Offline safe reinforcement learning has recently explored fixed datasets to reduce unsafe online exploration [15], [16], [17], but most such work directly learns a new constrained policy. This thesis instead uses the offline stage to pre-train a runtime risk estimator, rather than to replace the nominal policy.

3

Preliminaries and Problem Formulation

3.1 Preliminaries

3.1.1 Reinforcement Learning and Actor-Critic Methods

The standard framework for reinforcement learning (RL) is the Markov decision process (MDP) [27], denoted by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the immediate reward function, $\gamma \in [0, 1)$ is the discount factor, and ρ_0 is the initial state distribution. The agent aims to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted cumulative reward:

$$J(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]. \tag{3.1}$$

The action-value function $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$ satisfies the Bellman equation:

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma\, \mathbb{E}_{\boldsymbol{s}' \sim P(\cdot|\boldsymbol{s},\boldsymbol{a}),\, \boldsymbol{a}' \sim \pi(\cdot|\boldsymbol{s}')}[Q^\pi(\boldsymbol{s}', \boldsymbol{a}')]. \tag{3.2}$$

Actor-critic methods parameterize the policy (actor) and the value function (critic) separately, updating them in an alternating fashion [27]. Soft Actor-Critic (SAC) [9] introduces a policy-entropy regularization term into the standard RL objective:

$$J_{\mathrm{SAC}}(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t \Big(r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha\, \mathcal{H}(\pi(\cdot \mid \boldsymbol{s}_t))\Big)\right], \tag{3.3}$$

where $\alpha > 0$ is the temperature parameter. SAC employs twin Q-networks to mitigate the overestimation bias of the value function, and propagates gradients

through the stochastic policy via the reparameterization trick, achieving excellent sample efficiency and training stability on continuous control tasks.

3.1.2 Constrained and Offline Reinforcement Learning

One common formulation of safe reinforcement learning is the constrained Markov decision process (CMDP), used by constrained policy-learning methods such as constrained policy optimization (CPO) [12]. It augments the standard Markov decision process with a cost function $c : \mathcal{S} \times \mathcal{A} \to [0, 1]$, a constraint threshold $d \in \mathbb{R}_+$, and a cost discount factor $\gamma_c \in [0, 1)$. The learning objective is then:

$$\max_{\pi} J(\pi) \quad \text{s.t.} \quad J_C(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma_c^t \, c(\boldsymbol{s}_t, \boldsymbol{a}_t)\right] \leq d. \tag{3.4}$$

Offline reinforcement learning learns a policy solely from a static dataset $\mathcal{D}$, without online interaction. Its central challenge is out-of-distribution (OOD) error: when the learned policy visits state-action pairs not covered by the dataset, value estimates may become unreliable. Common responses include policy constraints and pessimistic or conservative value estimation [28], [29]. Implicit Q-Learning (IQL) [8] avoids explicit queries of OOD actions in the Q-function update. The value function is updated via expectile regression with expectile level $\tau \in (0.5, 1)$:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\mathcal{D}}\Big[L_\tau^2(Q_{\bar{\theta}}(\boldsymbol{s}, \boldsymbol{a}) - V_\psi(\boldsymbol{s}))\Big], \quad L_\tau^2(u) = |\tau - \mathbf{1}[u < 0]| \cdot u^2. \tag{3.5}$$

The Q-function is updated via a Bellman backup in which the next-step Q-value is replaced by V_ψ:

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')\sim\mathcal{D}}\left[\Big(r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \, V_{\bar{\psi}}(\boldsymbol{s}') - Q_\theta(\boldsymbol{s}, \boldsymbol{a})\Big)^2\right]. \tag{3.6}$$

This thesis adapts the IQL framework to a dual-objective variant: the Q-network simultaneously estimates hard-constraint violation risk (using a sigmoid output trained on binary or clipped bootstrapped hard-constraint targets) and the cumulative soft-constraint cost (via a Softplus output and mean-squared error). The two objectives share a common feature representation and are trained jointly to improve parameter efficiency.

3.2 Problem Setting

A *nominal policy* π_{nom}, mapping a state $\boldsymbol{s} \in \mathcal{S}$ to an action $\boldsymbol{a}_{\text{nom}} = \pi_{\text{nom}}(\boldsymbol{s})$ and trained purely to maximize task reward, has no notion of physical safety: nothing in its objective discourages it from grazing an obstacle if that completes the task faster. Executing $\boldsymbol{a}_{\text{nom}}$ near an obstacle can therefore lead to a collision—an unsafe event whose consequences, on a physical system, can be severe and irreversible.

Retraining the policy with an added safety objective is not an option here, because we treat the nominal policy as a **frozen black box**: its parameters and gradients are inaccessible—it may come from a third party or a legacy controller whose internals are unavailable—so it can be neither retrained nor differentiated through. Safety must instead be enforced from *outside* the policy, by a **guardrail**: an external module that inspects the nominal action at each step and overrides it only when needed.

Formally, the system pairs the frozen nominal policy π_{nom} with a per-step guardrail. At each step, the guardrail receives $\boldsymbol{a}_{\text{nom}}$ and returns the executed action $\boldsymbol{a}_{\text{guard}}$, either leaving it unchanged ($\boldsymbol{a}_{\text{guard}} = \boldsymbol{a}_{\text{nom}}$) or replacing it with a corrected action when the state is unsafe. Their composition defines the guarded policy π_{guard}. The defining feature of this setting is that safety intervention is handled entirely by the external guardrail, independent of how the nominal policy was obtained.

3.3 Dual-layer Safety Constraints

Physical safety constraints are hierarchical by nature. Collisions are a red line that must not be crossed, with consequences that are typically irreversible; proximity to a danger region is a risk that should be avoided but can be tolerated, with consequences that amount to "being more likely to collide" rather than "having collided already." Accordingly, this thesis models safety through two distinct classes of constraint: a binary hard constraint and a continuous soft constraint.

3.3.1 Hard Constraint

The hard constraint $c_{\text{hard}} : \mathcal{S} \to \{0, 1\}$ is the binary indicator of a collision, i.e. the agent coming within a small clearance ϵ_{coll} of an obstacle:

$$c_{\text{hard}}(\boldsymbol{s}) = \mathbf{1}[d_{\text{obs}}(\boldsymbol{s}) \leq \epsilon_{\text{coll}}]\,, \tag{3.7}$$

where $d_{\text{obs}}(\boldsymbol{s})$ is the clearance between the agent and the nearest obstacle, and ϵ_{coll} is the collision threshold, whose concrete value varies by environment.

The hard-constraint cost of a trajectory is the total time the agent spends in collision:

$$J_H(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1} c_{\text{hard}}(\boldsymbol{s}_t)\right]. \tag{3.8}$$

In the online setting considered here, a collision typically does not terminate the episode, so that the guardrail can attempt to steer the agent back to safety afterwards, and J_H therefore measures how long, not merely whether, the agent remains in an unsafe configuration. Whether a violation ends the episode is a configuration-dependent choice rather than an intrinsic property of the constraint.

3.3.2 Soft Constraint

The soft constraint $c_{\text{soft}}(\boldsymbol{s}_t) \in \mathbb{R}_{\geq 0}$ is a continuous per-step danger cost. Unlike the binary hard constraint, it is graded, providing a dense signal even before a collision occurs, and it measures danger along two aspects. First, it is **direction-sensitive**: it distinguishes "very close but moving away" (low risk) from "moderately close and moving toward" (high risk), so it depends on the agent's direction of motion, not on distance alone. Second, it is **anticipatory**: it reflects the collision tendency of the steps that follow from the current state, rather than only the danger of the current instant, so that risk is signalled before contact becomes imminent.

3.4 The Optimization Objective

Episodes are finite-horizon, with maximum length T. The task return is reported as an undiscounted episodic sum, matching the finite-horizon episodic setting of the experiments, while the cumulative soft-constraint cost uses a discount factor $\gamma_S \in (0, 1)$ to weight near-future risk more heavily than distant-future risk. Task reward is summarized as an episode-level return that need not be temporally weighted; safety relevance, by contrast, decays with the prediction horizon, which γ_S captures.

3.4.1 Nominal Policy Objective

The nominal policy π_{nom} is trained to maximize the cumulative task reward:

$$J_{\text{task}}(\pi_{\text{nom}}) = \mathbb{E}_{\pi_{\text{nom}}}\left[\sum_{t=0}^{T-1} r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]. \tag{3.9}$$

The guardrail does not modify this objective and treats π_{nom} purely as a black-box action generator.

3.4.2 Guardrail Objective

The guarded policy π_{guard} is chosen by a **dual-criterion** objective whose two criteria stand in **lexicographic** priority, enforced operationally at each intervention decision:

1. **Feasibility criterion (primary).** Minimize the hard-constraint cost,

$$\min_{\pi_{\text{guard}}} J_H(\pi_{\text{guard}}). \tag{3.10}$$

2. **Merit criterion (secondary).** Among policies that satisfy the feasibility criterion, prefer those with low cumulative soft-constraint cost and small deviation from the nominal policy, the latter preserving its task performance. The soft-constraint cost is

$$J_S(\pi_{\text{guard}}) = \mathbb{E}_{\tau\sim\pi_{\text{guard}}}\left[\sum_{t=0}^{T-1} \gamma_S^t \cdot c_{\text{soft}}(\boldsymbol{s}_t)\right]. \tag{3.11}$$

This priority is enforced at the level of each per-step intervention decision rather than as a claim of global policy optimality: no improvement in the merit criterion—however large the reduction in soft-constraint cost or in deviation—is allowed to justify an increase in the hard-constraint cost J_H. The per-step mechanism that realizes this dual-criterion objective—the priority cascade over intervention tiers, the Recovery and soft-correction actions, the feasibility-based acceptance test, and the procedure that solves for each correction—is the subject of Chapter 4.

4

Method

Chapter 3 stated the problem setting, the safety objective, and the motivation for separating hard and soft constraints. The central idea of the method is to keep the task policy frozen and insert a learned safety module immediately before action execution: at each step the module evaluates the nominal action, optionally searches for a lower-risk nearby action, and invokes a Recovery Actor only for emergency takeover. Together, the frozen nominal policy and this module form the guarded policy π_{guard}.

All experiments are conducted in simulation. No real robot hardware is used; therefore, hardware-specific deployment issues such as sensing latency, actuator delay, and contact-force estimation are outside the implementation scope of this thesis. The implementation uses three task families: D4RL-style PointMaze navigation [25], a PyBullet Franka Panda reach-obstacle task, and Meta-World pick-place-wall manipulation [26]. The full environment settings, network sizes, replay-buffer sizes, training lengths, and intervention thresholds are listed in Appendices A–C.

Table 4.1: Method-level implementation summary.

Task	**Nominal policy**	**Safety geometry**
PointMaze (7-D)	Behavior-cloned actor	Analytic wall distance and gradient
Panda reach-obstacle (9-D)	SAC with HER [10]	PyBullet closest-point distance
Meta-World (8-D)	Behavior-cloned actor	Tool-center-point to wall distance

Note. PointMaze includes the Medium and Large maps; HER denotes Hindsight Experience Replay. Detailed parameters are given in the appendices.

4.1 Overview of the Guardrail Framework

The framework consists of three artifacts that are trained or loaded in sequence:

- an offline dataset with hard-collision labels and continuous soft-risk labels;
- a dual-head safety critic estimating hard collision risk $\hat{Q}_H$ and cumulative soft cost $\hat{Q}_S$;
- a frozen nominal policy π_{nom}, followed by an online guardrail controller that performs last-step intervention.

The nominal policy is never updated during guardrail training. The safety critic and Recovery Actor are updated during online interaction, but the task policy remains a black-box action generator. Figure 4.1 summarizes the pipeline.

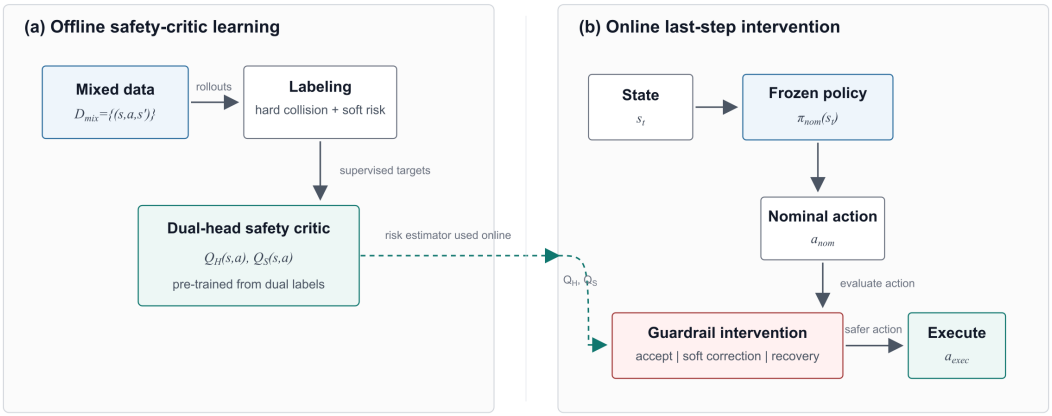

Figure 4.1: Overview of the offline-to-online guardrail pipeline.

4.2 Safety Features and Labels

The safety critic does not use the full task observation. Instead, it uses a compact safety feature vector $\phi(\boldsymbol{s})$ that contains the local geometric quantities most directly related to collision risk:

$$\phi(\boldsymbol{s}) = [\hat{\boldsymbol{g}},\ \boldsymbol{v},\ v_{\text{app}},\ d_{\text{obs}},\ d_{\text{goal}}]\,,$$

which serves as a general template: which of these components a given environment instantiates as critic inputs is environment-specific, and some quantities (such as the nearest-obstacle distance d_{obs}) may instead be used only as a gating variable

rather than as a critic feature. The exact per-environment feature sets are listed in Appendix A. Here $\hat{\boldsymbol{g}}$ is the unit vector pointing away from the nearest obstacle, $\boldsymbol{v}$ is the relevant agent motion vector, d_{obs} is the nearest-obstacle distance, and d_{goal} provides task context. The approach signal is

$$v_{\text{app}} = \boldsymbol{v}^{\top}(-\hat{\boldsymbol{g}}), \tag{4.1}$$

so positive values indicate motion toward the nearest obstacle. The motion vector $\boldsymbol{v}$—and hence v_{app}—is an environment-specific quantity. This feature is central because a state that is close to an obstacle but moving away from it should not be treated in the same way as a state moving into the obstacle.

The hard label is the binary collision indicator. Let ϵ_{coll} denote the environment-specific collision threshold; then

$$c_{\text{hard}}(\boldsymbol{s}_t) = \mathbf{1}[d_{\text{obs}}(\boldsymbol{s}_t) \leq \epsilon_{\text{coll}}]. \tag{4.2}$$

The soft label is constructed offline from two components: an instantaneous cost and an anticipatory cost.

Instantaneous GF-DWA cost. A per-step gradient-field dynamic-window (GF-DWA) potential cost $c_{\text{pot}}(t)$ combines a bounded proximity kernel k in the clearance d_{obs} (active range R) with a gradient-alignment term g that grows when the motion $\boldsymbol{v}$ points toward the obstacle:

$$c_{\text{pot}}(t) = Q_d\, k(d_{\text{obs}}(\boldsymbol{s}_t)) + Q_g\, g(\angle(\boldsymbol{v}_t, \hat{\boldsymbol{g}}_t)) \ \in [0,1]. \tag{4.3}$$

It depends only on the current geometry and motion, and is the per-step soft signal used online. Figure 4.2 illustrates its two terms.

Anticipatory unsafe-region risk. From the same geometry, a binary indicator flags pre-collision states using a warning band wider than the collision threshold, $e(\boldsymbol{s}_t) = \mathbf{1}[d_{\text{obs}}(\boldsymbol{s}_t) \leq \epsilon_{\text{unsafe}}]$ with $\epsilon_{\text{unsafe}} \geq \epsilon_{\text{coll}}$. The anticipatory risk is its discounted fraction over the next H steps, truncated at the episode boundary:

$$c_{\text{risk}}(t) = \frac{1}{Z}\sum_{k=0}^{H-1} \lambda^k\, e(\boldsymbol{s}_{t+k}), \qquad Z = \sum_{k=0}^{H-1} \lambda^k,\ \lambda \in (0,1). \tag{4.4}$$

Figure 4.3 illustrates the unsafe indicator and the discounted look-ahead.

Combined soft label. After quantile normalization $\tilde{c}_{\text{pot}}(t) \in [0,1]$, the two compo-

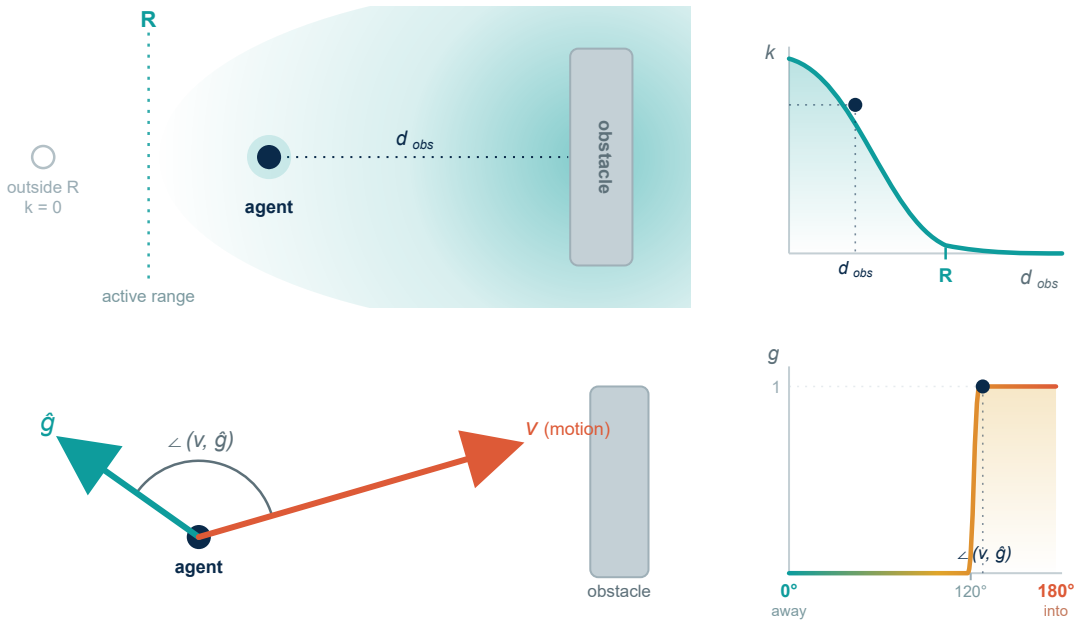

Figure 4.2: The two terms of the instantaneous GF-DWA cost c_{pot} (Eq. (4.3)). **Top:** the proximity term $k(d_{\text{obs}})$ is largest in contact, decreases as the clearance grows, and vanishes beyond the active range R. **Bottom:** the gradient-alignment term $g(\angle(\boldsymbol{v}, \hat{\boldsymbol{g}}))$ stays at zero while the motion is not strongly directed at the obstacle, and switches on only once the approach angle exceeds the threshold (here 120°), saturating to one for near-head-on approach.

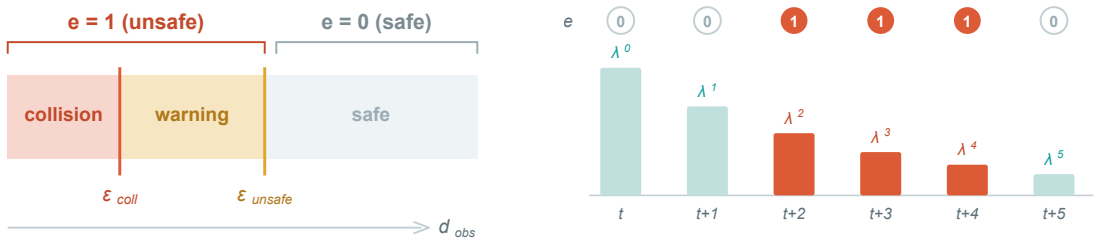

Figure 4.3: Construction of the anticipatory risk c_{risk} (Eq. (4.4)). **Left:** the unsafe indicator $e(\boldsymbol{s}) = \mathbf{1}[d_{\text{obs}} \leq \epsilon_{\text{unsafe}}]$ marks both the collision band ($d_{\text{obs}} \leq \epsilon_{\text{coll}}$) and the wider warning band as unsafe ($e = 1$); only the region beyond ϵ_{unsafe} is safe ($e = 0$). **Right:** c_{risk} looks ahead over the next H steps—each future step contributes its indicator $e(\boldsymbol{s}_{t+k})$ weighted by the discount λ^k (decaying with k), so that only unsafe future steps (highlighted) add to the sum, which is then normalized by Z.

nents are combined with non-negative weights:

$$c_{\text{soft}}(t) = w_{\text{risk}}\, c_{\text{risk}}(t) + w_{\text{pot}}\, \tilde{c}_{\text{pot}}(t). \tag{4.5}$$

The values of $R, \epsilon_{\text{unsafe}}, H, \lambda$, and the weights are environment-specific (Appendix A and B).

4.3 Offline Safety-Critic Pre-training

The offline dataset is collected with mixed behavior policies rather than with the nominal policy alone. This is necessary because a competent nominal policy may avoid many dangerous states, while the safety critic must learn what collision and pre-collision states look like. The mixed data therefore include task-oriented, random, avoidance, and collision-seeking behavior. The dataset records transitions $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$, the safety feature $\phi(\boldsymbol{s}_t)$, hard labels, soft labels, and the auxiliary geometry used to compute those labels.

4.3.1 Dual-head Critic Architecture

The safety critic consists of twin action-value networks and a two-headed state-value network. Each action-value network takes $(\phi(\boldsymbol{s}), \boldsymbol{a})$ as input and outputs one hard-risk logit and one non-negative soft-cost estimate:

$$(q_H^{(i)}, q_S^{(i)}), \qquad i \in \{1, 2\}.$$

At inference time, the two heads are combined as

$$\hat{Q}_H(\boldsymbol{s}, \boldsymbol{a}) = \sigma\Big(\max\Big(q_H^{(1)}, q_H^{(2)}\Big)\Big), \tag{4.6}$$

$$\hat{Q}_S(\boldsymbol{s}, \boldsymbol{a}) = \min\Big(q_S^{(1)},\ q_S^{(2)}\Big). \tag{4.7}$$

The hard head uses the larger of the twin logits to obtain a conservative collision-risk estimate. The soft head uses the smaller estimate to reduce overestimation in the continuous cost value.

The state-value network uses the same safety-feature input $\phi(\boldsymbol{s})$ for both values. It is one MLP with two shared hidden layers and a two-unit output layer, rather than two independent networks. A sigmoid maps the first output to the bounded hard-risk value $V_H(\boldsymbol{s}) \in [0, 1]$, while a softplus maps the second to the non-negative cumulative soft-cost value $V_S(\boldsymbol{s}) \in [0, \infty)$. The two outputs are required because

hard collision risk and soft cost have different targets, ranges, discounts, and expectile losses; sharing the hidden representation avoids duplicating the state encoder. In the implicit Q-learning style adopted here, V_H and V_S provide the respective bootstrap targets for the action-value updates below and are regressed onto the corresponding action-value estimates by expectile regression.

4.3.2 Training Targets

The offline critic is trained using an implicit Q-learning-style value-estimation procedure [8] adapted to risk and cost. Let $d_t \in \{0, 1\}$ denote the terminal flag stored in the dataset transition, and let $\gamma_H, \gamma_S \in (0, 1)$ denote the hard-risk and soft-cost bootstrap discounts, respectively. The hard head uses a one-step target with collision termination semantics:

$$y_H(t) = \text{clip}\left[c_{\text{hard}}(\boldsymbol{s}_t) + (1 - c_{\text{hard}}(\boldsymbol{s}_t))(1 - d_t) \cdot \gamma_H \cdot V_H(\boldsymbol{s}_{t+1}), 0, 1\right]. \tag{4.8}$$

The hard bootstrap discount is used only for critic learning; the hard objective J_H itself (Chapter 3) is undiscounted. When $c_{\text{hard}} = 1$, the target is clamped to one and no further bootstrap is used. This terminal treatment applies to the value backup only: in the online recovery setting the physical episode does not end on a collision (Chapter 3), so the guardrail can still steer the agent back to safety afterwards. Whether a violation also ends the physical episode is a configuration-dependent choice rather than an intrinsic property of the constraint. The soft head regresses on cumulative soft cost:

$$y_S(t) = c_{\text{soft}}(t) + (1 - d_t) \cdot \gamma_S \cdot V_S(\boldsymbol{s}_{t+1}). \tag{4.9}$$

The soft discount may differ from the hard discount because the hard target represents bounded collision risk, whereas the soft target accumulates a dense cost over time.

Let $\beta_S \geq 0$ denote the coefficient that balances the soft-cost loss against the hard-risk loss, whose coefficient is fixed to one. The Q-function loss is

$$L_Q = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')\sim\mathcal{D}}\left[\sum_{i=1}^{2}\left[\text{BCEWithLogits}\left(q_H^{(i)}, y_H\right) + \beta_S \cdot \text{MSE}\left(q_S^{(i)}, y_S\right)\right]\right] \tag{4.10}$$

The hard head is trained as a binary risk classifier, while the soft head is trained as a non-negative cost regressor.

For the value update, $\hat{Q}_{H,\text{tgt}}$ and $\hat{Q}_{S,\text{tgt}}$ denote outputs of the slowly updated target

copy of the twin action-value critic. They use the same hard-head maximum and soft-head minimum as Eqs. (4.6)–(4.7). The task-specific expectile levels $\tau_H, \tau_S \in (0, 1)$ control the asymmetric weighting of positive and negative residuals for the hard and soft outputs, respectively. The state-value outputs are then trained by

$$L_V = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\mathcal{D}}\left[L^2_{\tau_H}\left(\hat{Q}_{H,\mathrm{tgt}}(\boldsymbol{s},\boldsymbol{a}) - V_H(\boldsymbol{s})\right) + \beta_S\, L^2_{\tau_S}\left(\hat{Q}_{S,\mathrm{tgt}}(\boldsymbol{s},\boldsymbol{a}) - V_S(\boldsymbol{s})\right)\right], \quad (4.11)$$

where the expectile loss is

$$L^2_\tau(u) = |\tau - \mathbf{1}[u < 0]| \cdot u^2.$$

Because the critic estimates risk rather than reward, choosing $\tau_H, \tau_S > 0.5$ makes the corresponding value baselines more sensitive to high-risk samples. The numerical discounts, expectile levels, and loss weights are listed in Appendix B.

4.4 Frozen Nominal Policy

The nominal policy supplies the action that would be executed without the guardrail. PointMaze and Meta-World use behavior cloning from expert demonstrations. The PointMaze actor is stored in a SAC-compatible format so that it can be loaded through the same policy-loading interface as the other continuous-control policies, but only the actor is trained by behavior cloning. Panda reach-obstacle uses SAC with HER because sparse goal-conditioned reaching is better served by online relabeling. In all cases, after the nominal checkpoint is loaded by the guardrail trainer, π_{nom} is frozen:

$$\boldsymbol{a}_{\mathrm{nom}} = \pi_{\mathrm{nom}}(\boldsymbol{s}).$$

The deployed action, and the action used in the critic's bootstrap targets, is the deterministic (greedy) one; during online data collection the nominal is sampled stochastically so that the safety critic sees a more diverse state distribution. No gradient from the guardrail critic or Recovery Actor is propagated into the nominal policy.

4.5 Online Guardrail Learning and Intervention

During online training, the guardrail executes the action distribution it is learning to evaluate. The online Bellman target therefore bootstraps from the action that would actually be executed at the next state, not merely from the nominal action.

For a next state $\boldsymbol{s}'$, the nominal policy proposes $\boldsymbol{a}'_{\text{nom}}$, the guardrail decision rule is simulated, and the target critic evaluates the resulting action:

$$\boldsymbol{a}'_{\text{guard}} = \text{Guardrail}(\boldsymbol{s}', \boldsymbol{a}'_{\text{nom}}, \pi_{\text{rec}}, \hat{Q}_H, \hat{Q}_S).$$

The simulated decision rule follows the same three-tier priority used at deployment (Recovery, then soft correction, then the nominal action), and, unless disabled in an ablation, both the hard and soft targets are bootstrapped from this replayed action. This policy-consistent target reduces the mismatch between critic training and deployment-time decisions.

For stability, the implementation distinguishes the current critic, a slowly updated target critic, and a reference critic. The reference critic is used for trigger decisions when needed, so that the intervention rule does not oscillate with every stochastic update of the current network. The exact synchronization schedule is an implementation hyperparameter and is listed in Appendix B.

Figure 4.4 summarizes the complete network architecture before the detailed decision rules are introduced. The current twin critic evaluates both the nominal action and the candidates proposed by the local search; the return arrows carry candidate actions, while the forward arrows carry risk scores and action gradients. The Recovery Actor receives only the safety features and produces a residual that is added to the nominal action. The current critic supplies the soft-intervention trigger, whereas the reference critic supplies the stable Recovery trigger. Dashed arrows show the Bellman/expectile learning relation and the target and reference updates.

4.5.1 Decision Flow

At each time step, the guardrail follows the priority order shown in Figure 4.5. The nominal action is used unless one of the safety checks justifies intervention.

Directional gate. The guardrail first checks whether the action moves toward the nearest obstacle:

$$v_{\text{app}} > v_{\text{thre}}. \tag{4.12}$$

If this condition is false, the action is executed without correction. This prevents unnecessary intervention in states that are close to an obstacle but already moving away from it. The approach signal v_{app} is motion-derived in most environments, but in Meta-World it is computed from the commanded nominal action to avoid a wall-contact velocity deadlock.

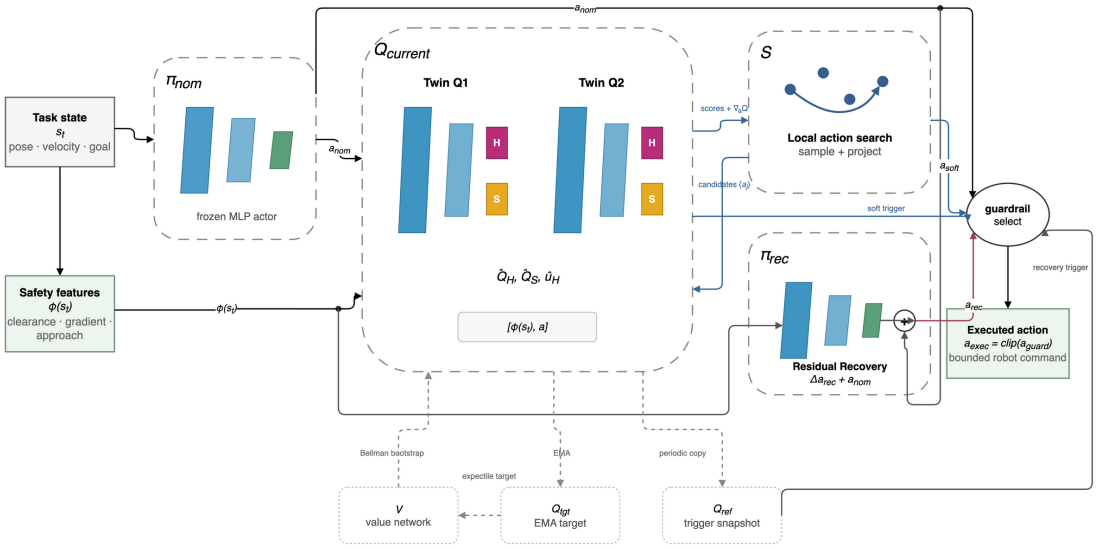

Figure 4.4: Architecture of the guardrail framework. Solid arrows show inference and action paths, blue arrows show critic-guided candidate evaluation, and dashed arrows show value learning and critic synchronization.

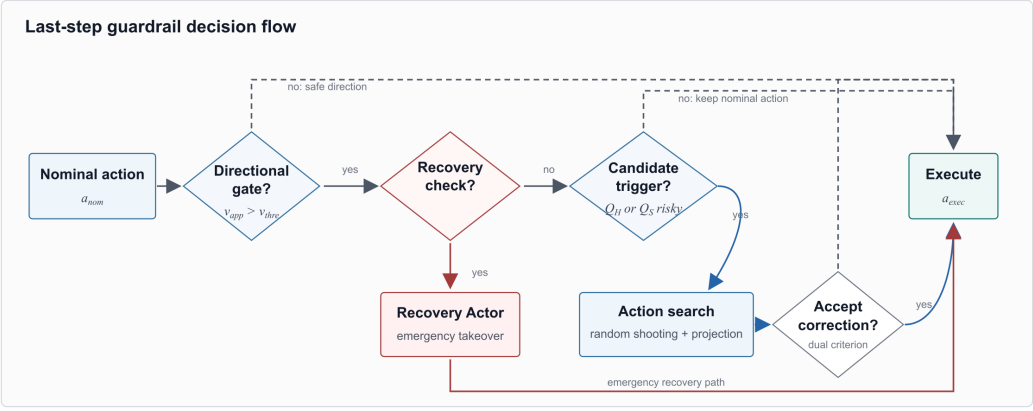

Figure 4.5: Decision flow of the last-step guardrail intervention mechanism.

Emergency Recovery. If the state is extremely close to an obstacle and the hard-risk estimate is high, the Recovery Actor takes over:

$$\hat{Q}_H^{\text{ref}}(\boldsymbol{s}, \boldsymbol{a}_{\text{nom}}) \geq \theta_{\text{rec}} \ \wedge \ d_{\text{obs}} < \delta_{\text{enter}} \ \wedge \ v_{\text{app}} > v_{\text{thre}}. \tag{4.13}$$

This branch is triggered rarely: its role is not to optimize task progress but to provide a last line of defense when soft correction is unlikely to have enough room.

Candidate trigger. If Recovery is not activated, the guardrail decides whether the nominal action should be softly corrected:

$$\text{Trigger} = C_1 \vee C_2 \vee C_3, \tag{4.14}$$

where $\hat{Q}_H$, $\hat{Q}_S$, and $\hat{u}_H$ below are evaluated at $(\boldsymbol{s}, \boldsymbol{a}_{\text{nom}})$:

$$C_1 : \hat{Q}_H \geq \theta_H^{\text{high}}, \tag{4.15}$$

$$C_2 : \hat{Q}_H \geq \theta_H^{\text{low}} \ \wedge \ \hat{Q}_S \geq \theta_S, \tag{4.16}$$

$$C_3 : \hat{u}_H \geq \theta_u, \tag{4.17}$$

where

$$\hat{u}_H(\boldsymbol{s}, \boldsymbol{a}_{\text{nom}}) = |\sigma(q_H^{(1)}(\boldsymbol{s}, \boldsymbol{a}_{\text{nom}})) - \sigma(q_H^{(2)}(\boldsymbol{s}, \boldsymbol{a}_{\text{nom}}))|$$

is a twin-network disagreement score. C_1 handles high hard-risk states, C_2 handles moderate hard risk with strong soft pre-collision warning, and C_3 handles critic uncertainty.

4.5.2 Soft Correction

Soft correction searches within a ball around the nominal action, $\mathcal{B}_\delta(\boldsymbol{a}_{\text{nom}})$, so that the executed action remains close to the task policy. First, a CEM-style sampling step draws candidate actions and scores them by

$$\text{score}(\tilde{\boldsymbol{a}}) = \hat{Q}_H(\boldsymbol{s}, \tilde{\boldsymbol{a}}) + w_S \hat{Q}_S(\boldsymbol{s}, \tilde{\boldsymbol{a}}) + w_d \|\tilde{\boldsymbol{a}} - \boldsymbol{a}_{\text{nom}}\|^2. \tag{4.18}$$

Only candidates that reduce $\hat{Q}_H$ relative to the nominal action are kept. The best candidates are then refined by projected gradient descent:

$$\boldsymbol{a}_{k+1} = \Pi_{\mathcal{B}_\delta(\boldsymbol{a}_{\text{nom}})} \left[\boldsymbol{a}_k - \eta \frac{\nabla_{\boldsymbol{a}_k} \mathcal{L}(\boldsymbol{a}_k)}{\|\nabla_{\boldsymbol{a}_k} \mathcal{L}(\boldsymbol{a}_k)\| + \epsilon} \right], \tag{4.19}$$

with

$$\mathcal{L}(\boldsymbol{a}) = \lambda_H \max(0, \hat{Q}_H(\boldsymbol{s}, \boldsymbol{a}) - \kappa \hat{Q}_H(\boldsymbol{s}, \boldsymbol{a}_{\text{nom}}))^2 + \lambda_S \hat{Q}_S(\boldsymbol{s}, \boldsymbol{a}) + \lambda_a \|\boldsymbol{a} - \boldsymbol{a}_{\text{nom}}\|^2. \quad (4.20)$$

The hinge target asks for a reliable local risk reduction rather than an aggressive move to zero estimated risk. This keeps the correction minimally invasive.

The best corrected action $\boldsymbol{a}_{\text{soft}}$ is accepted only if it improves the hard-risk estimate by at least a feasibility margin:

$$\Delta_F(\boldsymbol{a}_{\text{soft}}) = \hat{Q}_H(\boldsymbol{s}, \boldsymbol{a}_{\text{nom}}) - \hat{Q}_H(\boldsymbol{s}, \boldsymbol{a}_{\text{soft}}) \geq \epsilon_F. \quad (4.21)$$

If this condition fails, the guardrail executes $\boldsymbol{a}_{\text{nom}}$. This acceptance rule is the operational form of the dual-criterion objective from Chapter 3: hard-risk reduction is required, while soft-cost and action deviation guide the search among feasible corrections.

4.6 Training the Recovery Actor

The Recovery Actor outputs a residual correction on top of the nominal action:

$$\boldsymbol{a}_{\text{rec}} = \text{clip}\left(\boldsymbol{a}_{\text{nom}} + \tanh(\mu_\theta(\phi(\boldsymbol{s}))) \cdot \text{scale}, \boldsymbol{a}_{\text{low}}, \boldsymbol{a}_{\text{high}}\right). \quad (4.22)$$

The residual form keeps Recovery close to the nominal control interface and reduces the difficulty of learning absolute emergency actions.

Recovery training uses two types of supervision. The first is critic-based, where b indexes minibatch samples, w_b is a non-negative safety weight, and α_S controls the contribution of the soft-cost critic:

$$\mathcal{L}_{\text{safe}} = \frac{\sum_b w_b \left(\hat{Q}_H(\boldsymbol{s}_b, \boldsymbol{a}_{\text{rec},b}) + \alpha_S \hat{Q}_S(\boldsymbol{s}_b, \boldsymbol{a}_{\text{rec},b})\right)}{\sum_b w_b + \epsilon}. \quad (4.23)$$

The second is geometric: the residual should point away from the nearest obstacle and, when necessary, imitate a simple escape direction. Here w_b^{geo} is the geometric supervision weight, $\Delta\boldsymbol{a}_{b,\text{eff}}$ is the action displacement projected into the safety-relevant

motion space, and $\Delta \boldsymbol{a}_b^{\text{escape}}$ is the reference escape residual:

$$\mathcal{L}_{\text{wall}} = -\frac{\sum_b w_b^{\text{geo}} \langle \Delta \boldsymbol{a}_{b,\text{eff}}, \hat{\boldsymbol{g}}_b \rangle}{\sum_b w_b^{\text{geo}} + \epsilon}, \tag{4.24}$$

$$\mathcal{L}_{\text{imit}} = \frac{\sum_b w_b^{\text{geo}} \| \Delta \boldsymbol{a}_b - \Delta \boldsymbol{a}_b^{\text{escape}} \|^2}{\sum_b w_b^{\text{geo}} + \epsilon}. \tag{4.25}$$

During early online training, geometric losses dominate because the safety critic is still adapting to the executed distribution. Once the critic is more stable, the Recovery Actor is trained with

$$\mathcal{L}_{\text{rec}} = \lambda_{\text{safe}} \mathcal{L}_{\text{safe}} + \lambda_{\text{wall}} \mathcal{L}_{\text{wall}} + \lambda_{\text{imit}} \mathcal{L}_{\text{imit}}. \tag{4.26}$$

Meta-World additionally uses a small z-lift bias in the escape reference when the end effector is close to the wall but the away-from-wall gradient is nearly horizontal. This prevents Recovery from stalling in front of the wall; the exact gate values are listed in Appendix B.

4.7 Replay Buffer and Online Updates

Online trajectories are stored at the episode level and then written to the replay buffer. The hard head uses one-step targets because hard collision has termination semantics. The soft head can use a short n-step return to propagate dense pre-collision risk over a small horizon. To avoid forgetting offline collision examples during online training, each update batch mixes online replay samples with a fixed fraction of offline samples. The exact mixing ratios, update frequencies, and n-step horizons differ by environment and are reported in Appendix B.

Specialized buffers for hard-collision and near-wall samples were implemented as engineering interfaces, but the final experiments rely primarily on the main replay buffer plus offline-data mixing. This keeps the online update rule simple and makes the reported ablations easier to interpret.

5

Results

This chapter reports the quantitative and qualitative results for PointMaze navigation, Panda reach-obstacle, and Meta-World pick-place-wall [25], [26]. Each section presents the primary baseline comparison together with the generalization, ablation, or trajectory evidence needed to interpret the result. Additional diagnostic plots and secondary trajectories are provided in Appendix D.

5.1 Results on Maze Navigation

5.1.1 Baseline Comparison

The Dynamic Window Approach (DWA) provides a geometric baseline, while the remaining methods use the same nominal-policy backbone unless otherwise stated.

Table 5.1: Maze-navigation baseline comparison (mean over 3 seeds, 100 episodes each).

Environment	Method	Task success	Collision rate	Mean collisions	Collision-free success	Mean steps
Medium	Nominal	0.940	0.587	8.633	0.413	255.283
	Recovery RL	0.920	0.407	9.647	0.583	262.240
	DWA	0.937	0.390	**3.897**	0.610	**252.113**
	Ours	**0.943**	**0.290**	3.970	**0.703**	256.700
Large	Nominal	**0.820**	0.653	18.460	0.340	406.463
	Recovery RL	0.770	0.690	28.203	0.303	429.890
	DWA	0.797	0.503	**7.590**	0.470	408.883
	Ours	0.813	**0.413**	10.617	**0.537**	**402.053**

5.1.1.1 Safety and Collision-Free Success

Table 5.1 shows that the proposed method achieves the lowest collision rate in both mazes. Relative to Nominal, collision rate decreases from 0.587 to 0.290 on Medium,

a reduction of 50.6%, and from 0.653 to 0.413 on Large, a reduction of 36.7%. Mean collision count also falls from 8.633 to 3.970 and from 18.460 to 10.617, showing that the guardrail reduces repeated contact as well as the fraction of episodes with any collision.

Collision-free success combines task completion and zero collision in the same episode. The proposed method improves this metric from 0.413 to 0.703 on Medium and from 0.340 to 0.537 on Large, outperforming all three baselines. DWA obtains slightly fewer mean collisions than the proposed method, but its lower collision-free success indicates that low collision counts remain distributed across more episodes.

5.1.1.2 Task Performance and Baseline Behavior

Task success remains close to Nominal: 0.943 versus 0.940 on Medium and 0.813 versus 0.820 on Large. Mean episode length is similarly stable. The safety gain therefore does not arise from stopping early, avoiding the goal, or taking substantially longer paths.

Recovery RL improves safety on Medium but degrades both collision rate and task success on Large. Its collision rate rises to 0.690, above Nominal's 0.653, and its mean collision count reaches 28.203. By contrast, DWA remains safer than Nominal on both mazes but loses ground in the Large topology. These results indicate that a single takeover threshold and a purely reactive geometric rule are both sensitive to corridor complexity, whereas the layered guardrail remains effective in both layouts.

5.1.2 Generalization Experiments

The generalization experiments evaluate an unseen narrow wall and a five-step velocity perturbation using the Large-maze model without fine-tuning.

Table 5.2: Maze generalization results (Large maze, Nominal vs. Guardrail).

Scenario	Method	Task success	Collision rate	Mean collisions	Collision-free success	10-step survival
Narrow Wall	Nominal	0.900	0.700	13.600	0.300	—
	Guardrail	**0.940**	**0.440**	**6.460**	**0.560**	—
Perturbed	Nominal	0.900	0.780	15.540	0.220	0.810
	Guardrail	**0.940**	**0.660**	**8.440**	**0.320**	**0.850**

The guardrail improves every reported metric in both settings. On Narrow Wall, collision rate falls from 0.700 to 0.440 and collision-free success rises from 0.300 to

0.560. Under velocity perturbations, mean collisions decrease from 15.540 to 8.440, and 10-step post-perturbation survival rises from 0.810 to 0.850.

5.1.2.1 Unseen Obstacle Geometry

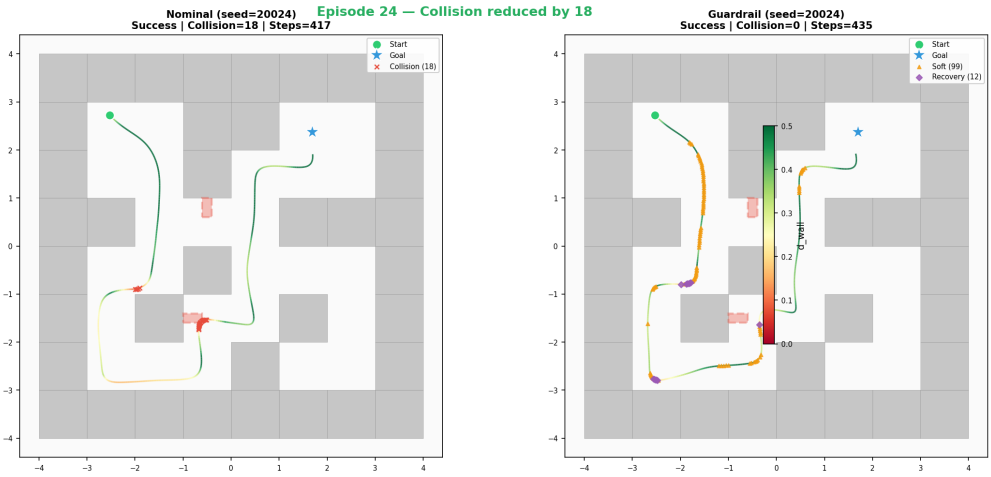

Figure 5.1: Representative Narrow Wall episode. The synthetic wall is absent during training; Guardrail interventions concentrate near the new obstacle and reduce repeated contact.

The Narrow Wall environment is evaluated without fine-tuning. Figure 5.1 shows that interventions concentrate around the added wall rather than at a memorized position from the training layout. The safety features are computed from the current obstacle direction, approach velocity, and clearance, so the same feature interface remains available when the wall geometry changes. The quantitative improvement therefore provides evidence of zero-shot transfer to this tested obstacle modification.

5.1.2.2 Velocity-Perturbation Stress Test

The five-step perturbation directly pushes the agent toward danger and temporarily overrides the selected action. The guardrail cannot prevent motion during the forced interval, but it can respond once control is returned. Figure 5.2 illustrates this recovery behavior, and the reduction in mean collisions and improvement in short-horizon survival indicate that the Recovery layer increases the probability of escaping the danger region after the perturbation.

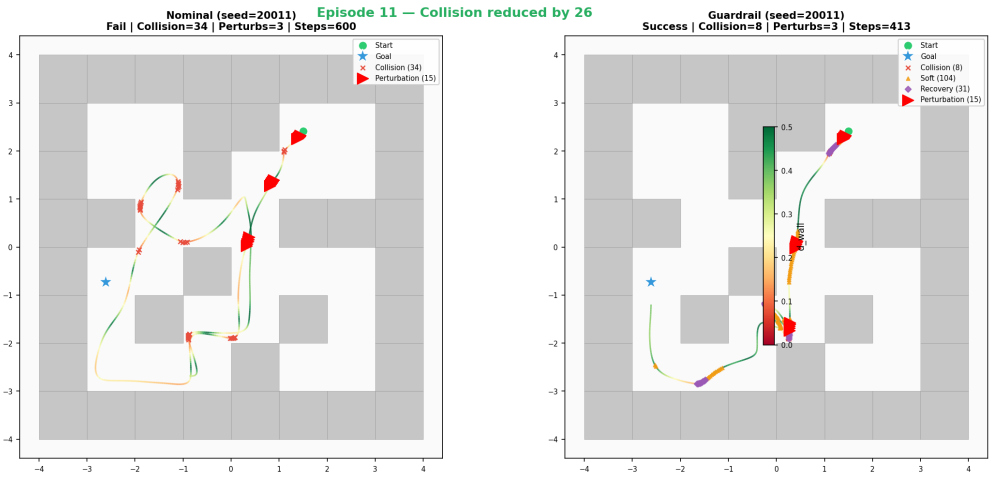

Figure 5.2: Representative velocity-perturbation episode. Red markers indicate the forced perturbation; after it ends, Guardrail retreats from the danger region and resumes progress toward the goal.

Table 5.3: Key maze-navigation ablation metrics (mean over 3 seeds).

Environment	Variant	Task success	Collision rate	Collision-free success
Medium	Nominal	0.933 ± 0.012	0.623 ± 0.068	0.377 ± 0.068
	Soft-only	**0.940 ± 0.022**	0.463 ± 0.062	0.533 ± 0.060
	Recovery-only	0.937 ± 0.012	0.417 ± 0.034	0.577 ± 0.039
	Full	0.937 ± 0.009	**0.307 ± 0.045**	**0.683 ± 0.050**
Large	Nominal	**0.810 ± 0.033**	0.633 ± 0.029	0.363 ± 0.029
	Soft-only	0.807 ± 0.047	0.463 ± 0.017	0.477 ± 0.025
	Recovery-only	0.800 ± 0.045	0.510 ± 0.014	0.443 ± 0.012
	Full	0.807 ± 0.040	**0.403 ± 0.045**	**0.537 ± 0.026**

5.1.3 Ablation Study

5.1.3.1 Medium and Large Maze Comparison

On Medium, Soft-only and Recovery-only reduce collision rate from 0.623 to 0.463 and 0.417, respectively, while Full reduces it further to 0.307. Recovery-only is stronger than Soft-only in this wider-corridor setting, where a close-range retreat can often resolve the remaining danger without creating a topological detour.

On Large, the ordering reverses: Soft-only reaches 0.463, compared with 0.510 for Recovery-only. The more complex corridors create longer moderate-risk approaches before the emergency threshold is reached, leaving more opportunity for preventive correction. Full again performs best at 0.403. Across all variants, task success changes by at most approximately one percentage point, so the safety differences are not explained by reduced task completion.

5.1.3.2 Layer Complementarity

The ablation establishes three useful patterns. First, each intervention layer independently improves safety over Nominal. Second, Full is better than either individual layer in both environments. Third, the stronger individual layer changes with geometry, showing that neither Soft-only nor Recovery-only is sufficient across both layouts. This supports the layered design in which soft correction handles developing risk and Recovery handles the remaining high-risk states.

5.1.4 Representative Maze Trajectory

Figure 5.3 shows a case in which Nominal fails to navigate the Large topology and accumulates 34 collisions over 800 steps. Guardrail completes the same episode in 447 steps with zero collisions. Soft corrections appear before and within narrow passages, while Recovery is used only briefly near the most dangerous wall. This example is consistent with the aggregate and ablation results: preventive corrections handle most of the trajectory, and takeover is reserved for the residual close-range risk. A corresponding Medium-maze example is provided in Figure D.1 in Appendix D.

5.2 Results on Panda Robot Reach-Obstacle

The Panda experiment extends the guardrail from 2D navigation to an A→B robot-reaching task with PyBullet closest-point queries over articulated links. Nominal

Figure 5.3: Representative Large-maze episode. Nominal exhausts the step budget with repeated collisions, whereas Guardrail completes the task with no collision. Soft correction appears around narrow passages and Recovery is limited to the highest-risk location.

and Guardrail are evaluated on the same 100 random seeds with a 2 cm collision threshold.

Table 5.4: Panda reach-obstacle task results (100 episodes).

Metric	**Nominal**	**Guardrail**
Task success	1.00	1.00
Collision rate	0.70	**0.00**
Average minimum clearance	3.9 mm	**13.2 mm**
Average episode length	36.3	40.3 (+11%)

5.2.1 Safety and Clearance

The guardrail eliminates episode-level collisions under this protocol while preserving task success at 1.00. Average minimum clearance increases from 3.9 mm to 13.2 mm, indicating that the intervention does more than react after contact. Figure 5.4 shows the same effect over all sampled steps: Nominal has substantial density near and below the collision threshold, whereas Guardrail shifts the distribution toward larger clearance.

The representative episodes in Figure 5.5 show that the protected trajectories remain at or above the collision threshold when the nominal trajectories enter contact. Intervention is concentrated around the low-clearance portion of each episode rather than throughout the full motion. Together with the distribution shift, these trajectories

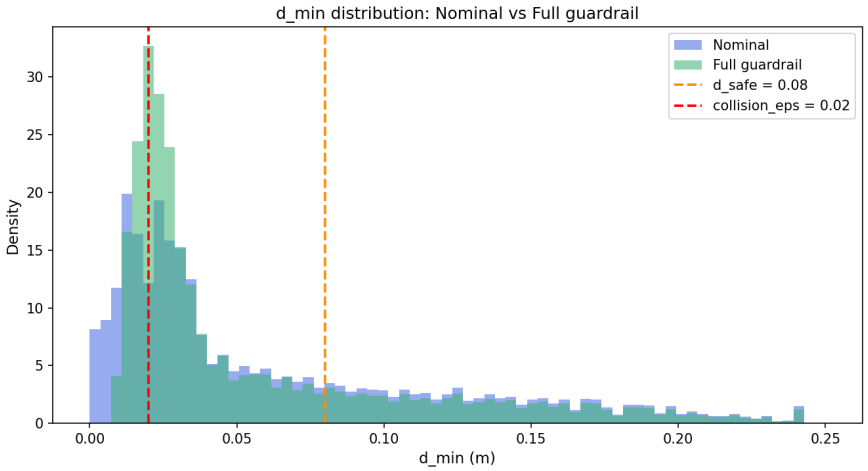

Figure 5.4: Nearest-obstacle distance distribution in Panda reach-obstacle. The dashed lines mark the collision threshold $\epsilon_{\text{coll}} = 0.02$ m and soft-safety threshold $d_{\text{safe}} = 0.08$ m.

support the interpretation that the guardrail changes the approach before sustained contact occurs.

5.2.2 Task Performance and Interface Transfer

Average episode length increases from 36.3 to 40.3 steps, an 11% overhead. This cost is small relative to the reduction from 0.70 to 0.00 collision rate and comes from local detours or slower approach near the obstacle. Since task success remains unchanged, the added steps do not prevent either stage of the A→B task from being completed.

Panda also tests a different geometric interface from PointMaze. The same conceptual safety features—nearest-obstacle direction, approach velocity, and clearance—are retained, but distances are computed through PyBullet closest-point queries over multiple articulated links rather than analytical 2D AABB distances. The result therefore shows that the framework is not restricted to point-mass map geometry under the evaluated setting.

5.3 Results on Robot-arm Manipulation

Meta-World pick-place-wall-v3 evaluates manipulation with a frozen behavior-cloned nominal policy. Training uses task-pool seed 7 and evaluation uses a non-overlapping seed-0 pool. The primary results use 150 paired task instances created through MuJoCo snapshot/restore, so Nominal and Guardrail begin each comparison from the same state. An independent 50-instance official MT1 protocol provides a second

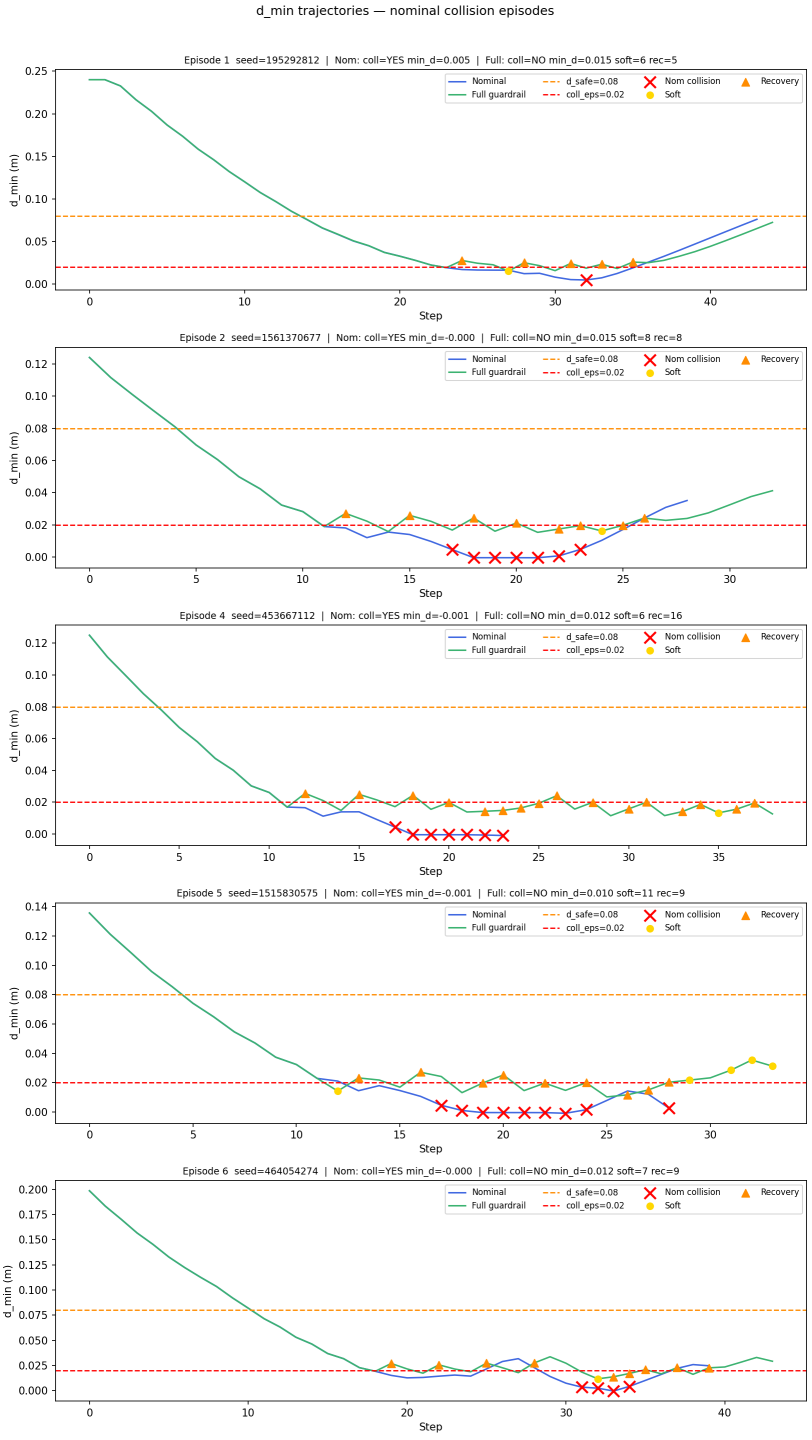

Figure 5.5: Nearest-obstacle distance time series for representative Panda episodes in which Nominal collides. Red crosses mark nominal collision steps; Guardrail soft-correction and Recovery events are shown on the corresponding protected trajectories.

evaluation pipeline. All main-table models use a single training seed; confidence intervals therefore represent evaluation-instance variation rather than training-seed variation.

5.3.1 Baseline Comparison

Table 5.5: Meta-World baseline comparison: 150 paired task instances.

Method	Task success (95% CI)	Hard-frame ratio (95% CI)	Collision rate (episode)
Nominal (behavior cloning)	0.900 [0.85, 0.95]	0.268 [0.23, 0.30]	0.940
Ours (full guardrail)	**0.953 [0.91, 0.99]**	**0.005 [0.00, 0.01]**	**0.133**
Recovery RL (ε=0.5)	0.000	0.003	0.060
Recovery RL (ε=0.8)	0.920	0.199	0.973

Note. Recovery RL confidence intervals are omitted because $\varepsilon = 0.5$ is a degenerate operating point and $\varepsilon = 0.8$ is a post-hoc threshold sweep on the same trained model.

5.3.1.1 Safety and Task Performance

The full guardrail reduces hard-frame ratio from 0.268 to 0.005, a reduction of 98.1%, and episode-level collision rate from 0.940 to 0.133. Collision-free instances increase from 9/150 to 130/150. Both paired bootstrap confidence intervals for the hard-frame-ratio change lie entirely below zero, so the reduction is not driven by a small subset of instances.

Task success rises from 0.900 to 0.953. In the paired contingency table, no instance succeeds under Nominal but fails under Guardrail, whereas eight instances fail under Nominal but succeed under Guardrail. These rescued instances are cases where avoiding or escaping the wall also prevents the policy from exhausting the step budget.

5.3.1.2 Recovery RL Comparison

Recovery RL illustrates the safety–task trade-off of direct takeover in this setting. At its training threshold $\varepsilon = 0.5$, it suppresses collision but achieves zero task success because the absolute recovery action replaces task progress once triggered. At the best post-hoc threshold in the sweep, task success recovers to 0.920 but hard-frame ratio remains 0.199 and episode-level collision rate is 0.973. The full guardrail therefore provides the strongest joint operating point among the evaluated methods.

5.3.1.3 Collision-Free Success and Efficiency

Under the 150-instance protocol, the intersection of task success and zero collision contains 123 instances, giving collision-free success $123/150 = 0.820$, compared with $9/150 = 0.060$ for Nominal. Mean episode length falls from approximately 79 steps for Nominal to 66 for Full. The reduction occurs because fewer episodes stall at the wall and reach the 150-step limit; it should therefore be interpreted as removal of a failure mode rather than more aggressive early termination.

5.3.2 Cross-Protocol Validation

Table 5.6: Comparison of the two Meta-World evaluation protocols.

	50-instance MT1 (official)		**150-instance paired protocol**	
Metric	Nominal	Ours	Nominal	Ours
Task success	0.880	0.940	0.900	0.953
Hard-frame ratio	0.272	0.006	0.268	0.005
Collision-free success	0.060	0.780	0.060	0.820
Change in success	—	+0.060	—	+0.053
Change in hard-frame ratio	—	−0.266	—	−0.262

Table 5.6 shows close agreement between the official MT1 protocol and the custom paired protocol. The differences between protocols are at most 0.013 for task success, hard-frame ratio, and their reported changes; collision-free success differs by four percentage points. This consistency indicates that the main result is not specific to one task-instance sampling or pairing procedure. The complete 50-instance baseline table is retained in Appendix D.

5.3.3 Ablation Study

Table 5.7: Meta-World soft-correction ablation (150-instance paired protocol).

Variant	**Task success**	**Hard-frame ratio**	**Collision-free instances**	**Soft / Recovery triggers per episode**
Nominal (behavior cloning)	0.900	0.268	9	—
Full (soft + Recovery)	**0.953**	**0.005**	**130**	15.8 / 10.8
Hard-only (Recovery only)	0.953	0.033	108	0.0 / 12.6

Full and Hard-only attain the same task success (0.953), but adding soft correction reduces hard-frame ratio from 0.033 to 0.005 and increases collision-free instances

from 108 to 130. The safety improvement therefore occurs without a measurable task-success cost.

Intervention frequencies provide additional evidence about how the layers interact. Full adds 15.8 soft corrections per episode but reduces Recovery activation from 12.6 to 10.8. Earlier low-magnitude corrections therefore change the subsequent trajectory sufficiently to reduce entry into states that require takeover. This result is consistent with the PointMaze ablation, where the full combination is also stronger than either layer alone.

Hard-only remains substantially more usable than vanilla Recovery RL: it preserves task success at 0.953 with hard-frame ratio 0.033, whereas Recovery RL at its training threshold collapses to zero success. The residual Recovery actor and gated trigger used by the proposed framework therefore avoid the complete task abandonment observed with absolute-action threshold switching.

5.3.4 Qualitative Example

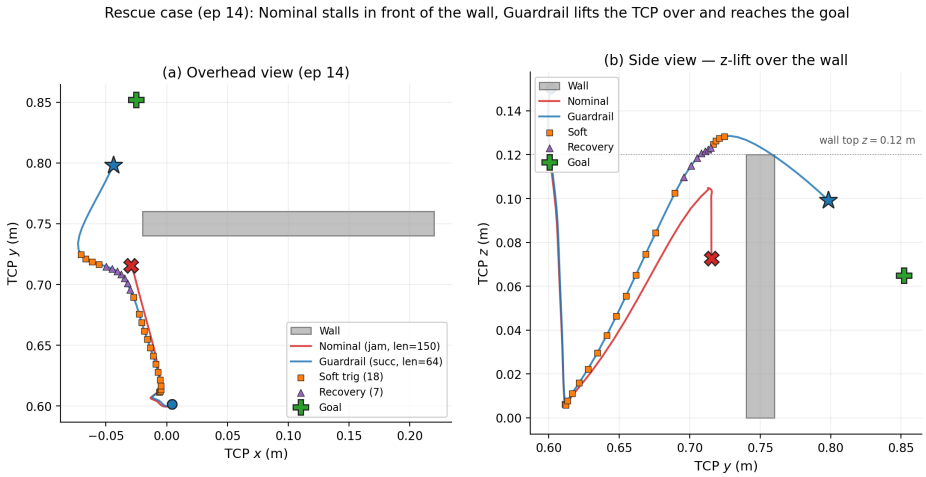

Figure 5.6: Nominal and Guardrail TCP trajectories on rescue episode 14. Nominal stalls at the wall, while Guardrail uses soft correction and a short Recovery interval to lift over the wall and reach the goal.

In rescue episode 14, Nominal remains in contact with the wall for 92 steps and exhausts the 150-step limit. Guardrail completes the task in 64 steps with no collision, using 18 soft corrections and 7 Recovery takeovers. The trajectory in Figure 5.6 shows the TCP rising to approximately 0.13 m and clearing the 0.12 m wall during the carry phase. Nominal reaches only approximately 0.105 m and stalls in front of the obstacle.

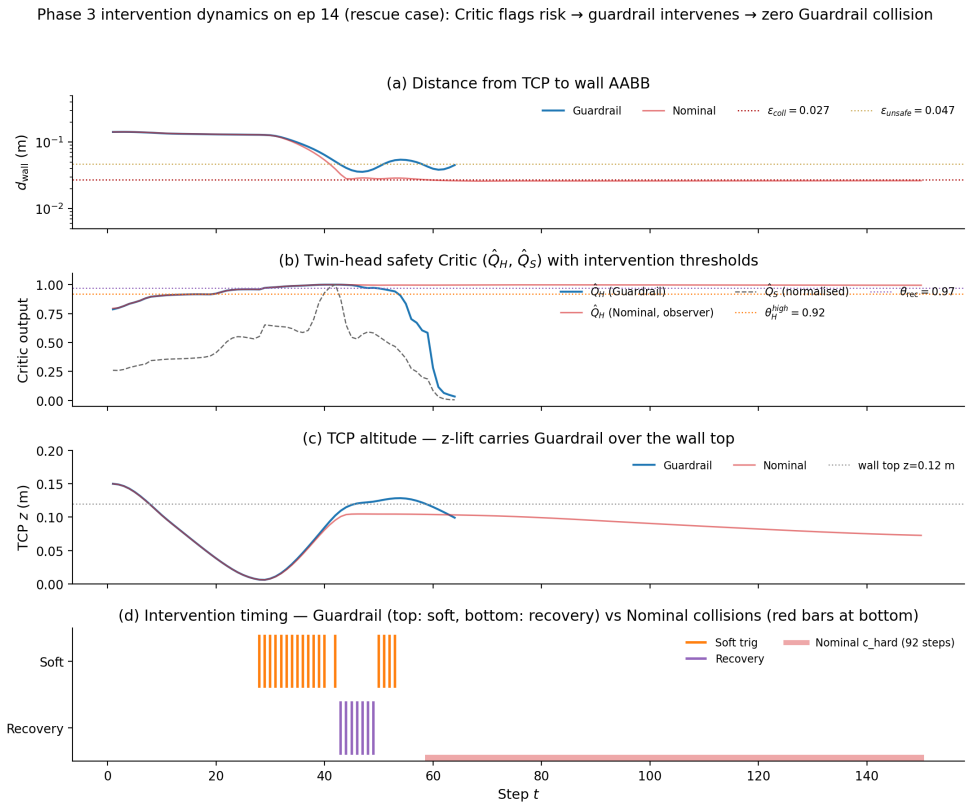

Figure 5.7: Per-step comparison on rescue episode 14: wall distance, dual-head critic outputs, TCP height, and intervention events. The critic risk rises before physical contact; after intervention, the Guardrail trajectory clears the wall without collision.

Figure 5.7 provides the temporal view. On the nominal trajectory, $\hat{Q}_H$ rises well before wall distance reaches the collision threshold and is near saturation before the first hard frame. On the protected trajectory, this warning activates soft correction and then a seven-step Recovery interval during wall crossing. The risk estimate subsequently falls as clearance is restored. This episode illustrates the intended sequence of early warning, local correction, brief takeover, and return to nominal control.

5.4 Summary of Findings

Across all three environments, the guardrail reduces collision-related metrics while maintaining task success. On the common episode-level collision-rate axis, the value decreases from 0.653 to 0.413 on PointMaze Large, from 0.70 to 0.00 on Panda, and from 0.940 to 0.133 on Meta-World. The environment-specific metrics show the same direction: PointMaze mean collisions decrease, Panda minimum clearance increases, and Meta-World hard-frame ratio falls from 0.268 to 0.005.

Task performance does not show a corresponding decline. PointMaze Large changes only from 0.820 to 0.813, Panda remains at 1.00, and Meta-World rises from 0.900 to 0.953. In PointMaze and Meta-World, some task failures are themselves consequences of collision or wall contact, so removing these failure modes can preserve or improve success rather than creating a strict safety–performance trade-off.

The ablations provide consistent evidence for layered intervention. Soft correction and Recovery each improve safety independently in PointMaze, their relative importance changes with maze geometry, and the full configuration performs best. In Meta-World, adding soft correction to Hard-only lowers hard-frame ratio from 0.033 to 0.005 at identical task success. Panda does not include a layer ablation, so this mechanism-level evidence is limited to PointMaze and Meta-World.

Finally, the experiments cover several forms of shift: unseen obstacle geometry, short forced velocity perturbations, a different distance-query interface, held-out manipulation instances, and two independent Meta-World evaluation protocols. These results establish the empirical findings used in Chapter 6, where their interpretation, causal limits, single-seed caveat, and external-validity boundaries are discussed.

6

Conclusions and Beyond

6.1 Discussion

What the results may indicate. The results across the three environments support one interpretation: low-dimensional geometric safety features are closer to sufficient statistics for collision-risk decisions than raw task observations. PointMaze uses AABB distance gradients, Panda uses PyBullet closest-point queries, and Meta-World uses analytical TCP-to-wall AABB distances. Although the sensing interfaces and action spaces differ, all three rely on local quantities such as nearest-obstacle direction, approach velocity, and safety distance. The results therefore suggest that the method learns a local risk structure similar to "near an obstacle, moving toward it, while still needing to make task progress," rather than memorizing absolute positions in a single map or robot model.

Complementarity between soft correction and Recovery. The ablations show that Soft and Recovery are not interchangeable safety switches. Soft correction is preventive: when hard risk has not yet reached the emergency threshold but soft cost indicates that the trajectory is moving toward danger, it changes the local action by a small amount. Recovery is a last defense: at very small distances and high hard risk, it sacrifices some task direction to move away from the obstacle. PointMaze Large shows that Soft-only can outperform Recovery-only in complex topology, while Meta-World shows that even when Recovery is available, the soft layer still lowers the hard-frame ratio from 0.033 to 0.005. These findings support the mechanism-level explanation that layered intervention covers different stages of risk.

Which claims are not causal claims. Although paired evaluation and ablation strengthen the plausibility of the mechanistic interpretation, this thesis should not claim that every local intervention is individually causal. Causal effect requires comparison with a counterfactual outcome: what would have happened in the same

state had the nominal action been executed, and what happens when the guardrail action is executed. The experiments compare complete policies under shared seeds or paired instances, but at a single time step the system observes only the outcome of the action actually executed. Thus, it is appropriate to say that enabling the guardrail is associated with lower collision rate and higher CF-Success, and that ablations support the roles of Soft and Recovery. It is not justified to claim that every trigger causally prevented a collision.

Selection bias induced by intervention. The guardrail is not randomly triggered. It triggers in high-risk states based on $\hat{Q}_H$, $\hat{Q}_S$, approach velocity, and distance thresholds. This creates a problem analogous to confounding by indication in clinical medicine: high-risk patients are more likely to receive treatment, and if the treatment succeeds, observational data may incorrectly suggest that the high-risk state was not dangerous. In this system, high-risk states are more likely to receive soft correction or Recovery. If intervention prevents a collision, subsequent online data may contain samples with high-risk features and no collision outcome. Without distinguishing the state's intrinsic risk from the effect of the intervention, the safety critic may gradually underestimate some dangerous states. This is not merely an implementation issue, but a causal feedback problem shared by adaptive risk-triggered safety systems.

Confounding, data bias, and external validity. The results also depend on experimental-design factors. First, offline data come from mixed behavior policies, and the proportions of active collision, avoidance, expert, and random behavior affect the distribution of dangerous states observed by the critic. Second, online data are collected under the guardrail policy itself, so the training distribution is reshaped by intervention. Third, all environments are simulated; collision thresholds, distance queries, and obstacle geometry are provided by the simulator. Real-world sensing delay, localization error, contact dynamics, and actuator limits may shift the true risk boundary. Fourth, PointMaze, Panda, and Meta-World cover several continuous-control settings, but they do not represent all robot tasks. The external validity of the results should therefore be understood as evidence for transfer potential in tasks with reliable local geometry, not as a universal guarantee.

Single-seed limitation on Meta-World and Panda. The Meta-World main-table results in Section 5.3 (task success, hard-frame ratio, collision-free instance count) are obtained from a single training seed for each of Full, Hard-only, and the Recovery RL baseline. The 95% bootstrap confidence intervals reported in those tables capture eval-instance variance under a fixed trained model, but do not estimate training-randomness variance across re-seeded runs. The +0.028 hard-frame-ratio

gap between Full and Hard-only could therefore in principle lie within seed-to-seed training variance; corroborating multi-seed evidence is left to future work. The Panda results (Section 5.2) are likewise obtained from a single training seed; the same training-randomness caveat applies, although the Panda evaluation does not include an ablation comparison whose marginal difference could fall within seed variance.

When the method may work and what remains to be tested. The method is most likely to be effective when reliable nearest-obstacle distance and away-from-obstacle direction are available; the nominal policy already has basic task competence; dangerous outcomes can be mitigated by local action correction or short-term retreat; and offline data cover enough near-obstacle, pre-collision, and collision states. The maze generalization experiments show that unseen obstacles and short velocity perturbations do not necessarily break the method if the safety features still capture the local risk. Dynamic obstacles, non-convex contact-rich geometry, sensor noise, and high-speed systems should therefore be treated as untested extension boundaries rather than as proven failure modes. They may require more accurate distance estimates, faster control loops, predictive state features, or explicit uncertainty modeling.

6.2 Conclusions

This thesis developed and evaluated an offline-to-online safety guardrail for frozen nominal robot policies. The implemented framework separates task-policy learning from deployment-time safety intervention. It uses dual safety labels and a dual-head critic to distinguish hard collisions from developing soft risk, and combines directional gating, candidate triggering, random-shooting + gradient-projection soft correction, and Recovery Actor takeover to decide whether to keep, correct, or replace each nominal action. The framework was implemented and evaluated on PointMaze, Panda reach-obstacle, and Meta-World pick-place-wall without modifying the frozen nominal policies.

Under the evaluation protocols used in this thesis, the guardrail substantially reduced collision-related metrics while preserving task performance. PointMaze Medium / Large collision rates dropped from 0.587 / 0.653 to 0.290 / 0.413; the Panda reach-obstacle collision rate dropped from 0.70 to 0.00 while task success remained 1.00; and the Meta-World $K = 150$ paired hard-frame ratio dropped from 0.268 to 0.005, with collision-free instances increasing from 9/150 to 130/150. The PointMaze

and Meta-World ablations further showed that soft correction reduced developing risk before the emergency threshold, while Recovery provided a final defense in high-risk states. Together, these results validate the layered intervention design in the evaluated simulation settings and show that, when the nominal policy already has basic task competence and local geometric risk is observable, a learned last-step guardrail can improve safety without changing the nominal policy parameters. This evidence is empirical and specific to the current simulation and evaluation protocols; it is not a formal safety guarantee for arbitrary real-world systems.

6.3 Future Work

Counterfactual safety-effect estimation. The current experiments compare full policies, but they do not estimate the per-trigger effect of replacing a nominal action with a guardrail action. Future work should define each trigger as a treatment, condition on the pre-intervention risk state, and estimate a horizon-level safety effect such as collision reduction or clearance improvement. Recent work on robust causal effect identification, closed-loop marginal structural models, and counterfactual-augmented off-policy evaluation provides suitable tools for this formulation [30], [31], [32].

Extension to dynamic-obstacle scenarios. The current framework assumes static obstacles. Future work could introduce predictive modeling of dynamic obstacles by incorporating obstacle-velocity estimates into the safety features, enabling the critic to anticipate the risks posed by moving obstacles.

Generalization to multi-constraint scenarios. The current framework focuses on a single class of collision constraint. It could be extended to scenarios with multiple coexisting constraint classes—such as joint-angular-velocity limits or workspace-boundary constraints—through a multi-head critic that estimates the violation probabilities of multiple constraint classes simultaneously.

Computational efficiency optimization. The online inference cost of random-shooting + gradient-projection is the main bottleneck for real-time deployment. Future work could explore distilling the soft-correction policy into a lightweight feed-forward network, or using warm-start sampling to accelerate convergence by leveraging the best candidates from previous decisions.

Integration with LLM-driven planning. As large language models see broader use in high-level robotic planning, the guardrail framework could serve as a runtime

safety filter for actions derived from LLM-generated instructions, adding a low-level physical safety check without modifying the LLM itself.

Bibliography

[1] L. Brunke et al., "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[2] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of zero-shot generalisation in deep reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 76, pp. 201–264, 2023.

[3] H. Ju, R. Juan, R. Gomez, K. Nakamura, and G. Li, "Transferring policy of deep reinforcement learning from simulation to reality for robotics," *Nature Machine Intelligence*, vol. 4, pp. 1077–1087, 2022.

[4] G. Dulac-Arnold et al., "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Machine Learning*, vol. 110, pp. 2419–2468, 2021.

[5] J.-P. A. Yaacoub, H. N. Noura, O. Salman, and A. Chehab, "Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommendations," *International Journal of Information Security*, vol. 21, no. 1, pp. 115–158, 2022.

[6] M. Colledanchise, "Address behaviour vulnerabilities in the next generation of autonomous robots," *Nature Machine Intelligence*, vol. 3, pp. 927–928, 2021.

[7] P. Kiourti, K. Wardega, S. Jha, and W. Li, "TrojDRL: Trojan attacks on deep reinforcement learning agents," in *ACM/IEEE Design Automation Conference*, 2020, pp. 1–6.

[8] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations*, 2022.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 1861–1870.

[10] M. Andrychowicz et al., "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[11] W. Zhao, T. He, R. Chen, T. Wei, and C. Liu, "State-wise safe reinforcement learning: A survey," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023, pp. 6814–6822.

[12] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 22–31.

[13] J. Ji et al., "Safety gymnasium: A unified safe reinforcement learning benchmark," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[14] J. Ji et al., "OmniSafe: An infrastructure for accelerating safe reinforcement learning research," *Journal of Machine Learning Research*, vol. 25, no. 285, pp. 1–6, 2024.

[15] H. Xu, X. Zhan, and X. Zhu, "Constraints penalized q-learning for safe offline reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 8753–8760.

[16] J. Lee et al., "COptiDICE: Offline constrained reinforcement learning via stationary distribution correction estimation," in *International Conference on Learning Representations*, 2022.

[17] Z. Liu et al., "Constrained decision transformer for offline safe reinforcement learning," in *Proceedings of the 40th International Conference on Machine Learning*, ser. PMLR, vol. 202, PMLR, 2023, pp. 21 611–21 630.

[18] Z. Liu, X. Li, and J. Zhang, "C2IQL: Constraint-conditioned implicit q-learning for safe offline reinforcement learning," in *Proceedings of the 42nd International Conference on Machine Learning*, ser. PMLR, vol. 267, PMLR, 2025, pp. 38 827–38 841.

[19] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[20] S. Carr, N. Jansen, S. Junges, and U. Topcu, "Safe reinforcement learning via shielding under partial observability," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 14 748–14 756.

[21] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[22] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. MIT Press, 2011.

[23] Z. Zhang, Y. Xue, N. B. Figueroa, and K. Åkesson, "Gradient field-based dynamic window approach for collision avoidance in complex environments," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 19 669–19 674.

[24] B. Thananjeyan et al., "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.

[25] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for deep data-driven reinforcement learning," in *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2020.

[26] T. Yu et al., "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Proceedings of the Conference on Robot Learning*, ser. PMLR, vol. 100, PMLR, 2020, pp. 1094–1100.

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[28] R. F. Prudêncio, M. R. R. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 8, pp. 10 237–10 257, 2024.

[29] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1179–1191.

[30] T. Gorbach, X. de Luna, J. Karvanen, and I. Waernbaum, "Contrasting identifying assumptions of average causal effects: Robustness and semiparametric efficiency," *Journal of Machine Learning Research*, vol. 24, no. 197, pp. 1–65, 2023.

[31] A. W. Levis, G. Loewinger, and F. Pereira, "Causal inference in the closed-loop: Marginal structural models for sequential excursion effects," in *Advances in Neural Information Processing Systems*, vol. 37, 2024.

[32] S. Tang and J. Wiens, "Counterfactual-augmented importance sampling for semi-offline policy evaluation," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

A

Environment Configuration Details

Maze-Navigation Environment Parameters

Table A.1: Environment configuration for the PointMaze maze-navigation experiments.

Item	**PointMaze-Medium**	**PointMaze-Large**
Base environment ID	`PointMaze_MediumDense-v3`	`PointMaze_LargeDense-v3`
Environment type	`medium`	`large`
Safety-feature dimensionality d_ϕ	7	7
Maximum episode length	600	900
Collision threshold ϵ_{coll}	$0.12 \times \ell_{\text{scale}}$	$0.12 \times \ell_{\text{scale}}$
Unsafe threshold ϵ_{unsafe}	$0.12 \times \ell_{\text{scale}}$	$0.12 \times \ell_{\text{scale}}$
Soft-cost activation range R	$0.5 \times \ell_{\text{scale}}$	$0.5 \times \ell_{\text{scale}}$
Reset strategy	Uniformly random	Stratified (near/medium/far ratio 3:4:3)

The safety feature vector used in the PointMaze experiments is

$$\phi(\boldsymbol{s}) = [\hat{g}_x, \hat{g}_y, v_x, v_y, v_{\text{app}}, d_{\text{goal}}, d_{\text{wall,norm}}]^\top \in \mathbb{R}^7.$$

Panda Robot Reach-Obstacle Environment Parameters

The safety feature vector used in the Panda experiments is

$$\phi(\boldsymbol{s}) = [\hat{g}_x, \hat{g}_y, \hat{g}_z, v_x, v_y, v_z, v_{\text{app}}, d_{\text{min,norm}}, d_{\text{goal,norm}}]^\top \in \mathbb{R}^9.$$

Table A.2: Key configuration parameters of the PandaReachObstacle environment.

Parameter	**Value**
Base environment	`PandaReachObstacle` (PyBullet Franka Panda)
Maximum steps	200
Task structure	A→B two-stage reaching
Observation space	15-dimensional flat observation: $\boldsymbol{p}_{\text{ee}}$, $\boldsymbol{v}_{\text{ee}}$, goal, goal-relative, passage-relative
Nominal-policy observation	SB3 dictionary observation: 9-dimensional observation plus achieved/desired goals
Action space	$[-0.01, 0.01]^3$ end-effector displacement increment
Obstacles	Two passage plates placed relative to goal_{A}
Passage gap	0.28 m (training and evaluation configuration)
Plate size	depth 0.12 m, thickness 0.02 m, height 0.40 m
Safety distance d_{safe}	0.08 m
Collision threshold ϵ_{coll}	0.02 m
Success threshold	0.05 m
Distance computation	PyBullet closest-point queries over distal links/gripper links against both obstacle plates

Robot-arm Manipulation Environment Parameters

Table A.3: Key configuration parameters of the Meta-World pick-place-wall environment.

Parameter	**Value**
Base environment ID	`Meta-World/MT1/pick-place-wall-v3`
Meta-World version	3.0.0 (Farama)
Environment step size Δt	0.0125 s
Maximum steps	200 (online guardrail training / offline collection) / 150 (Phase 3 evaluation)
Observation space	39 dimensions (Meta-World native obs, passed through directly)
Action space	$[-1, 1]^4$ (end-effector xyz + gripper)
Obstacle-wall body	`"wall"` (geom unnamed, AABB form)
Obstacle-wall AABB	min $[-0.02, 0.74, 0.00]$, max $[0.22, 0.76, 0.12]$ m
Collision threshold ϵ_{coll}	0.027 m (calibrated)
Unsafe threshold ϵ_{unsafe}	0.047 m (calibrated)
Soft-cost activation range R	0.127 m (calibrated)
Task-pool seed (`make_seed`)	7 for training, 0 for evaluation (zero-overlap asserted)
Reset strategy	MT1 `RandomTaskSelectWrapper` (task switched every episode)

The thresholds ϵ_{coll}, ϵ_{unsafe}, and R are jointly calibrated by a threshold-calibration script. The calibration is based on the distributions of d_{wall} under two reference policies (the wall-expert representing a safe policy and the no-wall expert representing a wall-hugging policy), together with real-MuJoCo statistics of the distance between hand/gripper contact anchors and the wall. The chosen values must clearly separate the c_{hard} frame fractions of the two reference policies while avoiding saturation. The values adopted in this thesis are those shown in the table above: $\epsilon_{\text{coll}} = 0.027\,\text{m}$, $\epsilon_{\text{unsafe}} = 0.047\,\text{m}$, and $R = 0.127\,\text{m}$.

B

Network Architectures and Training Hyperparameters

Safety Critic Network Architecture

Table B.1: Safety critic (TwinQ + VNet) hyperparameters.

Hyperparameter	**Value**
Hidden layer width	256
Number of hidden layers	2 (with ReLU activations)
Twin-Q structure	Two independent MLPs; each MLP shares its hidden layers between hard and soft outputs
State-value structure	One MLP with shared hidden layers and a two-unit output layer
Hard / soft output activation	Sigmoid / softplus
Safety-feature dimensionality d_ϕ	7 (PointMaze) / 9 (Panda) / 8 (Meta-World)
Hard discount γ_H	0.99 (PointMaze) / 0.50 (Panda)
Soft discount γ_S	0.95
Expectile level τ_H	0.8
Expectile level τ_S	0.7 (PointMaze and Panda) / 0.8 (Meta-World)
Offline soft-loss weight β_S	1.0
Target-network update coefficient τ_{ema}	0.005
Collision-sample positive weight w_{H+}	5.0
Reference-network synchronization interval	disabled (maze) / 40,000 steps (robot arm)

Maze Online-Training Hyperparameters

Unless otherwise stated, Medium and Large use the same training and guardrail hyperparameters. Their main differences lie in the maximum episode length and in the pre-trained nominal policy and safety critic used by each.

Table B.2: Training hyperparameters for PointMaze experiments.

Hyperparameter	**PointMaze-Medium**	**PointMaze-Large**
Total online training steps	100,000	100,000
Maximum episode length	600	900
Replay buffer size	500,000	500,000
Batch size	256	256
Update frequency	Every 5 environment steps	Every 5 environment steps
Number of updates per update step	1	1
Update starts after	2,000 steps	2,000 steps
Offline data mixing ratio	20%	20%
Soft-cost return length n	5	5
Hard-cost target	1-step	1-step
Hard discount factor γ_H	0.90	0.90
Soft discount factor γ_S	0.95	0.95
Hard-positive weight w_H^{pos}	20.0	20.0
Safety Critic learning rate	5×10^{-5}	5×10^{-5}
Recovery Actor learning rate	1×10^{-4}	1×10^{-4}
Optimizer	Adam	Adam
Target network update coefficient τ	0.005	0.005
Hidden layer size	256	256
Evaluation interval	Every 10,000 steps	Every 10,000 steps
Evaluation episodes	50	50
Reference network sync interval	999,999,999 steps	999,999,999 steps
Intervention start step	0	0
Seed	0	0

Table B.3: Guardrail and soft-correction hyperparameters for PointMaze experiments.

Hyperparameter	**PointMaze-Medium**	**PointMaze-Large**
Directional gate threshold v_{thre}	0.01	0.01
Candidate threshold θ_H^{high}	0.40	0.40
Candidate threshold θ_H^{low}	0.30	0.30
Soft-risk threshold θ_S	7.0	7.0
Uncertainty threshold θ_u	0.30	0.30
Recovery trigger threshold θ_{rec}	0.50	0.50
Recovery enter threshold δ_{enter}	0.25	0.25
Recovery exit threshold	0.35	0.35
Recovery max lock steps	20	20
random-shooting samples K	64	64
Refined candidates n_{refine}	4	4
Projection steps T_{proj}	8	8
Action trust-region radius δ	0.50	0.50
Projection step size η	0.08	0.08
Hard-risk loss weight λ_H	12.0	12.0
Soft-risk loss weight λ_S	0.10	0.10
Action deviation weight λ_a	1.0	1.0
Soft-risk score weight w_S	0.1	0.1
Action-distance score weight w_d	4.0	4.0
Feasibility threshold ϵ_F	0.0005	0.0005
Merit threshold ϵ_M	0.002	0.002
Merit penalty coefficient β_M	0.1	0.1

Table B.4: Recovery Actor training hyperparameters.

Hyperparameter	**PointMaze-Medium**	**PointMaze-Large**
Recovery Actor learning rate	1×10^{-4}	1×10^{-4}
Optimizer	Adam	Adam
Safe loss coefficient λ_{safe}	0.5	0.5
Wall loss coefficient λ_{wall}	5.0	5.0
Imitation loss coefficient λ_{imit}	3.0	3.0
Distance regularization coefficient λ_{dist}	0.0	0.0
Soft-risk coefficient in safe term α_S	0.2	0.2
Wall-distance threshold	0.5	0.5
Escape action maximum $a_{\max}$	1.0	1.0

Panda Online-Training and Guardrail Hyperparameters

Table B.5: Key training and guardrail hyperparameters for PandaReachObstacle.

Hyperparameter	**Value**
Nominal SAC+HER	
Total training steps	200,000
SAC network structure	$[256, 256]$
HER sampled goals	4, future strategy
Learning rate	3×10^{-4}
Batch size	256
Replay buffer	1,000,000
Discount factor γ	0.95
Fixed actor $\log \sigma$	-1.0
Offline Safety IQL	
Safety feature dimension / action dimension	9 / 3

Continued on next page

Table B.5 (continued)

Hyperparameter	**Value**
Training steps	200,000
Batch size	256
Q/V learning rate	3×10^{-4}
γ_H, γ_S	0.50, 0.95
τ_H, τ_S	0.8, 0.7
w_H, w_S	1.0, 1.0
Target-network update coefficient	0.005
Online Guardrail	
Total training steps	200,000
Main replay buffer capacity	300,000
Offline-data mixing ratio	20%
n-step soft return	3
Updates start after	2,000 steps
Intervention starts after	3,000 steps
Safety critic learning rate	5×10^{-5}
Recovery Actor learning rate	10^{-4}
Trigger and Soft Correction	
Candidate trigger threshold θ_H^{high}	0.50
Candidate trigger threshold θ_H^{low}	0.01
Soft-constraint trigger threshold θ_S	7.0
Uncertainty trigger threshold θ_u	0.30
Recovery trigger threshold θ_{rec}	0.80
Recovery enter distance δ_{enter}	0.02 m
Directional-gate threshold v_{thre}	0.005
random-shooting samples K	96

Continued on next page

Table B.5 (continued)

Hyperparameter	**Value**
Refined candidates	4
Gradient-projection steps T_{proj}	8
Action-ball radius δ	0.008
Step size η	0.0008
Loss weight λ_H	12.0
Loss weight λ_S	0.10
Action-deviation weight λ_a	1.0
Feasibility threshold ϵ_F	5×10^{-4}

Robot-arm Online-Training Hyperparameters

The Phase 3 hyperparameters for the robot-arm manipulation task are listed in Table B.6. The main differences from the maze-navigation configuration above (Tables B.2, B.3, and B.4) stem from the environment's collision density, the action dimensionality (which includes the gripper), and the need to align with the 200k-step training protocol of the Recovery RL baseline. In addition, the robot-arm-specific geometric gating parameters (a_{lift}, d_{thr}, z_{weak}) are used to avoid the `rec_stall` deadlock that arises when the TCP is stuck within the wall height range and $\hat{g}_z \approx 0$ (see Section 5.3 and the related discussion in Chapter 4).

Table B.6: Key hyperparameters for Phase 3 online training (robot-arm manipulation, Meta-World pick-place-wall).

Hyperparameter	**Value**
Training scale and data	
Total training steps	200,000
Replay buffer capacity	300,000

Continued on next page

Table B.6 (continued)

Hyperparameter	**Value**
Offline data mixing ratio	25%
n-step return length	5
Update interval `update_every`	2
Warm-up steps `update_after`	2,000
Intervention warm-up `intervention_start`	20,000
Recovery Actor learning rate	10^{-4}
Trigger thresholds	
Candidate trigger threshold θ_H^{high}	0.92
Candidate trigger threshold θ_H^{low}	0.80
Soft-constraint trigger threshold θ_S	5.0
Uncertainty trigger threshold θ_u	0.30
Recovery trigger threshold θ_{rec}	0.97
Recovery enter distance δ_{enter}	0.05 m
Directional-gate threshold v_{thre}	0
random-shooting + gradient-projection soft correction	
random-shooting samples K	64
Gradient-projection steps T_{proj}	8
Action-ball radius δ	0.5
Step size η	0.05
Loss weight λ_H	12.0
Loss weight λ_S	0.1
Action-deviation weight λ_a	1.0
Improvement ratio κ	0.9
Score weight w_S	0.1
Score weight w_d	4.0

Continued on next page

Table B.6 (continued)

Hyperparameter	**Value**
Feasibility threshold ϵ_F	5×10^{-4}
Recovery Actor loss and geometric gating	
Q_S weight α_S in $\mathcal{L}_{\text{safe}}$	0.2
λ_{safe}	0.5
λ_{wall}	5.0
λ_{imit}	3.0
Escape-action amplitude a_{lift}	0.5
Near-wall gating distance d_{thr}	0.10 m
z-signal weak threshold z_{weak}	0.2

C

Baseline Hyperparameters

DWA Baseline Hyperparameters

Table C.1: DWA baseline hyperparameters.

Hyperparameter	**Value**
Number of candidate actions N	64
Safety margin	0.15
Activation range	0.5
Sampling radius δ	0.5
Forward simulation timestep Δt	0.02
Goal-heading weight w_{goal}	1.0
Clearance weight $w_{\text{clearance}}$	2.0
Nominal-action penalty weight w_{nominal}	0.5

The DWA baseline samples candidate actions in the neighborhood of the nominal action and selects the final action through a weighted score based on the goal direction, the safety distance to walls, and the deviation from the nominal action.

RecoveryRL Baseline Hyperparameters

Table C.2: RecoveryRL baseline hyperparameters.

Hyperparameter	**PointMaze-Medium**	**PointMaze-Large**
Environment type	`medium`	`large`
Environment ID	`PointMaze_MediumDense-v3`	`PointMaze_LargeDense-v3`
Total training steps	100,000	100,000
Pretraining steps	10,000	10,000
Update starts after	2,000 steps	2,000 steps
Update frequency	Every 5 environment steps	Every 5 environment steps
Batch size	256	256
Replay buffer size	500,000	500,000
Hidden layer size	256	256
Risk Q learning rate	3×10^{-4}	3×10^{-4}
Recovery policy learning rate	3×10^{-4}	3×10^{-4}
Target update coefficient τ	0.005	0.005
Risk discount factor γ_{risk}	0.90	0.90
Risk threshold ϵ_{risk}	0.5	0.5
Evaluation interval	Every 10,000 steps	Every 10,000 steps
Evaluation episodes	50	50
Maximum episode length	600	900
Seed	0	0
Device	`cuda`	`cuda`
Optimizer	Adam	Adam

D

Additional Evaluation Results

This appendix collects secondary evaluation protocols and diagnostic visualizations omitted from the main Results chapter for concision.

Maze Representative Episodes

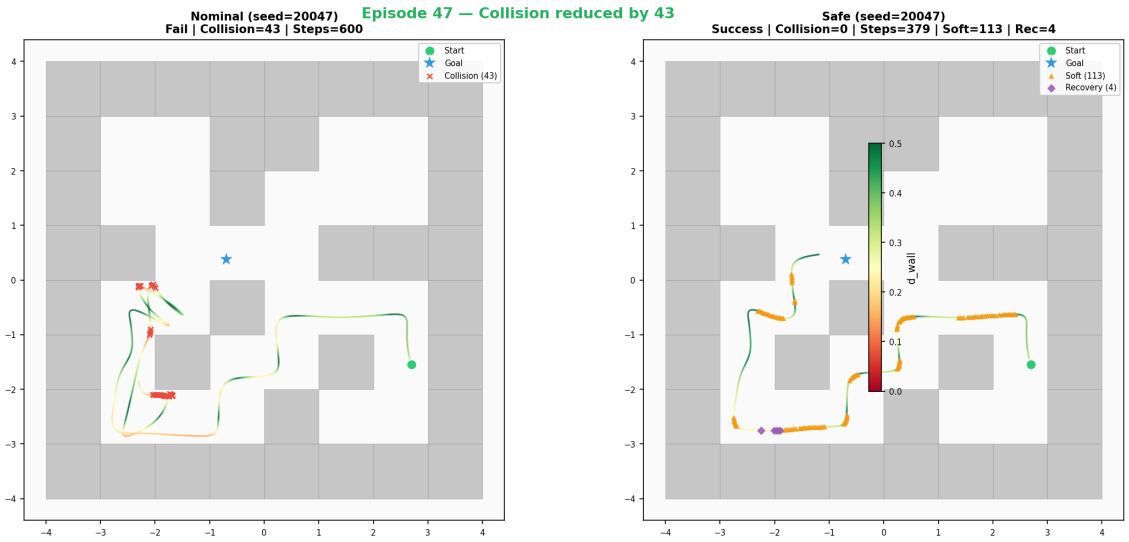

Figure D.1: Representative Medium-maze episode. Nominal times out with repeated collisions, whereas Guardrail completes the task without collision.

Maze Standard Evaluation Diagnostics

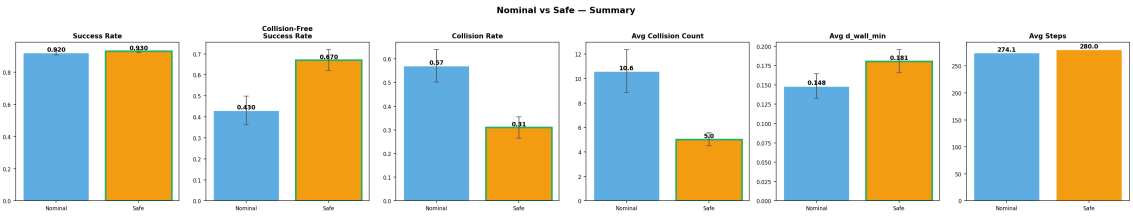

Figure D.2: Standard-evaluation summary for the Medium maze.

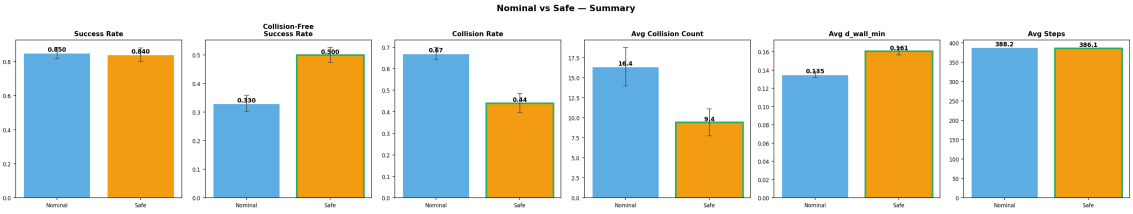

Figure D.3: Standard-evaluation summary for the Large maze.

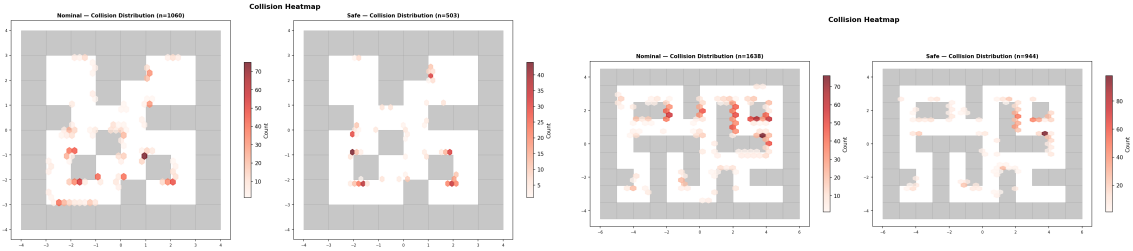

Figure D.4: Collision heatmaps (left: Medium; right: Large).

Additional Panda Robot Visualizations

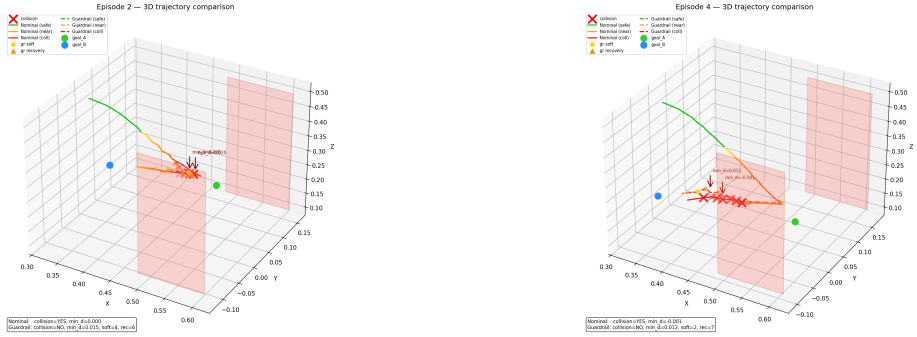

Figure D.5: Additional 3D trajectory visualizations for Panda reach-obstacle.

Additional Meta-World Evaluation

Table D.1: Meta-World baseline comparison: 50 official MT1 task instances.

Method	**Task success (95% CI)**	**Hard-frame ratio (95% CI)**	**Collision rate (episode)**
Nominal (behavior cloning)	0.880 [0.78, 0.96]	0.272 [0.21, 0.33]	0.940
Ours (full guardrail)	**0.940 [0.86, 1.00]**	**0.006 [0.00, 0.01]**	**0.160**
Recovery RL (ε=0.5)	0.000	0.002	0.060
Recovery RL (ε=0.8)	0.900	0.199	0.980

Note. Recovery RL confidence intervals are omitted because $\varepsilon = 0.5$ is a degenerate operating point and $\varepsilon = 0.8$ is a post-hoc threshold sweep on the same trained model.