UPPS-RL: Unified Predictive and Passive Safety In Quadrupedal Locomotion Control Using Reinforcement Learning

Peiyu Yang

Delft University of Technology

TU Delft

UPPS-RL: Unified Predictive and Passive Safety In Quadrupedal Locomotion Control Using Reinforcement Learning

by

Peiyu Yang

5992214

M.Sc. Robotics

Supervisor:	Dr. Cosimo Della Santina
	Dr. Jiatao Ding
Project Duration:	Mar. 2025 - Oct. 2025
Faculty:	Cognitive Robotics, Mechanical Engineering, TU Delft

Cover:	Unitree Go2 robot from Unitree website (modified): https://www.unitree.com/go2
Style:	TU Delft Style Cover (modified) combined with IEEE Conference Main Text

TU Delft

UPPS-RL: Unified Predictive and Passive Safety in Quadrupedal Locomotion Control Using Reinforcement Learning

Peiyu Yang

***Abstract*— Safe quadrupedal locomotion control with reinforcement learning (RL) has attracted increasing attention in recent years, where existing approaches can be broadly categorized into recovery RL, distributional RL, and constrained RL. However, recovery RL cannot provide predictive safety guarantees; distributional RL lacks passive safe performance; and constrained RL-while capable of both safety-often restricts exploration. To address these limitations, we propose UPPS-RL, a unified framework that integrates predictive and passive safety into quadrupedal locomotion control through three main components: a risk-aware task-level policy, a self-supervised risk network, and a risk-triggered recovery policy, forming a hierarchical control architecture that embeds unified safety without imposing explicit exploration constraints. Extensive simulations across composite scenarios, including steps, pit, slope, and rough plane terrains, demonstrate that UPPS-RL significantly suppresses catastrophic failures while maintaining a favorable trade-off between robustness and efficiency.**

I. INTRODUCTION

In recent years, quadrupedal robots have attracted increasing attention in robotics research due to their remarkable agility and adaptability to complex terrains [1]–[3]. These features make them suitable for applications such as disaster assistance [4], environmental protection [5], and industrial inspection [6]. However, to enable these applications, it is crucial to achieve reliable and safe locomotion control in complex environments. In particular, quadrupedal robots must not only provide efficient locomotion capabilities, but also satisfy two critical requirements: predictive safety [7] and passive safety [8]–[10]. Predictive safety, achieved via risk-aware control, involves perceiving and modeling uncertainties and hazards in the environment, thereby enabling risk avoidance and more robust control strategies. Passive safety, by contrast, concerns the robot's ability to avert catastrophic outcomes once it has entered risky conditions, including the prevention of falls, collisions, and the avoidance of joint over-torque, thereby mitigating hardware damage and mission disruption.

Although significant progress has been made in enhancing quadrupedal locomotion on complex terrains through both model-based approaches [11], [12] and learning-based approaches [13], [14], research on predictive and passive safety remains insufficient, with limitations arising from multiple factors. In model-based approaches, safety is generally enforced by incorporating manually designed constraints into the control framework. For instance, Grandia et al. [15] integrate control barrier functions (CBFs) into a trajectory-tracking framework to address the stepping-stone locomotion problem; Liao et al. [16], who propose a CBF-based nonlinear model predictive control (MPC) with dual optimization to handle polyhedral obstacles; and [17], [18] designed task-specific CBFs for stair-climbing scenarios. While these methods can theoretically guarantee safety, their implementation relies on manually crafted dynamic models and is fragile to disturbances and modeling errors, which impose considerable computational demands and complicate practical deployment. In addition, since risk is inherently

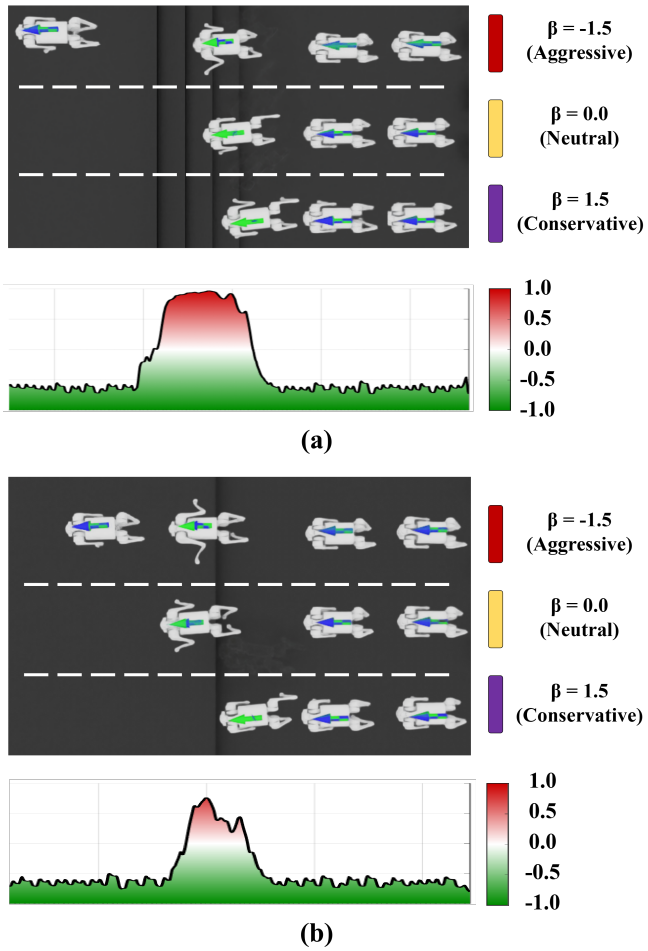

Fig. 1. Performance of UPPS-RL in different environments. (a) Execution in a step environment. (b) Execution in a pit environment. By adjusting β, the agent exhibits different risk preferences ranging from aggressive ($\beta = -1.5$) to conservative ($\beta = 1.5$). In addition, a neural network is designed to provide real-time perception of risk values, enabling the controller to anticipate hazards and adapt its actions accordingly.

Peiyu Yang is with Cognitive Robotics, Delft University of Technology, Building 34, Mekelweg 2, 2628 CD Delft, Netherlands (e-mail: p.yang-5@student.tudelft.nl).

hard to model and highly uncertain, existing model-based control methods, to the best of our knowledge, still lack explicit mechanisms for its representation and handling.

In contrast, learning-based methods offer more diverse solutions for risk awareness and safety. For predictive safety, model-free RL alleviates the reliance on accurate dynamics models, enabling agents to implicitly acquire risk information through extensive interactions with the environment. Such information is subsequently utilized to refine control strategies. For instance, several studies in [19]–[21] integrate distributional learning into RL frameworks to capture risk-related features from return distributions. In addition, Zhang et al. [22] propose a terrain-aware teacher–student learning framework that integrates a risk assessment network, thereby enhancing the stability of quadrupedal robots on complex terrains. On the other hand, regarding passive safety, a widely studied paradigm is recovery RL, which adopts a hierarchical structure where multiple policies are integrated and adaptively switched according to different risk levels to keep the robot confined within a predefined safe set. A key limitation of the above approaches, however, lies in the absence of an explicit integration between predictive and passive safety. As another approach, constrained RL [23]–[25] explicitly accounts for both predictive and passive safety. This is achieved by incorporating mechanisms such as control Lyapunov functions (CLFs), CBFs, or constrained Markov decision processes (CMDPs) into the learning process. Nevertheless, constrained RL typically enforces constraints prior to policy convergence, which restricts exploration and can lead to suboptimal returns [26], [27].

Therefore, my research question is: How can predictive and passive safety be achieved through RL while preserving exploration efficiency? To address this question, we propose the unified predictive and passive safety RL (UPPS-RL) framework, which achieves this integration through a hierarchical control scheme. Specifically, we employ a risk-aware RL policy as the task-level policy, parameterized by a tunable risk-preference that allows behaviors ranging from conservative to aggressive. To mitigate potential unsafe behaviors arising from the task-level policy, we design a recovery policy that intervenes when necessary. Finally, we introduce a self-supervised risk value network that quantifies risk levels, enabling adaptive adjustment of risk preference and generation of recovery commands. Through the integration of these components, our framework establishes a unified mechanism that combines adaptive risk modulation with formal safety assurance, thereby achieving both predictive and passive safety while preserving exploration efficiency.

In brief, the main contributions of this work are summarised as follows:

- A set of functional modules is developed, comprising a task-level policy for risk-aware locomotion, a self-supervised neural network for risk quantification, and a safety-oriented recovery policy for high-risk conditions;
- A hierarchical mechanism is proposed to coordinate functional modules, enabling the integration of predictive and passive safety control;

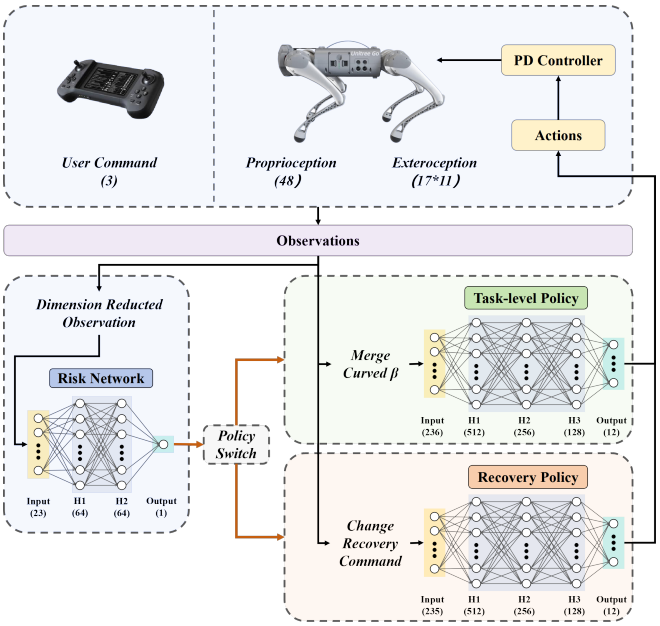

Fig. 2. Overview of the UPPS-RL framework. The framework consists of three modules: 1) **Task-level Policy**, trained with DPPO to produce behaviors ranging from conservative to aggressive, depending on the risk-preference parameter; 2) **Recovery Policy**, designed to take over control in risky condition, and maintain safety when tracking the recovery command that lowers the risk level; and 3) **Risk Network**, trained in a self-supervised manner to estimate the risk level in real-time. During deployment, the risk value network provides a modulation of risk-preference for the task-level policy, a recovery command for the recovery policy, and a switching signal for activating the dual-policy scheme. In this scheme, the task-level policy is executed under low-risk conditions, while the recovery policy is activated once the estimated risk exceeds a threshold to ensure safety.

- Extensive simulation experiments are conducted to comprehensively validate the functionality of each sub-module as well as the overall framework.

The remainder of this paper is organized as follows. Section II reviews the related works. Section III presents the overall framework and preliminaries. Section IV, V, VI, and VII describe the proposed methodology in detail, including the design of each sub-module and the overall system architecture. Section VIII reports the experimental results. Section IX concludes the paper, and Section IX presents the acknowledgments.

II. Related Works

A. Toward Predictive Safety in Reinforcement Learning

Predictive safety aims to prevent agents from entering unsafe regions by anticipating potential risks in advance. This objective is commonly addressed through risk-aware RL. Unlike traditional RL, which focuses solely on maximizing long-term return, risk-aware RL explicitly accounts for uncertainty during both training and deployment, aiming to maximize performance while mitigating catastrophic tail risk. Risk awareness in such methods is typically realized through model-based approaches, Bayesian formulations, and distributional RL.

Incorporating risk using model-based RL [28], [29] is a natural and effective strategy. One of the general paradigms

leverages either prior physical models or data-driven learned models to characterize environmental uncertainty and map it into an optimizable risk measure. This enables sampling procedures that penalize or reweight highly uncertain successor states, steering the learned policy toward high-confidence regions and reducing worst-case damages.

Within Bayesian approaches to risk-aware RL, the core objective is to represent decision risk arising from parametric uncertainty via the posterior distribution and, on that basis, to embed computable risk criteria and guarantees into policy evaluation and improvement. For parameter uncertainty, one line of work derives second-order approximations under a Dirichlet posterior to construct confidence bounds on value functions for risk-sensitive evaluation and optimization [30], yielding a direct and tractable data-to-interval mapping. Another line adopts quantile objectives and Conditional Value-at-Risk (CVaR) as a unified framework and, under Dirichlet priors, develops approximately solvable formulations with efficient evaluation [31]. Overall, these methods integrate posterior uncertainty into the optimization objective, conferring statistical robustness [32].

In recent years, distributional RL [33]–[36] has become an important direction for risk-aware learning. Unlike traditional frameworks that estimate only the expected return, distributional RL directly learns the complete return distribution, whose semantics are specified by the recursive distributional Bellman equation; value-based methods perform value iteration on this distribution and typically compare actions using the expectation of that distribution. Following Dabney et al. [35], the distribution representation in distributional RL can be classified into three categories: categorical approximations, quantile-regression representations that fit the cumulative distribution at fixed quantile points, and continuous representations that learn an implicit quantile function at arbitrary quantiles. In the domain of robotic control, distributional RL is often instantiated within the classical actor–critic architecture with a distributional critic, thereby leveraging both the shape and the location of the return distribution during policy improvement to realize risk-sensitive characteristics. Benefiting from its compatibility with the actor–critic framework widely used in contemporary quadruped locomotion control and its direct characterization of risk preference, several studies have applied it to quadrupedal locomotion—including risk-aware control [19], risk-averse control [20], multi-terrain robust locomotion [37], and bipedal walking [21]. While constrained RL is capable of achieving predictive safety through its constraint formulation [38], we present the related work in the next subsection, where its relation to recovery RL can be more clearly articulated.

Our method is consistent with distributional RL, but it departs from prior work that either prioritizes robustness without explicit control of risk preference, or relies on hand-tuned risk-attitude parameters to obtain differentiated policies across the full risk spectrum. We introduce a self-supervised risk network to adaptively regulate risk preference over the spectrum. By explicitly leverage the risk value, the proposed approach achieves a unified balance between performance and safety.

B. Toward Passive Safety in Reinforcement Learning

Passive safety aims to ensure that, once an agent enters a potentially unsafe region, mechanisms steer the robot back to the safe region without violating safety. In reinforcement learning, this objective has been widely studied and is commonly pursued through two methodological paradigms: end-to-end and hierarchical approaches, corresponding to constrained RL and recovery RL, respectively. Constrained RL enforces safety within a single policy-learning pipeline by formulating the problem as a CMDP, and seeking to maximize expected return subject to prescribed cost or safety constraints. Representative formulations include Lagrangian methods [39], [40], which solve a saddle-point problem by jointly updating policy parameters and Lagrange multipliers; penalty and interior-point methods [41]–[43], which embed constraints in the objective via external penalties or barrier terms to promote feasibility; alternating-optimization schemes, which alternate between reward improvement and cost reduction according to current feasibility; and projection-based methods [44], which take an unconstrained step followed by a projection or proximal update back to an approximate feasible set.

To address the suboptimality caused by constrained RL, recovery RL approaches [45], [46] have been advanced as a salient paradigm for safe RL. These methods typically rely on system dynamics together with control-theoretic safety certificates [47], in order to determine whether the system has entered a trigger set, and, upon detecting potential violations, switch to a recovery policy to ensure safety. However, such model-based switching mechanisms depend on accurate system identification and complex constraint design, imposing substantial requirements on prior knowledge and computational resources and thereby limiting scalability to high-dimensional, complex platforms such as quadrupeds. To alleviate these demands, a line of work replaces explicit models with learned safety critics and recovery policies, thereby preserving, to a large extent, the advantages of model-free training. In this context, He et al. [27] proposed the ABS framework, which is fully model-free and hierarchical, explicitly accounts for the interaction between a reach–avoid (RA) value network and a recovery policy, leading to more competitive overall performance. Nevertheless, the aforementioned hierarchical schemes remain reactive, focusing on post-entry safety rather than predictive, risk-aware control; moreover, including [27] among others, they have yet to couple the learned risk indicator with the task-level policy, which limits the latter's ability to fully exploit the information provided by the safety module. Accordingly, adhering to a fully model-free hierarchical control architecture, we propose a framework that integrates a risk value network with a risk-aware task-level policy, resulting in a more tightly coupled hierarchy that provides both predictive and passive safety.

TABLE I

NOMENCLATURE: SYMBOLS AND ABBREVIATIONS USED IN THIS WORK

Symbol / Abbrev.	Meaning
SYMBOLS	
$\mathcal{S}, \mathcal{A}, \mathcal{O}$	State space; action space; observation space
$\mathcal{C}$	Command space
s, a	State; action
$f(s,a)$	Dynamics
$P(s' \mid s,a)$	Transition kernel
$r(s,a), c(s,a)$	Reward and cost
$\gamma \in (0,1)$	Discount factor
$\pi_\theta(a \mid s)$	Task-level policy with neural network weights θ
$V^\pi(s), Q^\pi(s,a)$	State-/action-value under policy π
$Z^\pi(s,a)$	Return distribution
$\mathrm{VaR}_\alpha(\cdot)$	Value-at-Risk at level α
$\mathrm{CVaR}_\alpha(\cdot)$	Conditional Value-at-Risk at level α
$\rho_\psi(\cdot)$	Wang distortion with parameter ψ
$\mathcal{R}, \mathcal{T}, \mathcal{RT}$	Risk set; tracking set; risk–tracking set
π_{rec}	Recovery policy
ABBREVIATIONS	
RL	Reinforcement Learning
MDP, CMDP	Markov Decision Process; Constrained MDP
PPO	Proximal Policy Optimization
DPPO	Distributional Proximal Policy Optimization
MPC	Model Predictive Control
CLF, CBF	Control Lyapunov Function; Control Barrier Function
MSE	Mean-squared Error

III. PRELIMINARIES AND OVERVIEW

A. Preliminaries

1) Nomenclature: In the following text, sets and spaces are denoted by calligraphic capitals, while number sets are using blackboard bold. Matrices and vectors are denoted using bold symbols (e.g., $\mathbf{A}$ for matrices and $\boldsymbol{v}$ for vectors). The notation $(\cdot)[k]$ refers to the k-th element of a vector or a matrix. Scalars are represented using regular italic symbols (e.g., a). Subscripts and superscripts are used to indicate specific indices or contexts, where necessary. In pseudocode, parameters are denoted using italic symbols (e.g., *param*), while functions are represented using regular text (e.g., FunctionName)

Specifically, the key symbols and abbreviations used in this work are summarized in Table I.

2) Dynamics: Although our method does not rely on a parametric model, a brief specification of the robot dynamics is useful for clarifying the relationship between state spaces and action spaces. Let $\boldsymbol{s}_t \in \mathcal{S}$ denote the robot state vector at time t, where the state space satisfies $\mathcal{S} \subseteq \mathbb{R}^{n_s}$ and n_s is the state dimension. Likewise, let $\boldsymbol{a}_t \in \mathcal{A}$ denote the action vector at time t, where the action space satisfies $\mathcal{A} \subseteq \mathbb{R}^{n_a}$ and n_a is the action dimension. The robot's dynamics are modeled as

$$\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t), \qquad f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}, \tag{1}$$

where f denotes the dynamic transformation function that takes the current state–action pair $(\boldsymbol{s}_t, \boldsymbol{a}_t)$ as input and returns the next state $\boldsymbol{s}_{t+1}$.

3) Distribution and Risk Criteria: Let $Z(\boldsymbol{s})$ denote the return of an agent at state $\boldsymbol{s}$. Its cumulative distribution function and quantile function are:

$$F_Z(z) = \mathbb{P}(Z \le z), \qquad Q_Z(u) = \inf\{z : F_Z(z) \ge u\}, \tag{2}$$

where F_Z denotes the cumulative distribution function of Z, and Q_Z is the corresponding quantile function evaluated at probability level $u \in [0,1]$.

In the baseline view of risk, one often writes "probability $\times$ consequence" as $R = p \cdot C$, which can be understood as an expectation over outcomes with implicit weights. Two classical paradigms refine this baseline by emphasizing tail behavior or by reweighting probabilities.

(I) Tail-based measures: VaR [48] and CVaR [49]. For $\alpha \in (0,1)$, the *lower-tail VaR* is defined by

$$\mathrm{VaR}_\alpha(Z) = Q_Z(\alpha), \tag{3}$$

the return threshold below which outcomes fall with probability α (i.e., the α-quantile of Z). The corresponding *lower-tail CVaR* averages the same tail:

$$\mathrm{CVaR}_\alpha(Z) = \mathbb{E}[Z \,|\, Z \le Q_Z(\alpha)] = \frac{1}{\alpha}\int_0^\alpha Q_Z(u)\,du \quad . \tag{4}$$

(II) Distortion-based measures: the Wang transform [50]. Distortion risk measures first reweight probability/quantile levels via a monotone map $\psi : [0,1] \to [0,1]$ and then integrate the quantile function:

$$\rho_\psi(Z) = \int_0^1 Q_Z(u)\,d\psi(u). \tag{5}$$

The Wang transform uses the "normal-equivalent" distortion

$$\psi(u) = \Phi\big(\Phi^{-1}(u) + \lambda\big), \tag{6}$$

where Φ is the standard normal cumulative distribution function and λ controls risk attitude: $\lambda < 0$ increases weight on the lower tail, whereas $\lambda > 0$ emphasizes upper-tail gains. Unlike class (I), which targets a specific lower-tail quantile or its mean, the distortion class redistributes weight across all quantiles through ψ, thereby encoding tail importance directly into the overall evaluation of Z.

4) Policy: A policy maps observations to a distribution over actions,

$$\pi_\theta : \mathcal{O} \to \Delta(\mathcal{A}), \tag{7}$$

where $\mathcal{O}$ is the observation space and $\Delta(\mathcal{A})$ denotes probability measures over the action space. At each timestep, the agent samples an action

$$a_t \sim \pi_\theta(\cdot \mid o_t). \tag{8}$$

The environment dynamics are defined by the transition kernel:

$$P(s' \mid s,a), \tag{9}$$

which specifies the probability of reaching state s' given state s and action a. Together with a reward function $r_{\text{trk}} : \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to \mathbb{R}$, a discount factor γ_{RL}, a command set $\mathcal{C}$, and a command distribution $p_{\mathcal{C}}$, the policy is optimized to maximize the expected discounted return:

$$J(\pi_\theta) = \mathbb{E}_{\substack{v_{\text{cmd}} \sim p_{\mathcal{C}} \\ a_t \sim \pi_\theta(\cdot|o_t, v_{\text{cmd}})}} \left[\sum_t \gamma_{\text{RL}}^t \, r_{\text{trk}}(s_t, a_t, v_{\text{cmd}}) \right]. \quad (10)$$

5) Set Definitions and Trigger Mechanisms: Inspired by the classical reach–avoid problem in [27], we introduce similar definitions tailored to our setting. In accordance with the control task considered in this work—ensuring robot safety while tracking velocity—we introduce three subsets of the state space $\mathcal{S}$: the *risk set* $\mathcal{R} \subseteq \mathcal{S}$, the *tracking set* $\mathcal{T} \subseteq \mathcal{S}$, and the *risk–tracking set* (RT set) $\mathcal{RT}_\pi \subseteq \mathcal{S}$.

The risk set collects unsafe states (e.g., excessive tilt, collisions, and feet stumble) and is encoded as the zero–superlevel set of a Lipschitz function ζ:

$$\mathcal{R} := \{ s \in \mathcal{S} : \ \zeta(s) \geq 0 \}, \qquad \zeta : \mathcal{S} \to \mathbb{R} \quad (11)$$

The tracking set gathers states that correctly track the velocity command and is encoded as the zero–sublevel set of a Lipschitz function l:

$$\mathcal{T} := \{ s \in \mathcal{S} : \ l(s) \leq 0 \}, \qquad l : \mathcal{S} \to \mathbb{R} \quad (12)$$

The RT set is related to policy. For the nominal policy π, the RT set is the set of initial states whose closed-loop trajectories $\xi_s^\pi(t)$ stay in $\mathcal{T}$ while avoiding $\mathcal{R}$ over the horizon:

$$\mathcal{RT}_\pi := \left\{ s \in \mathcal{S} : \ \xi_s^\pi(t) \in \mathcal{T} \ \wedge \ \xi_s^\pi(t) \notin \mathcal{R} \ \ \forall t \in [0, T] \right\}. \quad (13)$$

Trigger mechanism is a binary switching map $\text{Trig} : \mathcal{S} \to \{0, 1\}$ selects between the nominal policy π and a recovery policy κ:

$$m_t = \text{Trig}(s_t), \qquad \mu(s_t) = (1 - m_t)\, \pi(s_t) + m_t \, \kappa(s_t). \quad (14)$$

The objective of the trigger mechanism is to ensure risk avoidance and tracking maintenance under μ. Let the robust interior of the tracking set be $\mathcal{T}^\delta := \{ s : \ l(s) \leq -\delta \}$ with $\delta > 0$. Then the guarantee is stated as

$$s_0 \in \mathcal{T}^\delta \implies \xi_{s_0}^\mu(t) \in \mathcal{T} \ \wedge \ \xi_{s_0}^\mu(t) \notin \mathcal{R}, \quad \forall t \in [0, T], \quad (15)$$

and, desirably, the inclusion $\mathcal{RT}_\pi \subseteq \mathcal{RT}_\mu \subseteq \mathcal{T} \setminus \mathcal{R}$, where $\mathcal{RT}_\mu := \{ s : \ \xi_s^\mu(t) \in \mathcal{T} \wedge \xi_s^\mu(t) \notin \mathcal{R}, \ \forall t \in [0, T] \}$, highlights that the recovery mechanism can only enlarge or match the capability region induced by π.

6) Risk Value and Time-Discounted Reach-Avoid Bellman Equation: We define the risk value under the nominal policy π, denoted $V_{\text{risk}}^\pi(s)$, whose zero–sublevel set strictly coincides with the nominal RT set:

$$V_{\text{risk}}^\pi(s) \leq 0 \iff s \in \mathcal{RT}_\pi(\zeta; l). \quad (16)$$

Following [27], we adopt a contracted time–discounted risk Bellman equation to characterize and compute this function. Let the successor state be $s^+ = f(s, \pi(s))$ and choose a discount $\gamma_{\text{risk}} \in (0, 1)$; then the value can be represented as:

$$V_{\text{risk}}^\pi(s) = \gamma_{\text{risk}} \max\{ \zeta(s), \min\{ l(s), V_{\text{risk}}^\pi(f(s, \pi(s))) \} \} + (1 - \gamma_{\text{risk}}) \max\{ l(s), \zeta(s) \}. \quad (17)$$

Since $\gamma_{\text{risk}} < 1$, the corresponding operator is a contraction in the sup-norm, hence the fixed point exists and is unique, and the value iteration $V_{k+1} = \mathcal{T}_{\text{risk}}^\pi(V_k)$ converges; formal proofs can be found in [27] and Appendix A of [51].

B. Overview

The architecture of the proposed UPPS-RL framework is illustrated in Fig. 2, which comprises three primary modules: a risk network, a task-level policy, and a recovery policy. The task-level policy π_T is trained under a distributional RL scheme to track twist commands while taking a risk-preference parameter as an additional input, thereby enabling tunable aggressiveness–conservatism (details in Section VIII-A). The recovery policy π_R, designed as a safety-first conservative controller, tracks a recovery command under safety constraints to steer the system back toward low-risk regions (details in Section VIII-C). Both policies output joint position targets in $\mathbb{R}^{12}$, which are mapped by a PD controller to low-level actuation commands.

In our setting, when the risk value is below a switching threshold, the system executes the action of the task-level policy; the policy's risk-preference parameter is generated by passing the risk value through a shaping curve, so that the controller is more aggressive at low risk and more conservative at moderate or high risk. When the risk value exceeds the switching threshold, the system executes the recovery policy, whose recovery command is obtained by a search operation performed by the risk network. In short, the selection can be summarized as:

$$\beta_t = \psi\big(V_{\text{risk}}^\pi(s_t)\big), \qquad c_t^{\text{rec}} = \text{Search}(\mathit{RiskNet}, s_t), \quad (18)$$

$$a_t = \begin{cases} \pi_T(o_t, c_t, \beta_t), & \text{if } V_{\text{risk}}^\pi(s_t) \leq \tau_{\text{sw}}, \\ \pi_R(o_t, c_t^{\text{rec}}), & \text{if } V_{\text{risk}}^\pi(s_t) > \tau_{\text{sw}}. \end{cases} \quad (19)$$

To summarize, the UPPS-RL is constructed by sequentially training three modules and then applying them jointly; the training and merging procedures are as follows:

- The task-level policy is trained under a DPPO framework, with its configuration defined in Section IV. Its core advantage is that the critic evaluates the overall distribution of returns and applies a distortion according to a sampled risk-preference parameter, yielding risk-aware characteristics.
- The risk network is trained via a self-supervised learning method, using the risk-neutral task-level policy as the reference policy for data collection and labeling, which are used to indicate the risk value. The training process is detailed in Section V.
- The recovery policy is trained under a PPO framework, with its configuration defined in Section VI.

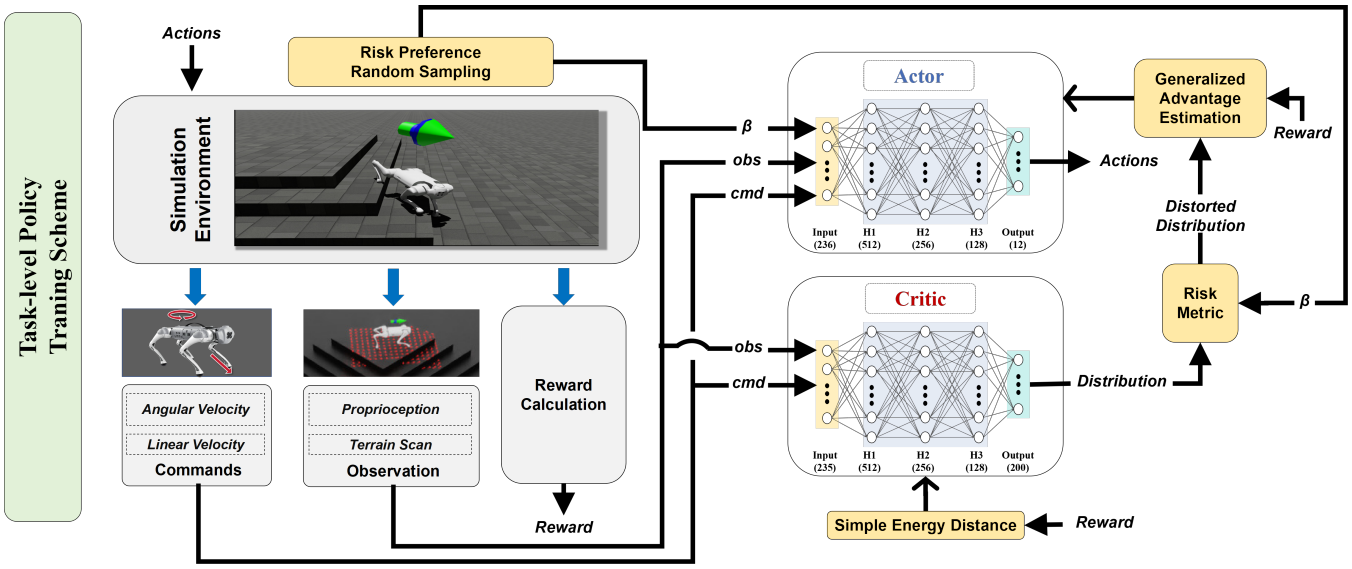

Fig. 3. Task-level policy training framework based on DPPO. The observations consist of proprioception, terrain-scan information, and twist commands. These inputs are fed into the actor–critic networks, where the actor outputs robot actions and the critic provides value estimates. The distributional output of the critic is further shaped by a risk metric under randomly sampled risk preferences, which is then used for generalized advantage estimation.

- The module composition method—including the mapping from risk value to the risk-preference parameter and the generation of the recovery command—is presented in Section VII.

IV. Task-level Policy

As a key component of the UPPS-RL framework, the task-level policy provides predictive safety assurance. Given motivations in Section II-A, we adopt the distributional RL approach in [19], called distributional proximal policy optimization (DPPO), to obtain risk-aware control: within a standard actor–critic framework, the scalar critic is replaced by the distributional value network that models the return distribution, and this distribution is then distorted according to a prescribed risk metric and a specified risk-preference parameter. At deployment, by tuning the risk-preference parameter, the policy can be explicitly adjusted to achieve an aggressiveness–conservatism trade-off.

This section presents the training details of the task-level policy, including the actor, the critic, the risk metric, the reward design, and the training setup.

A. Actor

As shown in Fig. 3, the actor is a three-layer fully connected MLP with hidden widths $[512, 256, 128]$. Its observation vector includes: base linear velocity $\boldsymbol{v}$, base angular velocity $\boldsymbol{\omega}$, the gravity projection $\boldsymbol{g}$ expressed in the body frame, the body-frame twist command $\boldsymbol{c}_t^{\text{body}}$, the joint positions $\boldsymbol{q}$, the joint velocities $\dot{\boldsymbol{q}}$, the previous action $\boldsymbol{a}_{t-1}$, the ray-caster height samples $\boldsymbol{h}_t$ in the body frame within a 1.6 m (x-axis) by 1.0 m (y-axis) area, and **the risk-preference parameter** β_t. Formally, the observation can be summarized as:

$$\boldsymbol{o}_t^{\text{Task-Actor}} = \left[\boldsymbol{v}_t;\ \boldsymbol{\omega}_t;\ \boldsymbol{g}_t;\ \boldsymbol{c}_t^{\text{body}};\ \boldsymbol{q}_t;\ \dot{\boldsymbol{q}}_t;\ \boldsymbol{a}_{t-1};\ \boldsymbol{h}_t;\ \beta_t\right]. \quad (20)$$

The actor outputs the policy action as a 12-dimensional joint target vector. Consistent with Section III-B, the joint targets are tracked by a PD controller, which maps joint targets to the low-level joint torques:

$$\boldsymbol{\tau}_t = \mathbf{K}_p\left(\boldsymbol{a}_t - \boldsymbol{q}_t\right) + \mathbf{K}_d\left(\dot{\boldsymbol{a}}_t - \dot{\boldsymbol{q}}_t\right), \quad (21)$$

where $\boldsymbol{a}_t \in \mathbb{R}^{12}$ is the joint-target vector, $\boldsymbol{q}_t$, $\dot{\boldsymbol{q}}_t$ are the measured joint positions and velocities, $\dot{\boldsymbol{a}}_t$ is the desired joint velocity which is set to zero, and $\mathbf{K}_p, \mathbf{K}_d \in \mathbb{R}^{12\times 12}$ are positive gain matrices.

We train the actor with the PPO-Clip objective [52], maximizing

$$\mathcal{L} = \min\left(\frac{\pi_\phi(a\,|\boldsymbol{o}_t^{\text{Task-Actor}})}{\pi_{\phi_{\text{old}}}(a\,|\boldsymbol{o}_t^{\text{Task-Actor}})} A^{\pi_{\phi_{\text{old}}}}(\boldsymbol{o}_t^{\text{Task-Critic}}, a),\right.$$
$$\left. g\big(\varepsilon,\ A^{\pi_{\phi_{\text{old}}}}(\boldsymbol{o}_t^{\text{Task-Critic}}, a)\big)\right), \quad (22)$$

where,

$$g(\varepsilon, A) = \begin{cases} (1+\varepsilon)A, & A \geq 0, \\ (1-\varepsilon)A, & A < 0. \end{cases}$$

We use a truncated version of generalized advantage estimation (GAE) [53], as employed in [54] and [55]. Specifically, for horizon T, it is computed as:

$$A^\pi(\boldsymbol{o}_t^{\text{Task-Critic}}, \boldsymbol{a}_t) = \sum_{l=0}^{T-t-1} (\lambda\gamma)^l\, \delta_{t+l}^\pi, \quad (23)$$

where

$$\delta_t^\pi = r_t + \gamma V(\boldsymbol{o}_{t+1}^{\text{Task-Critic}}) - V(\boldsymbol{o}_t^{\text{Task-Critic}}), \quad (24)$$

γ is the discount factor, λ is the bias/variance trade-off hyperparameter, r_t is the immediate reward, and $V(\cdot)$ is the state value function.

Since the critic in the proposed framework produces a full return distribution, it is not directly compatible with the scalar value function $V(\boldsymbol{o})$ used in standard PPO. Accordingly, we introduce a risk metric (detailed in Section IV-C) that operates on the critic's distributional output, applies a distortion, and yields a scalar value, which is then used as $V(\boldsymbol{o}^{\text{Task-Critic}})$.

B. Critic

We adopt Quantile Regression DQN (QR-DQN) [56] as the distributional critic, which uses uniform weights $1/N$ and learns the quantile locations $\{\theta_i(s)\}_{i=1}^{N}$. The return distribution is represented as

$$Z_\theta(s) \;=\; \frac{1}{N}\sum_{i=1}^{N}\delta_{\theta_i(s)}, \tag{25}$$

where $\delta_{\theta_i(s)}$ denotes the Dirac measure at $\theta_i(s)$.

Different from the actor, the critic is used solely to predict the discounted return at state s and is invariant to the choice of risk-preference; therefore, the critic's observation set does not include the risk-preference parameter. Its observation set can be expressed as follows:

$$\boldsymbol{o}_t^{\text{Task-Critic}} = \big[\boldsymbol{v}_t;\ \boldsymbol{\omega}_t;\ \boldsymbol{g}_t;\ \boldsymbol{c}_t^{\text{body}};\ \boldsymbol{q}_t;\ \dot{\boldsymbol{q}}_t;\ \boldsymbol{a}_{t-1};\ \boldsymbol{h}_t\big]. \tag{26}$$

Motivated by [34], we train the critic with a simple energy–distance objective, as

$$\mathcal{L}_{\text{critic}} = 2\,\mathbb{E}_{i,j}\big[\,|\theta_i - T\theta_j|\,\big] - \mathbb{E}_{i,j}\big[\,|T\theta_i - T\theta_j|\,\big] - \mathbb{E}_{i,j}\big[\,|\theta_i - \theta_j|\,\big], \tag{27}$$

where $\theta_i, \theta_j \sim Z_\theta(s_t)$ are samples from the current critic distribution, $T\theta_j$ denotes samples from the $\mathrm{SR}(\lambda)$ target distribution obtained by discounting and shifting with r_t and γ, and $\mathbb{E}_{i,j}$ is the empirical average over sample pairs.

C. Risk Metric

To enable both upper-tail (aggressive) and lower-tail (conservative) reweighting, we adopt a distortion-based risk metric. Specifically, we use the Wang metric to distort the distribution produced by the critic. Let $Z_\theta(s)$ denote the return distribution at state s with quantile function $Q_{Z_\theta(s)}$. For a risk-preference parameter $\beta \in \mathbb{R}$, define

$$\psi_\beta(u) = \Phi\big(\Phi^{-1}(u) + \beta\big), \qquad u \in [0,1], \tag{28}$$

where Φ is the standard normal CDF. The Wang risk value is then

$$V_\beta(\boldsymbol{o}_t^{\text{Task-Critic}}) = \sum_{i=1}^{N}\big[\psi_\beta(\tau_i) - \psi_\beta(\tau_{i-1})\big]\ \theta_i(\boldsymbol{o}_t^{\text{Task-Critic}}), \qquad 0 = \tau_0 < \tau_1 < \cdots < \tau_N = 1. \tag{29}$$

which emphasizes upper-tail returns when $\beta > 0$ (more aggressive) and lower-tail returns when $\beta < 0$ (more conservative). During training, β is sampled from $[-1.5, 1.5]$. This range induces substantial and practically useful distortions of the critic's return distribution; larger magnitudes drive most mass into the extreme head or tail and offer little additional benefit, hence an expansion of the range is unnecessary.

D. Rewards

Inspired by [27] and [19], we set the reward function to consist of three components: penalty, task, and regularization terms:

$$r_{\text{total}} = r_{\text{penalty}} \cdot \Delta t + r_{\text{task}} \cdot \Delta t + r_{\text{regularization}} \cdot \Delta t, \tag{30}$$

where Δt is the rendering time step.

1) Penalty: We penalize robot collisions and foot-stumbles:

$$r_{\text{penalty}} = -3.0 \cdot\ r_{\text{collication}} - 1.5 \cdot\ r_{\text{stumble}}. \tag{31}$$

The two components are

$$r_{\text{collication}} = \frac{1}{F_{\max}}\sum_{i\in\mathcal{U}}\big\|\boldsymbol{f}_i\big\|_2, \tag{32}$$

$$r_{\text{stumble}} = \frac{1}{|\mathcal{F}|}\sum_{j\in\mathcal{F}}\mathbf{1}\left\{\frac{\ell_j}{v_j+\varepsilon} > R_{\text{thr}}\right\}. \tag{33}$$

where $\mathcal{U}$ is the set of undesired-contact bodies, i.e. trunk, $\boldsymbol{f}_i \in \mathbb{R}^3$ is the net contact force on body i, and $F_{\max} = 100N$ is the normalization constant. $\mathcal{F}$ is the set of feet; for foot j, the lateral and vertical force magnitudes are

$$\begin{aligned} \ell_j &= \sqrt{(f_j^x)^2 + (f_j^y)^2}, \\ v_j &= |f_j^z|; \end{aligned} \tag{34}$$

$\varepsilon = 10^{-6}$ is designed to avoid division by zero, and $R_{\text{thr}} > 4.0$ is a ratio threshold that flags excessive lateral loading. The indicator $\mathbf{1}\{\cdot\}$ returns 1 if the condition holds and 0 otherwise.

2) Task: The task reward encourages tracking of the linear- and angular-velocity commands while staying alive:

$$r_{\text{task}} = 1.5 \cdot r_{\text{lin\_xy}} + 1.0 \cdot r_{\text{yaw\_z}} + 0.1 \cdot r_{\text{alive}}. \tag{35}$$

The linear- and angular-velocity tracking reward terms are

$$r_{\text{lin\_xy}} = \exp\left(-\frac{Err_{lin}}{\sigma_{\text{trk}}}\right), \tag{36}$$

$$r_{\text{ang\_z}} = \exp\left(-\frac{Err_{yaw}}{\sigma_{\text{trk}}}\right), \tag{37}$$

where

$$Err_{lin} = \big\|\boldsymbol{v}_{\text{cmd}}^{xy} - \boldsymbol{v}^{xy}\big\|_2^2, \qquad Err_{yaw} = \big(\omega_{\text{cmd}} - \omega_z\big)^2.$$

Here $\boldsymbol{v}^{xy}, \boldsymbol{v}_{\text{cmd}}^{xy}$ are the measured and commanded linear velocities on the xy plane; ω_z and ω_{cmd} are the measured and commanded yaw rate; and $\sigma_{\text{trk}} = 0.25$ is the tuning constant.

The alive reward term is

$$r_{\text{alive}} = 1. \tag{38}$$

3) Regularization: In this work, we set the regularization rewards as:

$$\begin{aligned} r_{\text{regularization}} = &-2.0 \cdot r_{\text{lin\_z}} - 0.01 \cdot r_{\text{ang\_xy}} - 1.5e^{-4} \cdot r_{\text{torques}} \\ &- 5.0e^{-8} \cdot r_{\text{dof\_acc}} + 0.5 \cdot r_{\text{feet\_air\_time}} \\ &- 0.01 \cdot r_{\text{action\_rate}} - 0.5 \cdot r_{\text{gait}}. \end{aligned} \tag{39}$$

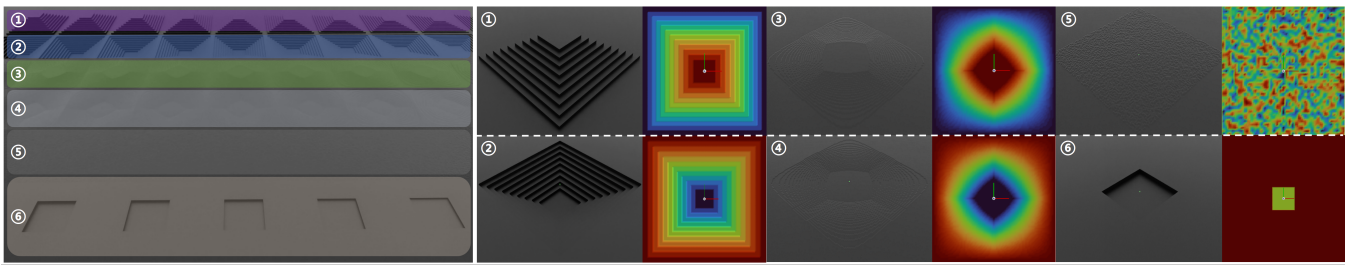

Fig. 4. The training environment is composed of multiple terrain types arranged in increasing difficulty. From top to bottom: (1) step, (2) inverted step, (3) slope, (4) inverted slope, (5) rough terrain, and (6) pit. Terrain elevation is visualized as color heatmaps to mitigate ambiguity due to finite simulation rendering fidelity; the heatmaps are sourced from the Isaac Sim official documentation.

The vertical linear- and lateral angular-velocity regularization terms are

$$r_{\text{lin\_z}} = v_z^2, \tag{40}$$

$$r_{\text{ang\_xy}} = \left\|\boldsymbol{\omega}^{xy}\right\|_2^2, \tag{41}$$

where v_z is the z-component of the body-frame linear velocity $\boldsymbol{v}$, and $\boldsymbol{\omega}^{xy} = [\,\omega_x,\ \omega_y\,]^\top$ stacks the x- and y-components of the body-frame angular velocity $\boldsymbol{\omega}$.

We limit the torque and joint acceleration with

$$r_{\text{torque}} = \|\boldsymbol{\tau}\|_2^2 = \sum_{j=1}^{n_{\text{dof}}} \tau_j^2, \tag{42}$$

$$r_{\text{acc}} = \|\ddot{\boldsymbol{q}}\|_2^2 = \sum_{j=1}^{n_{\text{dof}}} \ddot{q}_j^2, \tag{43}$$

where $\boldsymbol{\tau}$ denotes the applied joint torques and $\ddot{\boldsymbol{q}}$ the joint accelerations.

To encourage the robot to raise its legs, feet air time is added as a positive value to the regular reward:

$$r_{\text{air}} = \kappa \sum_{j\in\mathcal{F}} \left(t_j^{\text{air}} - \tau_{\text{air}}\right) \mathbf{1}\{\text{fc}_j\}, \tag{44}$$

where $\kappa = \mathbf{1}\{\|\boldsymbol{v}_{\text{cmd}}^{xy}\|_2 > v_{\min}\}$, $\tau_{\text{air}} = 0.6$, $v_{\min} = 0.1$; t_j^{air} is the last air time of foot j, and fc_j means the first contact of foot j in the current step, and $\mathbf{1}\{\cdot\}$ denotes the indicator function, which equals 1 if the condition inside holds and 0 otherwise.

The action-rate is also included as a regularizer:

$$r_{\text{act}} = \left\|\boldsymbol{a} - \boldsymbol{a}^-\right\|_2^2 = \sum_{j=1}^{n_{\text{dof}}} \left(a_j - a_j^-\right)^2, \tag{45}$$

where $\boldsymbol{a} \in \mathbb{R}^{n_{\text{dof}}}$ is the current actions and $\boldsymbol{a}^-$ is the previous actions.

To suppress the erroneous gait in which one leg remains airborne for an extended period, we add a gait regularization term:

$$r_{\text{gait}} = \kappa \cdot \max_{j\in\mathcal{F}} \left[t_{j,\text{cur}}^{\text{air}} - \tau_{\text{air}} \right]_+, \tag{46}$$

where $[x]_+ = \max(x, 0)$, $t_{j,\text{cur}}^{\text{air}}$ is the current air time of foot j, and $\tau_{\text{air}} = 0.8\,\text{s}$ is the threshold. The activation factor κ enables the penalty only when a nontrivial planar speed is commanded.

E. Training Setup

1) Simulator and Hardware: We trained the task-level policy in the Isaac Sim simulator using the Isaac Lab training framework. For the actor-critic algorithm, the hyperparameters were: PPO clipping ratio $\varepsilon = 0.2$, entropy coefficient $\beta_{entropy} = 0.01$, GAE parameter $\lambda = 0.95$, discount factor $\gamma = 0.95$, and learning rate $\eta = 1 \times 10^{-3}$. The training process was conducted on a laptop equipped with an Intel Core i9-13900HX (24 cores/32 threads), 16 GB RAM, and an NVIDIA GeForce RTX 4060 Laptop GPU (8 GB VRAM), running Ubuntu 20.04.6 LTS (kernel 5.15.0–139). 1,024 parallel agents were trained for 6,000 iterations, which required approximately 2.5 hours.

2) Environment: We adopt a parameterized suite of diverse terrains based on [57], to train the policy's terrain adaptivity and risk awareness. As shown in Fig. 4, the terrain set comprises: pyramid-steps with step height $h_{\text{step}}^+ \sim \mathcal{U}[0.05, 0.15]$ and step width $w_{\text{step}} = 0.3$; inverted pyramid-steps with height $h_{\text{step}}^- \sim \mathcal{U}[0.03, 0.15]$ and the same width 0.3; pyramid-slopes with gradient $s^+ \sim \mathcal{U}[0.02, 0.15]$; inverted pyramid-slopes with gradient $s^- \sim \mathcal{U}[0.02, 0.15]$; stochastic roughness modeled as zero-mean uniform elevation noise $\eta \sim \mathcal{U}[-\varepsilon, \varepsilon]$ with amplitude $\varepsilon \sim \mathcal{U}[0.00, 0.01]$; and pit terrains with depth $d_{\text{pit}} \sim \mathcal{U}[0.01, 0.25]$. All the lengths are in meters, and slopes are dimensionless gradients. To better expose the policy to risk, the upper bounds of the step-height and pit-depth ranges are set to the Unitree Go1's nominal gait limits [58], i.e., 0.15 m for step height and 0.25 m for pit depth.

3) Domain Randomization: We employ domain randomization to facilitate sim2sim and sim2real transfer. For each episode, we inject independent zero-mean uniform noise to the signals: joint position noise $\eta_q \sim \mathcal{U}[-0.01, 0.01]$ rad; joint velocity noise $\eta_{\dot{q}} \sim \mathcal{U}[-1.5, 1.5]$ rad/s; base angular-velocity noise $\eta_\omega \sim \mathcal{U}[-0.2, 0.2]$ rad/s; projected-gravity components noise $\eta_g \sim \mathcal{U}[-0.05, 0.05]$; ray-caster height noise $\eta_h \sim \mathcal{U}[-0.01, 0.01]$ m; and an added base-mass perturbation $\Delta m \sim \mathcal{U}[-1.0, 1.0]$ kg.

At each episode reset, the initial state is randomized as follows: base position offsets $\Delta p_{x,y} \sim \mathcal{U}[-0.4, 0.4]$ m; yaw angle $\psi_0 \sim \mathcal{U}[-\pi, \pi]$; base linear velocity $v^{xy} \sim \mathcal{U}[-0.2, 0.2]$ m/s; and base yaw rate $\dot{\psi}_z \sim \mathcal{U}[-0.2, 0.2]$ rad/s.

4) Curriculum: We employ a terrain difficulty curriculum to prevent early training from being blocked by overly challenging terrain. Following [59], the workspace is partitioned into tiles arranged by difficulty levels. At each reset, we assess progress on the current tile; if the agent's planar displacement since episode start exceeds half of the tile length, it is promoted to the next level. However, since our terrain set includes highly risky configurations, we introduce a downgrade buffer to maintain exposure to risk: when the velocity command exceeds the designed threshold but the agent does not traverse half a tile before reset, we record a failure; after three such failures, the agent is demoted by one level. The failure counter is cleared upon promotion or a successful episode.

V. Risk Values

Although the task-level policy in the proposed framework affords predictive safety, it neither enables online adaptation of its risk preference nor provides formal guarantees of passive safety. We cast risk evaluation in the RA formalism. Building on data-driven RA value-function learning in the existing works [60], we train a risk network via self-supervised learning. In contrast to approaches that approximate a global RA value function [51], our method follows the policy-conditioned paradigm [27], i.e., we approximate the value function induced by a fixed task-level policy. Since the task-level policy admits a tunable conservative–aggressive trade-off and the labels must be invariant to the data-collection process during self-supervision, we therefore collect data under the risk-neutral policy by fixing the risk preference parameter to $\beta = 0$. This restriction of the sampling space does not compromise the risk network's capacity to generalize over the risk-preference parameter β, because data gathered under a risk-neutral policy probe the environment's intrinsic risk characteristic. By contrast, overly conservative or overly aggressive policies skew visitation and confound estimation, introducing policy-dependent biases into the inferred risk.

In this section, we present the training details of the risk network, including the network configuration, the labeling and training process, and the training setup.

A. Risk Network

We set the risk network as a two-layer fully connected multilayer perceptron with hidden width $[64, 64]$. Following the works in [61], [62], we provide a low-dimensional vector composed of most important observation features as the risk network's observation space: the base linear velocity $\mathbf{v}$, the base angular velocity $\boldsymbol{\omega}$, the gravity projection $\mathbf{g}$ expressed in the body frame, the commanded body-frame twist $\mathbf{c}_t^{\text{body}}$, and a 12-dimensional set of ray-caster height samples obtained by reducing the original 187-dimensional height profile. Formally, the network input is the concatenation

$$\mathbf{o}_t^{risk} = \left[\mathbf{v}_t;\ \boldsymbol{\omega}_t;\ \mathbf{g}_t;\ \mathbf{c}_t^{\text{body}};\ \mathbf{h}_t^*\right],$$

where $\mathbf{h}_t^* \in \mathbb{R}^{12}$ denotes the dimension-reduced height features, and the principles underlying the dimensionality reduction of the ray-caster height samples are illustrated in Fig. 6.

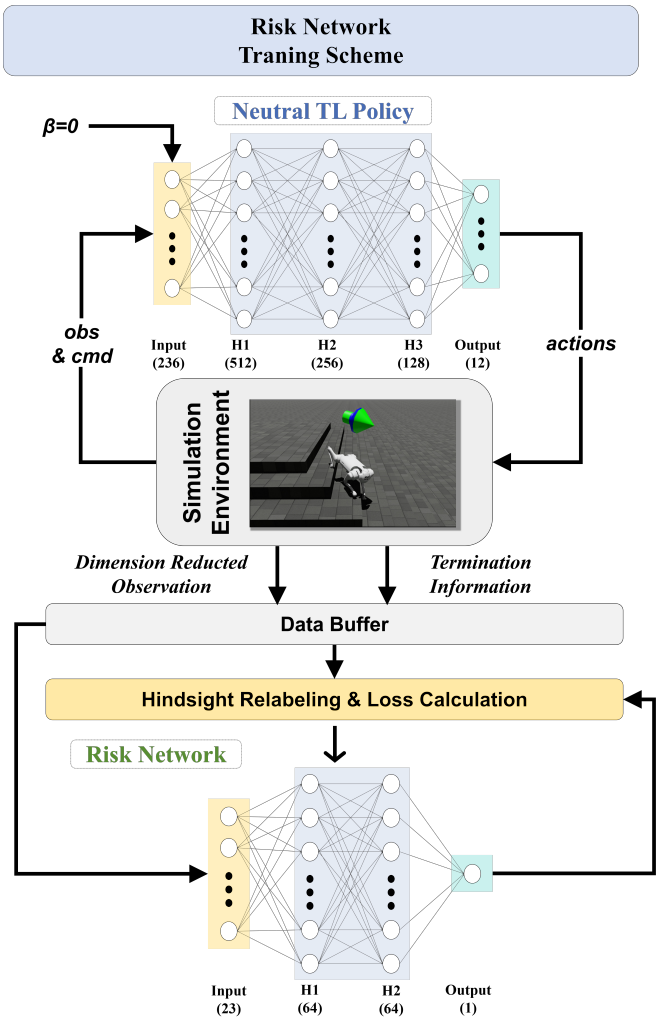

Fig. 5. Training framework of the risk network. The Neutral TL Policy is obtained from the task-level training described in Section IV and is used to drive the robot's motion in the environment. The robot states are processed through the simulator and converted into dimension-reduced observations and termination information, which are stored in the data buffer. The collected data are then relabeled in hindsight and used for loss calculation, producing training labels for supervising the risk network.

The risk network outputs a single scalar-valued risk score $r_t \in [-1, 1]$.

B. Labeling and Training

For notational convenience, we denote the risk value by $V_{\text{risk}}^{\Pi_{\text{neutral}}}\left(\mathbf{o}_t^{\text{risk}}\right)$, and the corresponding training label by $\widehat{V}\left(\mathbf{o}_t^{\text{risk}}\right)$.

Consistent with Section III-A.6, we define the training label for the risk network as:

$$\widehat{V}^{\text{target}}\left(\mathbf{o}_t^{\text{risk}}\right) = \gamma_{\text{risk}} \max\left\{\zeta(s_t), \min\left\{\ell(s_t), \widehat{V}^{\text{old}}\left(o_{t+1}^{\text{risk}}\right)\right\}\right\} + \left(1-\gamma_{\text{risk}}\right) \max\left\{\ell(s_t), \zeta(s_t)\right\}, \quad (47)$$

where we set the discount $\gamma_{\text{risk}} = 0.999999$, following [27], a large discount yields a tighter approximation to the true

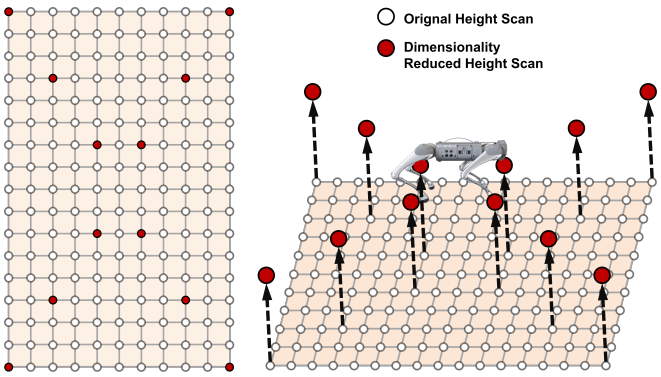

Fig. 6. Dimension-reduced height scan used in the risk network, where the white circles represent the original 17×11 height scan and the red circles denote the extracted 12-dimensional features.

value. The term $\widehat{V}^{\text{old}}(\cdot)$ is computed by the snapshot of the risk network from the previous training iteration and serves as the bootstrap target.

As described in Section III-A.5, both the task function $\ell(s_t)$ and the risk function $\zeta(s_t)$ are required to be Lipschitz continuous. We first instantiate the task shaping function as:

$$\ell(s_t) \;=\; \tanh\Big(\log\big(\tfrac{e_t}{\tau}+\varepsilon\big)\Big), \tag{48}$$

where $e_t = \big\|\mathbf{v}_t^b - \mathbf{u}_t^b\big\|_2$ denotes the body-frame velocity-tracking error in xy plane, $\tau = 0.1$ m/s is the velocity tolerance, and $\varepsilon = 10^{-8}$ is a small regularizer to avoid numerical error.

For the risk function, we set over-tilt, undesired collision, and foot stumble as risky events and construct a hindsight label that increases linearly over the $N = 10$ frames preceding the event, as:

$$\zeta^*(s_t) \;=\; \frac{1}{N}\big(t-(\tau-N)\big)_+ \mathbf{1}\{t \le \tau\}, \tag{49}$$

where τ denotes the risk event time, $(x)_+ \triangleq \max\{0,x\}$ denotes the positive part operator, and $\mathbf{1}\{\cdot\}$ is the indicator function. And it was map to $[-1,1]$ by

$$\zeta(s_t) = 2\,\zeta^*(s_t) - 1. \tag{50}$$

The risk network is trained to satisfy

$$V_{\text{risk}}^{\Pi_{\text{neutral}}}\big(\mathbf{o}_t^{\text{risk}}\big) \;\approx\; \widehat{V}\big(\mathbf{o}_t^{\text{risk}}\big). \tag{51}$$

Since the agents are safe at most times in the rollout, we adopt an importance-weighted mean-squared error (MSE) loss following [63], [64]:

$$L \;=\; \lambda\,\frac{1}{T}\sum_{t=1}^{T} w_t\left(V_{\text{risk}}^{\Pi_{\text{neutral}}}\big(\mathbf{o}_t^{\text{risk}}\big) - \widehat{V}\big(\mathbf{o}_t^{\text{risk}}\big)\right)^2, \tag{52}$$

with per-sample weights

$$w_t \;=\; \begin{cases} \dfrac{\alpha}{\hat{p}_{\text{risk}}+\varepsilon}, & \text{if sample } t \text{ is } \textit{risk-positive} \\[2ex] \dfrac{1-\alpha}{\hat{p}_{\text{safe}}+\varepsilon}, & \text{if sample } t \text{ is } \textit{safe} \end{cases}, \tag{53}$$

where T is the mini-batch size; $\widehat{V}(\cdot)$ is the supervisory target defined in Eq. 47; *risk-positive* denotes samples labeled positive by the risk shaping signal $\zeta(s_t)$ and *safe* otherwise; $\hat{p}_{\text{risk}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{1}\{\text{risk-positive at } t\}$ is the empirical risk-positive fraction in the batch; $\alpha \in (0,1)$ sets the desired class contribution (we use $\alpha = 0.5$ for balance), $\varepsilon = 10^{-8}$ prevents division by zero, and $\lambda = 100$ scales gradients without affecting the minimizer.

C. Training Setup

We trained on the laptop configuration described in Section IV-E, using 1024 parallel agents to collect trajectories. The agents are operated in the designed environment; however, unlike task-level policy training, we *uniformly* assigned agents across all terrain tiles at reset and disabled the terrain curriculum. During rollout, every 20 simulation steps, we performed an offline pass over a rolling per-environment buffer of the most recent 1000 transitions, with mini-batches of size 200 for each agent. The entire run comprised 200,000 steps and took approximately 4.5 hours of wall-clock time on our setup.

VI. Recovery Policy

The recovery policy is a safety-prioritized policy for tracking body-frame twist commands and is trained using a standard PPO framework. This section presents the training details, including the actor and critic, the reward design, and the training setup.

A. Actor and Critic

We use a three-layer fully connected MLP with hidden widths $[512, 256, 128]$ for both actor and critic. The observation vector is defined as: base linear velocity $\boldsymbol{v}$, base angular velocity $\boldsymbol{\omega}$, the gravity projection $\boldsymbol{g}$ expressed in the body frame, the body-frame twist command $\boldsymbol{c}_t^{\text{body}}$, the joint positions $\boldsymbol{q}$, the joint velocities $\dot{\boldsymbol{q}}$, the previous action $\boldsymbol{a}_{t-1}$, and the ray-caster height samples $\boldsymbol{h}_t$ which is the same with task-level policy. Formally, the observation can be summarized as:

$$\boldsymbol{o}_t^{\text{Recovery}} = \big[\boldsymbol{v}_t;\ \boldsymbol{\omega}_t;\ \boldsymbol{g}_t;\ \boldsymbol{c}_t^{\text{body}};\ \boldsymbol{q}_t;\ \dot{\boldsymbol{q}}_t;\ \boldsymbol{a}_{t-1};\ \boldsymbol{h}_t\big]. \tag{54}$$

The actor outputs the policy action as a 12-dimensional joint target vector, which is tracked by the PD controller in Eq.21. The critic outputs a scalar-valued estimate of the return.

B. Rewards

Consistent with the preceding design, the recovery policy's reward function is also decomposed into task, penalty, and regularization components:

$$r_{\text{total}} = r_{\text{penalty}}\cdot\Delta t + r_{\text{task}}\cdot\Delta t + r_{\text{regularization}}\cdot\Delta t, \tag{55}$$

In designing the recovery policy's reward, we follow [27] and [65], assigning increased weight to components that promote balance and safety so as to prioritize stability while preserving command-tracking performance.

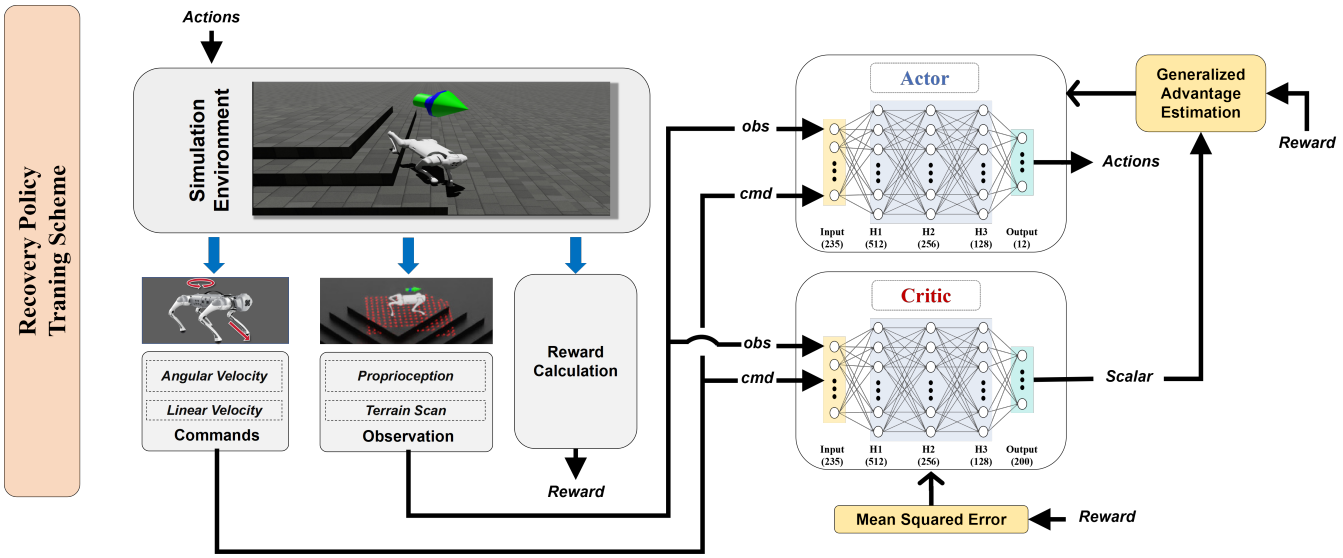

Fig. 7. Recovery policy training framework based on standard PPO. The policy is optimized through simulator interactions and reward signals, with generalized advantage estimation used to update the actor–critic networks.

1) Penalty: We penalize robot collisions:

$$r_{\text{penalty}} = -3.0 \cdot r_{\text{collication}}, \tag{56}$$

where $r_{\text{collication}}$ is shown in Eq. 32.

2) Task: Within the task-reward channel, we assign a higher weight to the *alive* term and introduce two additional shaping signals—base orientation and target posture—to explicitly encourage balance:

$$\begin{aligned} r_{\text{task}} = 1.5 \cdot r_{\text{lin\_xy}} + 0.75 \cdot r_{\text{yaw\_z}} + 0.75 \cdot r_{\text{alive}} \\ + 0.05 \cdot r_{\text{orientation}} - 0.02 \cdot r_{\text{posture}}, \end{aligned} \tag{57}$$

where $r_{\text{lin\_xy}}$, $r_{\text{yaw\_z}}$, and r_{alive} are designed, respectively, in Eq. 36, Eq. 37, and Eq. 38.

The base orientation term is

$$r_{\text{orientation}} = \boldsymbol{g}^2, \tag{58}$$

The target posture term is aimed at tracking the normal standing posture:

$$r_{\text{posture}} = \sum_{i=1}^{12} \left| q_i - \bar{q}_{\text{rec},i} \right|, \tag{59}$$

where $q = [q_1, \ldots, q_{12}]^\top \in \mathbb{R}^{12}$ denotes the joint-angle vector, and $\bar{q}_{\text{rec}} = [\bar{q}_{\text{rec},1}, \ldots, \bar{q}_{\text{rec},12}]^\top$ is the normal stand posture.

3) regularization: The regularization rewards for the recovery policy are designed as:

$$\begin{aligned} r_{\text{regularization}} = &-2.0 \cdot r_{\text{lin\_z}} - 0.01 \cdot r_{\text{ang\_xy}} - 1.5e^{-4} \cdot r_{\text{torques}} \\ &- 5.0e^{-8} \cdot r_{\text{dof\_acc}} + 0.5 \cdot r_{\text{feet\_air\_time}} \\ &- 0.01 \cdot r_{\text{action\_rate}}, \end{aligned} \tag{60}$$

where all reward terms are defined in Section IV-D.3.

C. Training Setup

The training setup closely follows Section IV-E, with the following deviations:

1) Simulator and Hardware: Owing to the reduced GPU–memory footprint of the standard PPO pipeline, we are able to employ 4096 parallel agents.

2) Domain Randomization: We augment domain randomization with stricter initialization perturbations, added random base rotational offset $\Delta\theta_{x,y} \sim \mathcal{U}[-0.5, 0.5]$ rad; set base linear velocity $v^{xy} \sim \mathcal{U}[-0.5, 0.5]$ m/s, and base yaw rate $\dot{\psi}_z \sim \mathcal{U}[-0.4, 0.4]$ rad/s, to emulate the robot's state after a policy switch.

3) Curriculum: We lowered the promotion threshold in the curriculum: at the end of each episode, the agent is promoted if its planar displacement exceeds one-third of a terrain tile's side length. We also removed the downgrade buffer, so that a single failure can immediately trigger demotion.

VII. Integration of Modules

A. Recovery Command Generation

The recovery command is computed by scanning the risk network over candidate planar headings in the body frame and selecting the least–risky direction. N candidate headings are sampled uniformly on $[0, 2\pi)$:

$$\theta_k = \frac{2\pi k}{N}, \qquad \mathbf{d}_k = \left[\cos\theta_k, \sin\theta_k\right], \quad k = 0, \ldots, N-1. \tag{61}$$

Let $\mathbf{o}_t^{risk}$ denote the current risk network observation vector. For each candidate k, a temporary observation $\mathbf{o}_{t\_k}^{risk}$ is built by cloning $\mathbf{o}_t^{risk}$ and forcing the base linear velocity and commanded body-fame twist channel to the candidate heading (i.e., assuming perfect tracking for all direction), and evaluate the scalar risk value

$$r_k = \mathcal{R}\left(\mathbf{s}_t^{(k)}\right). \tag{62}$$

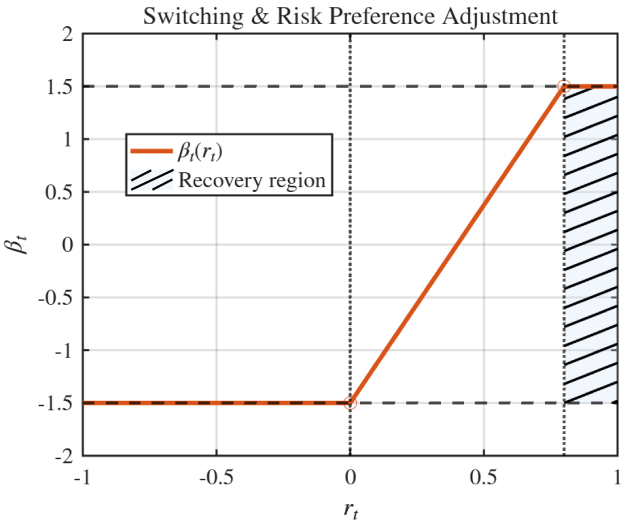

Fig. 8. Switching and risk-preference adjustment. $\beta_t(r_t)$ follows Eq. (64) with threshold $\tau_{\text{sw}} = 0.8$; the hatched light-blue area indicates the recovery region.

The recovery command $\mathbf{v}_t^{\text{recovery}}$ is then selected by choosing the safest heading and forming a fixed–magnitude command

$$k^\star = \arg\min_k r_k, \qquad \mathbf{v}_t^{\text{recovery}} = v_0\,\mathbf{d}_{k^\star}, \quad v_0 = 0.5 \text{ m/s}. \tag{63}$$

$\mathbf{v}_t^{\text{recovery}}$ is written into the command channel $\boldsymbol{c}_t^{\text{body}}$ of the recovery observation and passed to the policy.

This procedure runs every control cycle at 50 Hz.

B. Switching and Risk Preference Adjustment

As described in Section III-B, we also use the risk value $V_{\text{risk}}^{\pi}(s_t)$ for policy switching and risk preference adjustment. We adopt a deterministic gating rule with the threshold of τ_{sw} in Eq. 19, and set $\tau_{\text{sw}} = 0.8$.

When the task policy is active, the risk preference parameter β is modulated by a piecewise-linear map:

$$\beta_t = \begin{cases} -1.5, & r_t \le 0, \\ -1.5 + \dfrac{3}{0.8}\, r_t, & 0 < r_t \le 0.8, \end{cases} \tag{64}$$

which varies monotonically from a aggressive setting ($\beta = -1.5$) to an conservative setting ($\beta = 1.5$) as $V_{\text{risk}}^{\pi}(s_t)$ increases on $[0, 0.8]$, while clamping to -1.5 for $r_t \le 0$. The switching mechanism and risk-preference adjustment are shown in Fig. 8.

VIII. Experiments

A. Task-level Policy

1) Critic Evaluation: To characterize the distributional critic in DPPO under varying risk preferences, and to elucidate how the task-level training framework yields risk-aware policies, we conduct evaluations on six representative terrain settings: rough plane (a), pit (b), step (c), inverted step (d), slope (e), and inverted slope (f). In each setting, an aggressive policy is employed to explore the environment, from which the critic's output distributions are collected. Subsequently, the Wang transform is applied to obtain risk-distorted distributions. The results are shown in Fig. 9.

Across terrain configurations and risk preferences, the critic outputs exhibit consistent and interpretable patterns. On relatively benign terrains such as the rough plane and the slopes, the critic produces distributions with higher means and smaller variances, indicating that the agent can achieve both larger and more stable returns in low-risk environments. Moreover, in these mild settings, the discrepancy between conservative and aggressive distortions remains small, suggesting limited uncertainty and constrained tail risks as well as potential extreme rewards.

In contrast, in more challenging terrains such as the pit and the step/inverted step, the original distributions are broader and exhibit lower means. Notably, the step setting attains a relatively higher mean compared to the pit and inverted step, as the agent does not need to overcome gravity and can more easily track velocity commands, thereby achieving higher expected returns. Overall, these complex terrains result in lower expected values and substantially increased uncertainty relative to mild configurations. After applying the Wang transform, the conservative distortion consistently shifts toward higher expected returns, whereas the aggressive distortion shifts toward lower expected returns, producing pronounced differences. It is worth noting that, due to the GAE calculation in the actor–critic framework illustrated by Eq. 24, the relationship between critic distributional shifts and policy aggressiveness may appear counterintuitive.

In summary, these findings demonstrate that the distributional critic effectively captures scenario-dependent uncertainty structures, while the Wang transform provides a unified and tunable mechanism that explains the emergence of risk-aware characteristics in the training framework. Overall, the results validate that replacing the conventional scalar critic with QR-DQN enables the learning of return distributions; by adjusting the parameter β, one can systematically reweight uncertainty and thereby induce differentiated risk-sensitive policy learning.

2) Policy Evaluation: We evaluate the performance of policies with different risk preferences across multiple terrain configurations. For each terrain–preference setup, we execute 40 independent trials under a fixed forward velocity command of $1\,\text{m/s}$. The results are summarized in Table II. Outcomes are categorized into three classes: *Avoid*, *Fail*, and *Succ*, where *Avoid* denotes that the robot halts at the terrain boundary without attempting to traverse; *Fail* denotes that the robot attempts to traverse but does not succeed (e.g., falls over or gets stuck); *Succ* indicates that the robot enters the terrain region and successfully exits it.

On the flat terrain, all risk preferences achieve the 100% success rate, indicating that environmental risk is negligible and risk sensitivity has virtually no impact on performance. In the Step_inv and Pit settings, shaded in red and green, success rates decrease markedly with increasing terrain difficulty across all risk preferences, and a clear separation emerges between conservative and aggressive policies. Conservative policies ($\beta > 0$) tend to avoid high-risk states, with the

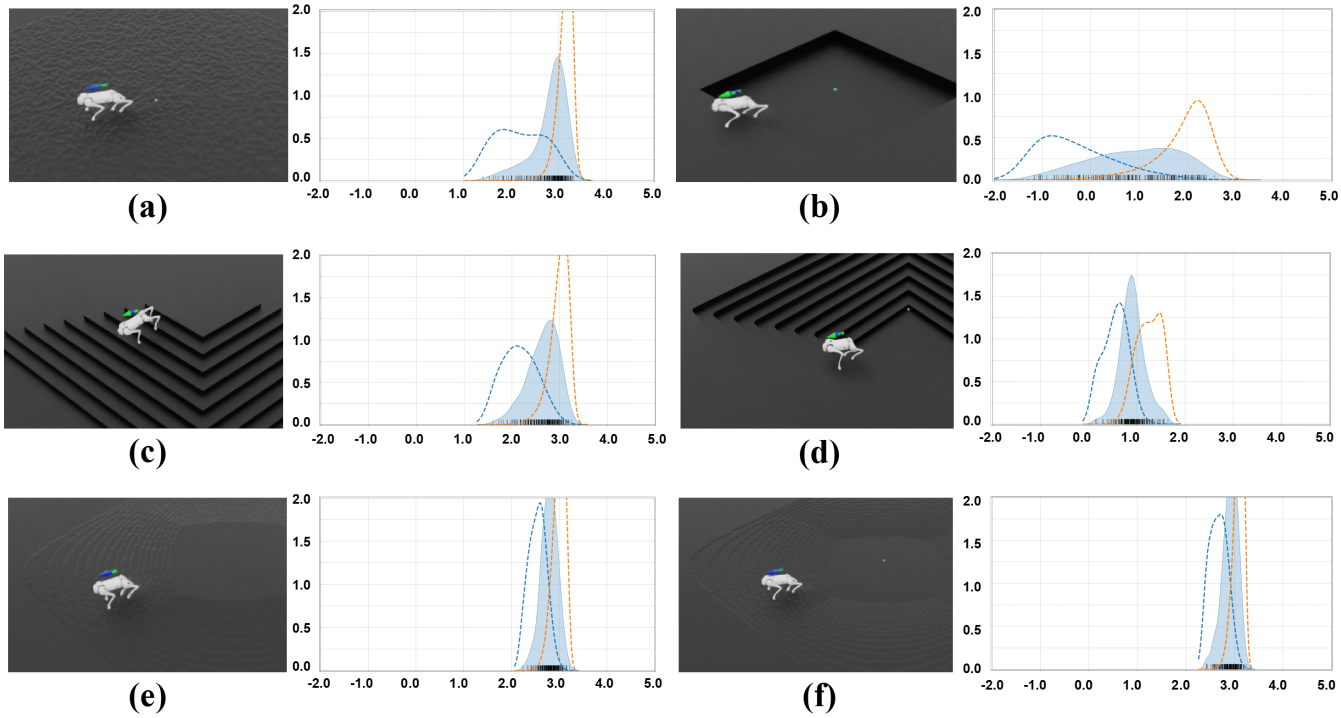

Fig. 9. Visualization of the critic output distributions, together with the distorted distributions distorted by the Wang metric, where the conservative setting corresponds to $\beta = 1.5$ (orange) and the aggressive setting corresponds to $\beta = -1.5$ (blue). Subfigures illustrate different terrain settings: (a) rough plane, (b) pit, (c) step, (d) step_inv, (e) slope, and (f) slop_inv.

TABLE II

PERFORMANCE OF THE TASK-LEVEL POLICY ACROSS DIFFERENT TERRAINS AND RISK PREFERENCES. CELL BACKGROUND COLORS INDICATE AGGRESSIVENESS/CONSISTENCY WITH RISK PREFERENCE (RED/GREEN) AND DEVIATIONS FROM THE TREND (YELLOW).

Step_inv										**Pit**									
	5cm			8cm			12cm				15cm			20cm			25cm		
β	Avoid	Fail	Succ	Avoid	Fail	Succ	Avoid	Fail	Succ	β	Avoid	Fail	Succ	Avoid	Fail	Succ	Avoid	Fail	Succ
1.5	0	0	100	95	0	5	100	0	0	1.5	0	2.5	97.5	90	5	5	100	0	0
0.8	0	0	100	55	0	45	97.5	2.5	0	0.8	0	2.5	97.5	5	42.5	52.5	97.5	0	2.5
0	0	0	100	7.5	2.5	90	75	25	0	0	0	7.5	92.5	0	20	80	40	17.5	42.5
-0.8	0	0	100	2.5	7.5	90	50	10	10	-0.8	0	5	95	0	5	95	5	35	60
-1.5	0	0	100	0	15	85	5	37.5	57.5	-1.5	0	7.5	92.5	0	15	85	0	32.5	67.5
Step										**Flat**									
	5cm			8cm			12cm				———								
β	Avoid	Fail	Succ	Avoid	Fail	Succ	Avoid	Fail	Succ	β	Avoid			Fail			Succ		
1.5	0	5	95	0	12.5	87.5	0	80	20	1.5	0			0			100		
0.8	0	10	90	0	10	90	0	65	35	0.8	0			0			100		
0	0	7.5	92.5	0	15	85	0	65	35	0	0			0			100		
-0.8	0	12.5	87.5	0	17.5	82.5	0	45	55	-0.8	0			0			100		
-1.5	0	7.5	92.5	0	7.5	92.5	0	40	60	-1.5	0			0			100		

proportion of *Avoid* increasing as difficulty rises; aggressive policies ($\beta < 0$), however, attempt traversal more frequently and, while achieving higher success rates than conservative policies, also incur higher failure rates.

The Step setting departs from this trend, as marked in yellow. Even at higher difficulties, all risk preferences exhibit elevated failure rates, yet the conservative policy rarely produces *Avoid* outcomes. The failure mechanism here is dominated by rollovers: as the robot moves from the step top to the bottom, excessive pitch induces a rollover that triggers an immediate reset during training. Consequently, trajectories are truncated upon failure onset and cannot continue to accrue terms such as collision or feet-stumble rewards; the expected-return distribution is thus shifted globally toward lower values rather than developing heavier adverse tails, which attenuates the observable risk-aware signature.

Overall, task-level policies exhibit differentiated behaviors under distinct risk preferences: conservative policies prioritize safety at the expense of success rate, whereas aggressive policies pursue higher success at the cost of elevated failure risk. By tuning the risk parameter β, one can smoothly trade off performance and safety.

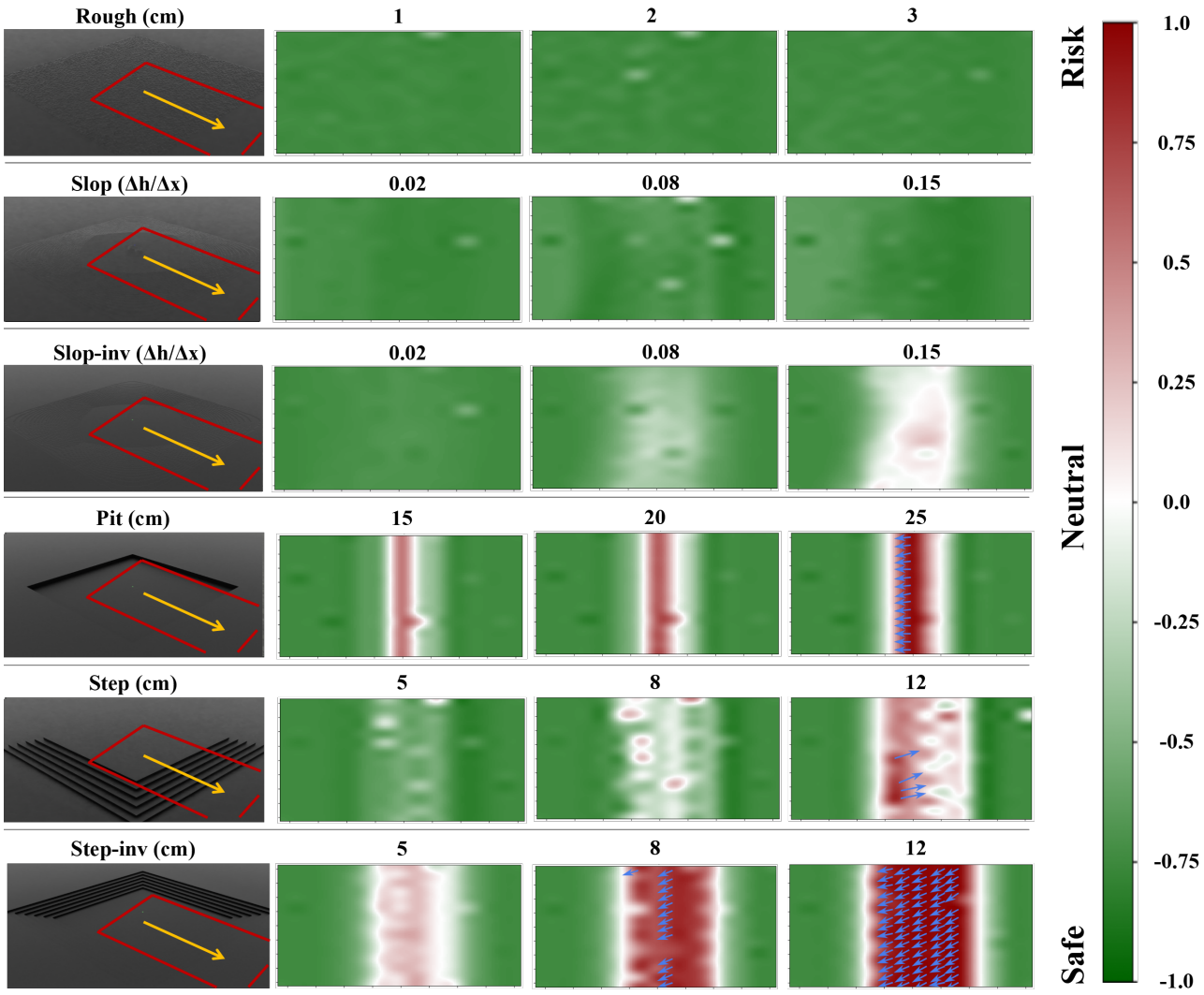

Fig. 10. Heatmap visualization of risk network predictions under varying terrain difficulties. The robot is initialized within the red box and commanded to explore the terrain at a forward velocity of 1 m/s (along the yellow arrow). The risk network computes risk values in real time within the range $[-1, 1]$, where -1 (green) denotes safe regions and 1 (red) denotes risky regions, indicating the estimated risk of the configuration. The blue arrows represent the recovery commands generated by the risk network, as described in Section VII.

B. Risk Network

We evaluated the risk network as follows: the robot was assigned to regions with different terrain types and difficulty levels to collect data. The collected data were then processed offline by the risk network, and the outputs were visualized as heatmaps shown in Fig. 10. Furthermore, whenever the network output exceeded the threshold τ_{sw} defined in Section VII, the corresponding recovery command was added as blue arrows at those locations to indicate the motion direction received by the recovery policy.

In low–difficulty configurations (i.e., rough plane, slope, slope_inv), the predictions generally form a uniform low–risk background, with near–neutral values appearing only under a few locally harsher conditions; the safe region remains broad and continuous. In contrast, in high–difficulty configurations (i.e., pit, step, step_inv), the spatial risk distribution is highly consistent with the terrain geometry: high values first emerge in the central area of the terrain features; as the difficulty increases, both the peak magnitude and the spatial proportion of high–risk bands increase, gradually expanding toward the terrain boundaries, and at the highest difficulty nearly covering the entire terrain extent. Moreover, in pit, step, and step_inv at high difficulty, as well as in step_inv at medium difficulty, the risk values exceed the threshold; the associated recovery commands point away from the hazardous regions, namely in the directions of moving down into the pit or descending the steps. This is consistent with empirical observations, where leveraging gravity results in safer motions than opposing it.

In summary, the outputs of the risk network are consistent with intuition: as the terrain becomes more challenging, the risk set expands and the corresponding risk values increase; at the same time, the safety–recovery directions indicated by

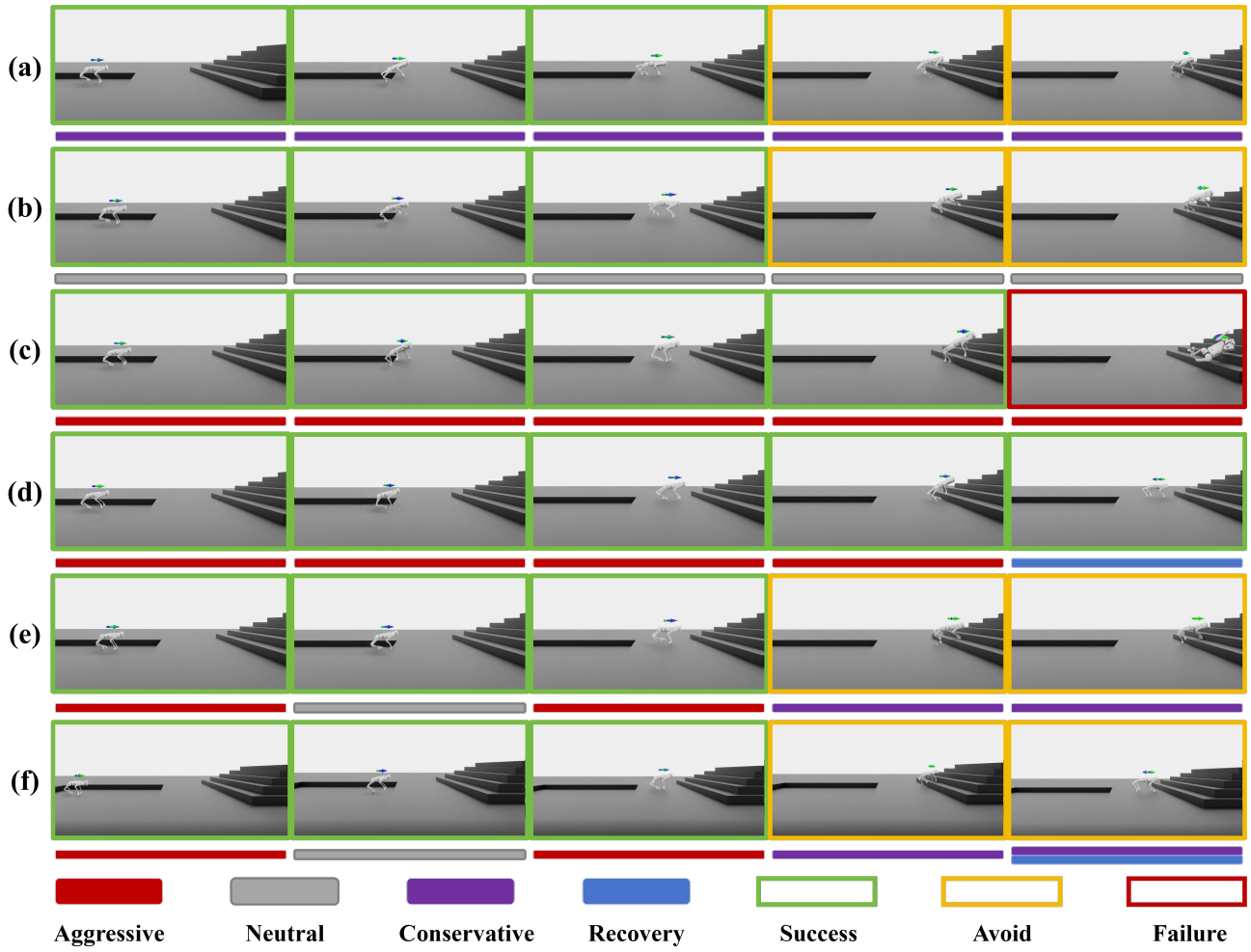

Fig. 11. Comparative rollouts under five setups: (a) conservative policy; (b) neutral policy; (c) aggressive policy; (d) aggressive policy with recovery policy; (e) adaptive-β policy; and (f) adaptive-β policy with recovery policy. Each row shows one complete trail, with frame colors indicating the state condition—success (green), avoid (yellow), and failure (red). The colored bars beneath each state denote the policy type: aggressive (red), neutral (gray), conservative (purple), and recovery (blue).

TABLE III
SUCCESS RATE OF RECOVERY POLICY INTERVENTION

	Baseline	**Recovery Policy**
Success Rate (%)	57.5	**95.0**
Settling Time (S)	0.7624	**0.3888**

the risk network are correct and consistent.

C. Recovery Policy

To ensure fairness, we trained the baseline policy using the same training framework as the recovery policy, with the reward and training configurations specified in [66]. On this basis, we designed a randomized experiment to evaluate the intervention success rate and average settling time of the policy. Specifically, at the beginning of each trial, the robot was randomly assigned a wide range of initial heights, initial pitch and roll angles, as well as linear and angular velocities, in order to simulate the diverse states that may occur when the switching mechanism of the overall framework is triggered. For both the baseline and the recovery policy, a forward velocity command of $1\,\mathrm{m/s}$ was assigned, and 40 independent trials were conducted for each policy. The recovery success rate and the average settling time (time to reach target velocity within 0.1 m/s) were recorded. The experimental results are shown in Table III.

The results show that the baseline policy achieved a success rate of 57.5%, with an average settling time of 0.7624 s in successful cases. Under the same conditions, however, the recovery policy significantly improved the success rate to 95.0% and reduced the average settling time to 0.3888 s. Failures occurred only in extreme cases, where both the rolling angle and the initial height simultaneously reached the limits of the recovery policy, leading to agent collapse. These findings indicate that adopting a safety-prioritized recovery policy within the proposed policy-switching mechanism can substantially increase the intervention success rate and enable rapid tracking of the recovery command, thereby ensuring the effectiveness of UPPS-RL in achieving passive safety and providing reliable operation of robots in hazardous

environments.

D. Holistic System Experiment

1) Qualitative Evaluation: To evaluate the effectiveness of the proposed UPPS-RL framework in passive safety and predictive safety, we conducted comparative experiments on the five configurations shown in Fig. 11. Each agent was initialized on flat ground at the left side of the mosaic tile and was commanded to track a forward speed of $1\,\mathrm{m/s}$. The test environment first contained a 15 cm pit (low difficulty), followed by a step_inv terrain with a step height of 12 cm (high difficulty). We let each trial last 20 s, and no reset logic was configured so as to observe the full policy execution. A trial was labeled *success* if the velocity command was tracked; *avoid* if standstill actions were taken before or at the early stage of entering the risk region; and *failure* in the case of severe collision, falling, or getting stuck. For each trial—especially (d), (e), and (f)—the policy mode and risk preference were recorded to analyze the control strategy across states.

The results of (a), (b), and (c) are consistent with the quantitative findings in Section VIII-A: in low risk regions, the three risk preferences exhibit similar control performance and all track the speed command successfully; in high risk regions, the conservative and neutral policies take *avoid* actions, ensuring safety at the expense of command tracking. Moreover, the conservative policy tends to stop at the geometric boundary of the risky terrain, whereas the neutral policy proceeds into the risk region. The aggressive risk preference prioritizes command tracking but has a higher probability of failure within the risk region. With the recovery mechanism, upon entering the risk region, the aggressive policy triggers recovery, and the command guides the agent back to a safe region (i.e., moving down off the steps in this experiment setup), effectively suppressing catastrophic failures. In addition, the combined adaptive-β policy with recovery exhibits more flexible risk-aware control: remaining aggressive in safe regions to achieve a higher success rate, while proactively lowering the risk preference before approaching the risk region to increase safety. In summary, the results indicate that adjustable risk preference based on risk assessment, together with recovery control, effectively improves safety.

2) Quantitative Evaluation: As a complement, we conducted additional tests following the same methodology as in Section VIII-A.2. Specifically, we evaluated the *adaptive-β policy* (a), the *adaptive-β policy with baseline recovery policy* (a+r), and the *adaptive-β policy with the proposed recovery policy* (a+r*). In this setting, an additional outcome class, *Recovery*, is introduced to represent cases where the recovery mechanism is triggered and successfully guides the robot back to the safe region. The results are shown in Table IV.

The comparison with Table II shows that the adaptive-β policy, by automatically adjusting risk preference, enables the robot to maintain a low failure rate across terrains of different difficulty levels: adopting a neutral strategy on medium-difficulty terrains and a conservative strategy on high-difficulty terrains, thereby achieving a balance between success and safety. Furthermore, the results of a+r and a+r* indicate that the proposed recovery policy can significantly reduce the failure rate, whereas the baseline recovery policy may even increase failures in some cases due to aggressive maneuvers during policy switching. These findings demonstrate that combining the policy-switching mechanism with the proposed recovery policy can effectively enhance the safety of the robot.

IX. Conclusion

This work presents the UPPS-RL framework, an RL-based locomotion control method that incorporates both passive safety and predictive safety. It comprises three complementary modules: (*i*) a risk-aware task-level policy built upon a distributional critic, in which the value distribution is distorted by the Wang distortion risk measure to realize a spectrum of policies—adjustable via the risk preference parameter β—from conservative to aggressive; (*ii*) a risk network trained offline in a self-supervised learning method from trajectories generated by the neutral policy ($\beta=0$), self-labeled using velocity-tracking performance and termination events, to provide terrain–action–conditioned risk estimates; and (*iii*) a recovery policy triggered by the predicted risk, which given safety-prioritized control under high-risk conditions. This architecture integrates passive and anticipatory safety within a unified control pipeline while avoiding the introduction of constraints that could limit exploration during training.

Systematic evaluations and ablations across different scenarios—step/step_inv, pit, slope, and rough plane—demonstrate that the recovery mechanism suppresses catastrophic failures; a fixed aggressive policy attains stronger speed-tracking capability but incurs elevated failure risk; fixed conservative/neutral policies tend to avoid in high-risk regions at the expense of reachability; and an adaptive-β policy combined with recovery lowers the risk preference before, or at the early stage of, entering risky regions, thereby yielding a superior safety–success trade-off in multi-task difficulty settings. Visual analyses of the distributional critic, risk heatmaps responsive to terrain difficulty, together with quantitative statistics, further explain and substantiate the effectiveness of UPPS-RL.

Acknowledgement

First and foremost, I would like to express my sincere appreciation to my supervisors, Dr. C. (Cosimo) Della Santina and Dr. J. (Jiatao) Ding. Throughout the course of my research, they have provided me with invaluable guidance and patient instruction. Their generous support in shaping my academic thinking and research methodology has enabled me to continuously improve and grow.

I would also like to extend my heartfelt thanks to my parents for their constant understanding, encouragement, and support, which allowed me to devote myself to my studies and research without distraction.

TABLE IV

PERFORMANCE OF THE OVERALL FRAMEWORK ACROSS DIFFERENT TERRAINS. CELL BACKGROUND COLORS HIGHLIGHT THE COMPARISON OF FAILURE RATES.

Step_inv													**Pit**											
	5cm				8cm				12cm					15cm				20cm				25cm			
Algo	Av	Rec	Fail	Suc	Av	Rec	Fail	Suc	Av	Rec	Fail	Suc	Algo	Av	Rec	Fail	Suc	Av	Rec	Fail	Suc	Av	Rec	Fail	Suc
a	0	0	0	100	20	0	0	80	97.5	0	2.5	0	a	5	0	2.5	92.5	20	0	10	70	100	0	0	0
a+r	0	0	0	100	0	25	0	75	0	82.5	17.5	0	a+r	0	17.5	0	82.5	5	32.5	0	62.5	15	82.5	2.5	0
a+r*	0	0	0	100	0	22.5	0	77.5	10	90	0	0	a+r*	0	15	0	85	0	32.5	0	67.5	2.5	92.5	0	5
Step													**Flat**											
	5cm				8cm				12cm					———											
Algo	Avoid	Recov.	Fail	Succ	Avoid	Recov.	Fail	Succ	Avoid	Recov.	Fail	Succ	Algo	Avoid		Recover		Fail		Success					
a	0	0	2.5	97.5	0	0	15	85	0	0	50	50	a	0		0		0		100					
a+r	0	0	0	100	0	0	15	85	0	0	50	50	a+r	0		0		0		100					
a+r*	0	0	0	100	0	0	10	90	0	0	37.5	62.5	a+r*	0		0		0		100					

In addition, I am deeply grateful to my colleagues and friends. Their help and companionship, both in research and in life, have filled this journey with warmth and strength.

Finally, I would like to thank the Unitree Go1 robot in our lab, which has accompanied me along the way. Although it was regrettably unable to participate in the experiments of this thesis due to damage, it provided me with valuable support and inspiration in previous projects and has been an indispensable partner in my research journey.

REFERENCES

[1] V. Atanassov, J. Ding, J. Kober, I. Havoutis, and C. Della Santina, "Curriculum-based reinforcement learning for quadrupedal jumping: A reference-free design," *IEEE Robotics & Automation Magazine*, 2024.

[2] F. Vezzi, J. Ding, A. Raffin, J. Kober, and C. Della Santina, "Two-stage learning of highly dynamic motions with rigid and articulated soft quadrupeds," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 9720–9726.

[3] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[4] R. Siegwart, M. Hutter, P. Oettershagen, M. Burri, I. Gilitschenski, E. Galceran, and J. Nieto, "Legged and flying robots for disaster response," in *World engineering conference and convention (WECC)*. ETH-Zürich, 2015.

[5] L. Amatucci, G. Turrisi, A. Bratta, V. Barasuol, and C. Semini, "Vero: A vacuum-cleaner-equipped quadruped robot for efficient litter removal," *Journal of Field Robotics*, vol. 41, no. 6, pp. 1829–1842, 2024.

[6] C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter, "Anymal in the field: Solving industrial inspection of an offshore hvdc platform with a quadrupedal robot," in *Field and service robotics: results of the 12th international conference*. Springer, 2021, pp. 247–260.

[7] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, "A predictive safety filter for learning-based racing control," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7635–7642, 2021.

[8] W. S. Cortez, C. K. Verginis, and D. V. Dimarogonas, "Safe, passive control for mechanical systems with application to physical human-robot interactions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3836–3842.

[9] S. Bouraine, T. Fraichard, and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in dynamic environments," *Autonomous Robots*, vol. 32, no. 3, pp. 267–283, 2012.

[10] J. Ding, T. L. Lam, L. Ge, J. Pang, and Y. Huang, "Safe and adaptive 3-d locomotion via constrained task-space imitation learning," *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 6, pp. 3029–3040, 2023.

[11] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5761–5768.

[12] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2436–2442.

[13] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo, "Learning quadrupedal locomotion on deformable terrain," *Science Robotics*, vol. 8, no. 74, p. eade2256, 2023.

[14] H. Kim, H. Oh, J. Park, Y. Kim, D. Youm, M. Jung, M. Lee, and J. Hwangbo, "High-speed control and navigation for quadrupedal robots on complex and discrete terrain," *Science Robotics*, vol. 10, no. 102, p. eads6192, 2025.

[15] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8352–8358.

[16] Q. Liao, Z. Li, A. Thirugnanam, J. Zeng, and K. Sreenath, "Walking in narrow spaces: Safety-critical locomotion control for quadrupedal robots with duality-based optimization," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2723–2730.

[17] W. Liu, Y. Gao, F. Gao, and S. Li, "Trajectory adaptation and safety control via control barrier functions for legged robots," in *2021 China Automation Congress (CAC)*. IEEE, 2021, pp. 5571–5576.

[18] C. Li, X. Peng, W. Lan, and X. Yu, "Autonomous and safety-critical stair climbing via nonlinear model predictive control for quadrupedal robots," in *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2023, pp. 1–6.

[19] L. Schneider, J. Frey, T. Miki, and M. Hutter, "Learning risk-aware quadrupedal locomotion using distributional reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 451–11 458.

[20] J. Shi, C. Bai, H. He, L. Han, D. Wang, B. Zhao, M. Zhao, X. Li, and X. Li, "Robust quadrupedal locomotion via risk-averse policy learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 459–11 466.

[21] Y. Zhang, R. Corcodel, and D. Zhao, "Bipedalism for quadrupedal robots: Versatile loco-manipulation through risk-adaptive reinforcement learning," *arXiv preprint arXiv:2507.20382*, 2025.

[22] H. Zhang, J. Wang, Z. Wu, Y. Wang, and D. Wang, "Terrain-aware risk-assessment-network-aided deep reinforcement learning for quadrupedal locomotion in tough terrain," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2021, pp. 4538–4545.

[23] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3642–3649, 2020.

[24] J. Lee, L. Schroth, V. Klemm, M. Bjelonic, A. Reske, and M. Hutter, "Exploring constrained reinforcement learning algorithms for quadrupedal locomotion," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 11 132–11 138.

[25] E. Chane-Sane, P.-A. Leziart, T. Flayols, O. Stasse, P. Souères, and N. Mansard, "Cat: Constraints as terminations for legged locomotion reinforcement learning," in *2024 IEEE/RSJ International Conference*

on Intelligent Robots and Systems (IROS). IEEE, 2024, pp. 13 303–13 310.

[26] A. Ray, J. Achiam, and D. Amodei, "Benchmarking safe exploration in deep reinforcement learning," *arXiv preprint arXiv:1910.01708*, vol. 7, no. 1, p. 2, 2019.

[27] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," in *Robotics: Science and Systems (RSS)*, 2024.

[28] M. Rigter, B. Lacerda, and N. Hawes, "One risk to rule them all: A risk-sensitive perspective on model-based offline reinforcement learning," *Advances in neural information processing systems (NeurIPS)*, vol. 36, pp. 77 520–77 545, 2023.

[29] C.-Y. Kuo, A. Schaarschmidt, Y. Cui, T. Asfour, and T. Matsubara, "Uncertainty-aware contact-safe model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3918–3925, 2021.

[30] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *Advances in neural information processing systems (NeurIPS)*, vol. 29, 2016.

[31] C. E. Luis, A. G. Bottero, J. Vinogradska, F. Berkenkamp, and J. Peters, "Value-distributional model-based reinforcement learning," *Journal of Machine Learning Research*, vol. 25, no. 298, pp. 1–42, 2024.

[32] ——, "Model-based uncertainty in value functions," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2023, pp. 8029–8052.

[33] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International conference on machine learning (ICML)*. PMLR, 2017, pp. 449–458.

[34] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, vol. 32, no. 1, 2018.

[35] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *International conference on machine learning (ICML)*. PMLR, 2018, pp. 1096–1105.

[36] D. Yang, L. Zhao, Z. Lin, T. Qin, J. Bian, and T.-Y. Liu, "Fully parameterized quantile function for distributional reinforcement learning," *Advances in neural information processing systems*, vol. 32, 2019.

[37] S. Li, Y. Pang, P. Bai, J. Li, Z. Liu, S. Hu, L. Wang, and G. Wang, "Learning locomotion for quadruped robots via distributional ensemble actor-critic," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1811–1818, 2024.

[38] W. Chen, D. Subramanian, and S. Paternain, "Probabilistic constraint for safety-critical reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 69, no. 10, pp. 6789–6804, 2024.

[39] S. Bhatnagar and K. Lakshmanan, "An online actor–critic algorithm with function approximation for constrained markov decision processes," *Journal of Optimization Theory and Applications*, vol. 153, no. 3, pp. 688–708, 2012.

[40] Q. Liang, F. Que, and E. Modiano, "Accelerated primal-dual policy optimization for safe reinforcement learning," *arXiv preprint arXiv:1802.06480*, 2018.

[41] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*. PMLR, 2017, pp. 22–31.

[42] Y. Emam, G. Notomista, P. Glotfelter, Z. Kira, and M. Egerstedt, "Safe reinforcement learning using robust control barrier functions," *IEEE Robotics and Automation Letters*, 2022.

[43] E. Chane-Sane, P.-A. Leziart, T. Flayols, O. Stasse, P. Souères, and N. Mansard, "Cat: Constraints as terminations for legged locomotion reinforcement learning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 13 303–13 310.

[44] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," in *International Conference on Learning Representations (ICLR)*, 2020.

[45] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Comput. Surv.*, vol. 54, no. 5, Jun. 2021.

[46] W. Zhu and M. Hayashibe, "A hierarchical deep reinforcement learning framework with high efficiency and generalization for fast and safe navigation," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 5, pp. 4962–4971, 2023.

[47] T.-Y. Yang, T. Zhang, L. Luu, S. Ha, J. Tan, and W. Yu, "Safe reinforcement learning for legged locomotion," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2454–2461.

[48] J. Morgan, "Riskmetrics-technical document," *Morgan Guaranty Trust Company*, 1996.

[49] R. T. Rockafellar, S. Uryasev *et al.*, "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.

[50] S. S. Wang, "A class of distortion operators for pricing financial and insurance risks," *Journal of risk and insurance*, pp. 15–36, 2000.

[51] K. Hsu, V. Rubies-Royo, C. Tomlin, and J. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," in *Robotics*, ser. Robotics: Science and Systems (RSS), D. Shell, M. Toussaint, and M. Hsieh, Eds. United States: MIT Press Journals, 2021, publisher Copyright: © 2021, MIT Press Journals, All rights reserved.; 17th Robotics: Science and Systems, RSS 2021 ; Conference date: 12-07-2021 Through 16-07-2021.

[52] N.-C. Huang, P.-C. Hsieh, K.-H. Ho, and I.-C. Wu, "Ppo-clip attains global optimality: Towards deeper understandings of clipping," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 11, 2024, pp. 12 600–12 607.

[53] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2018.

[54] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1928–1937.

[55] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Proceedings of The 6th Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 403–415.

[56] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 32, no. 1, Apr. 2018.

[57] "Isaac lab documentation: Terrains api," https://isaac-sim.github.io/IsaacLab/main/source/api/lab/isaaclab.terrains.html, accessed: 2025-09-04.

[58] Unitree Robotics, "Unitree go1 user manual (english)," https://static.generation-robots.com/media/user-manual-go1-unitree-robotics-en.pdf, accessed: 2025-09-04.

[59] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 91–100.

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

[61] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[62] D. Jain, K. Caluwaerts, and A. Iscen, "From pixels to legs: Hierarchical learning of quadruped locomotion," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 91–102.

[63] J. Byrd and Z. Lipton, "What is the effect of importance weighting in deep learning?" in *International conference on machine learning*. PMLR, 2019, pp. 872–881.

[64] J. Ren, M. Zhang, C. Yu, and Z. Liu, "Balanced mse for imbalanced visual regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022, pp. 7926–7935.

[65] Y. Lu, Y. Dong, J. Ma, J. Zhang, and P. Lu, "Learning an adaptive fall recovery controller for quadrupeds on complex terrains," *arXiv preprint arXiv:2412.16924*, 2024.

[66] N. I. Lab, "Go1 rough environment configuration," https://github.com/isaac-sim/IsaacLab/blob/main/source/isaaclab_tasks/isaaclab_tasks/manager_based/locomotion/velocity/config/go1/rough_env_cfg.py, 2025, accessed: 2025-09-11.