Dynamic Legged Ball Manipulation on Rugged Terrains with Hierarchical Reinforcement Learning

***Abstract*—Achieving reliable object manipulation while traversing complex terrains is the missing link between agile quadruped locomotion and practical autonomy. Specifically, using traditional end-to-end reinforcement learning (RL) for dynamic ball manipulation in rugged environments presents two key challenges. The first is coordinating distinct motion modalities to integrate terrain traversal and ball control seamlessly. The second is overcoming sparse rewards in end-to-end RL, which impedes efficient policy convergence. To address these challenges, we propose a hierarchical RL framework. A high-level policy, informed by proprioceptive data and ball position, adaptively switches between pre-trained low-level skills such as ball dribbling and rough terrain navigation. We further propose Dynamic Skill-Focused Policy Optimization to suppress gradients from inactive skills and enhance critical skill learning. Both simulation and real-world experiments validate that our method outperforms baseline approaches in dynamic ball manipulation across rugged terrains, highlighting its effectiveness in challenging environments.**

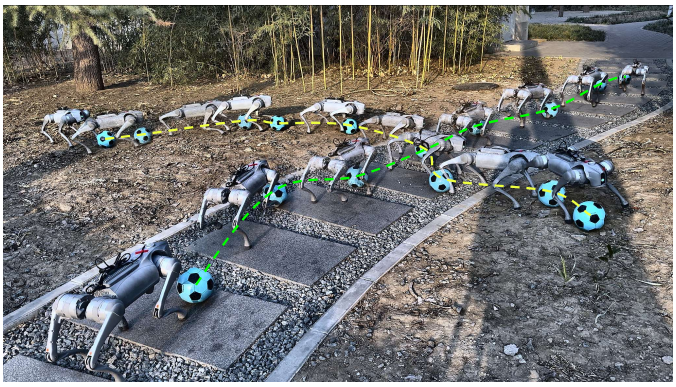


Fig. 1. **Experiments on dynamic ball dribbling over rugged terrains.** The green trajectory traces downhill dribbling over stone slabs and loose gravel; the yellow trajectory shows dribbling across a mixed surface of wet mud, raised curbs, stone slabs, and loose gravel.

I. Introduction

With rapid advances in robotics and AI, quadruped robots can perform agile movements such as running [1], parkour [2], and football shooting [3], showing great potential in various practical applications. Currently, loco-manipulation—manipulating objects while executing dynamic movements—remains a central challenge [4]. By reusing the legs for manipulation, loco-manipulation enhances the multi-tasking ability of quadruped robots and reduces operational costs. This capability is crucial for tasks such as search-and-rescue, package delivery, and robot soccer competitions. Ball manipulation, as a canonical benchmark for such loco-manipulation tasks, has received extensive research attention in recent years [5], [6].

As a dynamic mobile manipulation task, ball manipulation necessitates a seamless integration of visual perception, dynamic locomotion, and object manipulation. DribbleBot [5] trains in simulation with reinforcement learning (RL) and domain randomization, demonstrating its ability to perform zero-shot transfer to dribble the ball on flat ground. DexDribbler [6] adds a reward term for feedback control and a context-aided estimator to strengthen ball control. However, these works lack the capability to both traverse complex terrains and manipulate dynamic objects on rugged surfaces.

Extending dribbling from flat ground to rugged terrain is not merely a matter of improving robustness but introduces a fundamental challenge. The task is inherently hybrid, requiring seamless switching between two *distinct motion modalities*—one centered on terrain traversal that prioritizes the robot's own stability, and the other centered on manipulation that demands precise control of an external object. These modalities impose conflicting requirements on the control policy [7], making standard end-to-end RL approaches difficult to train due to sparse reward signals and exploration challenges [5], [6]. To address these hybrid dynamics, we propose a Hierarchical Reinforcement Learning (HRL) framework where a high-level policy coordinates pre-trained, task-specialized low-level skills (i.e., dribbling and traversal) (Fig. 1).

However, optimizing this policy in a hybrid action space—comprising discrete skill selection and continuous command generation—is non-trivial. Standard methods often struggle with credit assignment for unexecuted skills, leading to training instability. To address this, we propose Dynamic Skill-Focused Policy Optimization (DSF-PO). By incorporating a confidence-aware weighting mechanism, DSF-PO effectively regulates gradient updates, mitigating the premature mode collapse often observed in such hybrid control tasks. Furthermore, targeted reward design and curriculum learning are employed to further boost policy convergence.

The main contributions of our work are as follows:

1) A hierarchical RL framework for agile control over low-level skills to achieve dynamic ball manipulation on rugged terrains.
2) A novel loss formulation, DSF-PO, designed for handling the simultaneous discrete and continuous outputs of the high-level policy.
3) Demonstration of ball manipulation on rugged terrains in simulation and successful transfer to a real-world robot.

II. Related Work

A. Locomotion and Manipulation with Quadruped Robots

Recent advances in learning-based methods have enabled quadruped robots to perform increasingly complex locomotion and manipulation tasks [1], [8]–[13]. A dominant paradigm is training controllers in physics simulators, often with large-scale GPU parallelism, and subsequently deploying them in

the real world [14]–[19]. For manipulation, learning-based approaches typically generate end-effector trajectories [20], [21] or learn policies directly. Trajectory-following methods, for instance, often use RL-based motion tracking networks to follow Bezier curves generated by a high-level planner [3], [22]–[24]. For more dynamic tasks like in-motion ball dribbling, direct RL optimization with techniques such as domain randomization has been explored [5], [6].

However, these methods have primarily focused on flat-ground scenarios. Robustly integrating dynamic locomotion with precise manipulation to achieve tasks like dribbling on complex, rugged terrains remains a significant challenge. This motivates our development of a unified framework that seamlessly integrates locomotion and dribbling strategies.

B. Hierarchical RL for Robot Control

HRL frameworks, such as the options framework [25] and goal-conditioned hierarchies [26], are a natural paradigm for such complex tasks, offering advantages in exploration and long-horizon credit assignment [27]. In robotics, these frameworks are primarily classified into two categories. The first approach utilizes a task-independent low-level controller, which can be a learning-based trajectory tracker [23], [24], [28], [29] or a model-based generator [30]. This low-level controller is commanded by a task-specific high-level policy that provides either goal positions [31], [32] or trajectory parameters [3], [22]. The second category focuses on integrating multiple specialized skills. For instance, some works incorporate diverse skills using residual modules [33], [34], while others suggest that different skills can specialize in distinct behaviors and be composed, akin to a mixture-of-experts [7].

Inspired by hierarchical control frameworks that leverage both task decomposition and multi-skill integration, we adopt this approach for legged ball dribbling. By leveraging a hierarchical structure, our method ensures flexible skill coordination and enhances adaptability to diverse environments.

III. Methodology

A. Problem Formulation

We model cross-terrain dribbling as a partially observable Markov decision process (POMDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \gamma), \quad \mathbf{o}_t = \Phi(\mathbf{s}_t) \in \mathcal{O},\ \mathbf{a}_t \in \mathcal{A}, \quad (1)$$

with state $\mathbf{s}_t = (\mathbf{x}_t^r, \mathbf{x}_t^b, \eta_t) \in \mathcal{S}$, where $\mathbf{x}_t^r$ is the robot state, $\mathbf{x}_t^b = (\mathbf{p}_t^{\text{ball}}, \mathbf{v}_t^{\text{ball}})$ the ball state, and η_t a terrain embedding. Contact modality between robot and ball is distinguished by the impulse indicator

$$i_t \triangleq \mathbf{1}(\|\mathbf{J}_t\| > 0) \in \{0, 1\}, \quad (2)$$

where $\mathbf{J}_t$ denotes the net robot–ball impulse on the ball over $(t - \Delta t/2, t + \Delta t/2)$. The transition kernel decomposes by contact modality as

$$\mathcal{T}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t) = \begin{cases} \mathcal{T}_{\text{nc}}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t), & i_t = 0, \\ \mathcal{T}_{\text{c}}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t), & i_t = 1. \end{cases} \quad (3)$$

Since $\mathcal{T}_{\text{nc}}$ and $\mathcal{T}_{\text{c}}$ differ in their dynamics, the two modalities are treated as distinct regimes within the formulation while sharing the same observation and action spaces. The objective of policy π is

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(\mathbf{s}_t, \mathbf{a}_t)\right]. \quad (4)$$

B. Hierarchical Policy Architecture

To address the hybrid dynamics discussed above, we propose a novel hierarchical framework depicted in Fig. 2. This architecture is composed of two layers: a library of pre-trained low-level skills that execute fundamental behaviors, and a high-level policy that learns to coordinate them.

1) Low-level Skills: The foundation of our framework is a set of four pre-trained, specialized low-level skills that tackle the distinct motion modalities of the task. Specifically, two dribbling skills perform dribbling when the ball is nearby (corresponding to $\mathcal{T}_{\text{c}}$), whereas two locomotion skills handle rapid ball approaching from a distance or traversing rough terrains (corresponding to $\mathcal{T}_{\text{nc}}$). They serve as a modular library of primitive actions for the high-level policy to compose.

- Dribbling Skills (π_1^L, π_2^L): These skills are designed to handle the contact modality. Their objective is to use the robot's front limbs to impart an appropriate impulse to the ball, so that it rolls with a prescribed direction and speed. We train these skills with PPO [35] in Isaac Gym [36] under a flat-terrain setting. The training setup for the dribbling skills adopts similar designs inspired by prior work, with π_1^L following [5] and π_2^L following [6]. Specifically, π_1^L executes a stronger kick, while π_2^L applies a gentler kick and consequently achieves a smaller turning radius.
- Locomotion Skills (π_3^L, π_4^L): These skills address the non-contact modality, aiming to enable the robot to advance toward the ball over uneven terrain while keeping the ball within its controllable range as much as possible. We train these skills using PPO on both flat terrain (for π_3^L) and rugged terrain (for π_4^L). The terrain design for the latter follows the *Environment Design* description in Sec. III-B2. Specifically, π_3^L is capable of moving at relatively high speeds on flat terrain, whereas π_4^L enables robust traversal over rough and sloped surfaces.

For each skill, the observation vector is

$$\left[\, \mathbf{c}_t^L,\ \mathbf{o}_t^{\text{proprio}},\ \boldsymbol{\theta}_t,\ (\mathbf{p}_t^{\text{ball}} \text{ if dribbling}) \,\right],$$

where $\mathbf{c}_t^L$ denotes the low-level command, $\mathbf{o}_t^{\text{proprio}}$ is a subset of proprioceptive signals, and $\boldsymbol{\theta}_t$ is a timing reference variable following [5]. The proprioceptive subset includes joint positions and velocities $\mathbf{q}_t, \dot{\mathbf{q}}_t$, the gravity unit vector $\mathbf{g}_t$ in the body frame, and the global body yaw ψ_t. The command $\mathbf{c}_t^L$ encodes a target ball-velocity command $(v_x^{\text{cmd}}, v_y^{\text{cmd}})$ or a locomotion command $(v_x^{\text{cmd}}, v_y^{\text{cmd}}, \omega^{\text{cmd}})$. For dribbling skills, the ball position $\mathbf{p}_t^{\text{ball}}$ is additionally provided. Each policy takes a 30-step historical observation vector as input and outputs an action $\mathbf{a}_t \in \mathbb{R}^{12}$, interpreted as joint-position targets for the quadruped's twelve actuators, applied at 50 Hz.

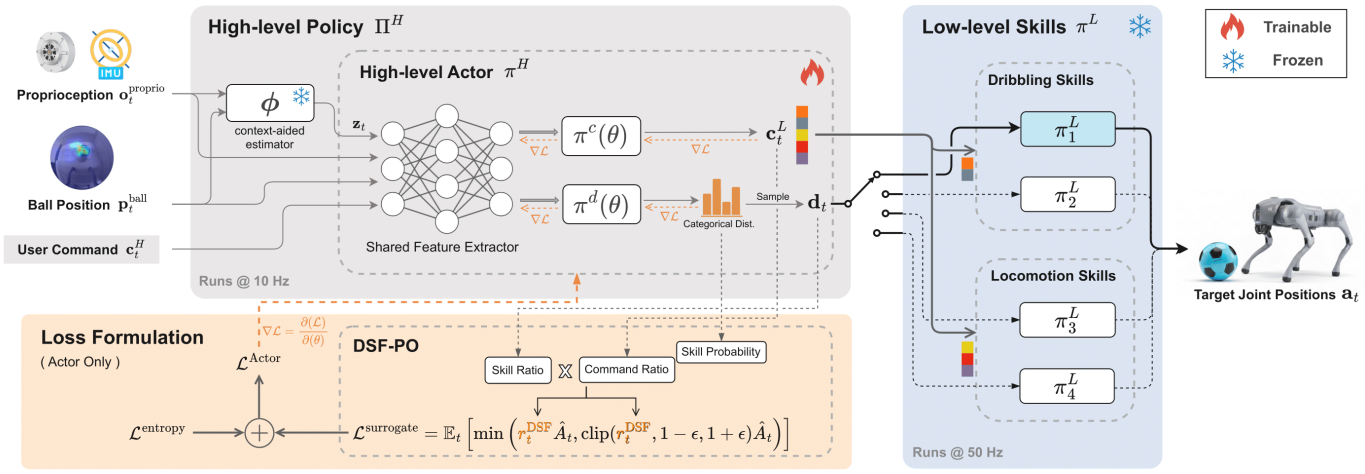


Fig. 2. **Hierarchical policy architecture and DSF-PO training paradigm.** A high-level policy running at 10 Hz consumes three types of observations to i) produce a categorical distribution over dribbling vs. locomotion skills and sample a skill index, and ii) output continuous command signals comprising both dribbling and locomotion commands. The actor's two heads share a feature extractor augmented by a frozen context-aided estimator. Pre-trained low-level skills execute at 50 Hz to track their respective subsets of high-level commands and generate the quadruped's target joint positions. Training optimizes the actor primarily with the DSF-PO loss, whose importance ratio is jointly modulated by the skill ratio and the skill probability. Value network is omitted for clarity.

2) High-level Policy: The high-level policy coordinates pre-trained low-level skills and orchestrates seamless transitions between dribbling and locomotion. Unlike [21], we adopt RL to train the high-level policy for robust ball-dribbling on rugged terrains. The high-level policy Π^H consists of a context-aided estimator ϕ and a high-level actor π^H (Fig. 2). The estimator ϕ provides structured, environment- and motion-related information to the actor. Specifically, it takes proprioception and ball position as input and predicts terrain parameters, the robot's posture, and the ball's motion states, yielding a compact context vector $\mathbf{z}_t$. Following [6], ϕ is trained with supervised learning using simulator ground truth and kept frozen during high-level policy training.

Observation Space: The high-level observation space $\mathbf{O}^H$ contains three components: proprioceptive signals, ball position, and user commands. The estimator ϕ processes the first two to produce $\mathbf{z}_t$, while the actor π^H consumes the full observation, i.e.,

$$\mathbf{o}_t = \left[\mathbf{z}_t,\ \mathbf{q}_t,\ \dot{\mathbf{q}}_t,\ \mathbf{g}_t,\ \boldsymbol{\psi}_t,\ \ \mathbf{p}_t^{\text{ball}},\ \mathbf{c}_t^H,\ \ \mathbf{a}_{t-1}\right],$$

where $\mathbf{c}_t^H$ are user commands specifying the target ball velocity $\mathbf{v}_t^{\text{cmd}}$ in the global frame.

Action Space: The high-level actor π^H outputs a hybrid action $(d_t, \mathbf{c}_t^L)$, where $d_t \in \{1, \ldots, 4\}$ is a skill index and $\mathbf{c}_t^L \in \mathbb{R}^5$ is a partitioned command vector serving both 2D dribbling and 3D locomotion skills:

$$\mathbf{c}_t^L = \left[v_x^{\text{cmd}}, v_y^{\text{cmd}} \;\middle|\; v_x^{\text{cmd}}, v_y^{\text{cmd}}, \omega^{\text{cmd}}\right].$$

The appropriate sub-vector $\mathbf{c}_t^{d_t}$ is extracted via a skill-indexed selection matrix $\mathbf{S}_{d_t} \in \{\mathbf{S}_{\text{drib}}, \mathbf{S}_{\text{loc}}\}$, where the selection matrices are defined in block form as $\mathbf{S}_{\text{drib}} = [\ \mathbf{I}_2 \mid \mathbf{0}_{2\times 3}\ ]$ and $\mathbf{S}_{\text{loc}} = [\ \mathbf{0}_{3\times 2} \mid \mathbf{I}_3\ ]$. The selection is performed as:

$$\mathbf{c}_t^{d_t} = \mathbf{S}_{d_t}\mathbf{c}_t^L. \tag{5}$$

For locomotion skills ($d_t \in \{3, 4\}$), this command is then applied as a residual to the user command $\mathbf{c}_t^H$ with a scaling factor $\alpha \in (0, 1]$:

$$\tilde{\mathbf{c}}_t^{\text{loc}} = \mathbf{c}_t^H + \alpha\,\mathbf{c}_t^{d_t}. \tag{6}$$

Network Architecture: As shown in Fig. 2, the actor network employs a shared feature extractor to process observations. To handle the hybrid action space, the network branches into two probabilistic heads: the Skill Head π^d outputs a categorical distribution for discrete skill selection, and the Command Head π^c parameterizes a normal distribution for continuous low-level control $\mathbf{c}_t^L$.

Environment Design: We design five types of terrain—flat ground, ramp-up, ramp-down, rough terrain, and stair descent—to train the robot's ball dribbling ability in complex environments. At the start of every episode, the robot is randomly placed on one of these terrains with a random yaw orientation, and its joint angles are initialized around a nominal pose. The soccer ball is sampled within a $2\,\text{m}$ radius of the robot. We adopt the domain randomization settings in [5], specifically incorporating ball teleportation and randomized delays, to randomize the properties of the robot and the ball, thereby improving real-world robustness. User commands $\mathbf{c}_t^H$ are sampled via a curriculum, as detailed in Sec. III-C2.

C. Policy Training

In this section, we detail the training strategy for the high-level policy. To ensure efficient and stable learning, we introduce two key components: a novel loss formulation, Dynamic Skill-Focused Policy Optimization (DSF-PO), and a comprehensive training scheme involving reward shaping and curriculum learning.

1) Dynamic Skill-Focused Policy Optimization: As described in Sec. III-B2, the high-level policy, which selects low-level skills, mirrors Mixture-of-Experts (MoE) models like

Switch Transformer [37] and introduces similar optimization challenges. Specifically, the high-level actor emits a hybrid action $(d_t, \mathbf{c}_t^L)$, where only the selected sub-vector $\mathbf{c}_t^{d_t}$ is actually executed by the environment. If one applies the vanilla PPO likelihood to the entire command vector, the log-probability decomposes across all skill-specific subspaces,

$$\log \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) = \log \pi_\theta^d\left(d_t \mid \mathbf{s}_t\right) + \sum_{k=1}^{K} \log \pi_\theta^c\left(\mathbf{c}_t^k \mid \mathbf{s}_t, k\right), \tag{7}$$

so gradients can flow into $\pi^c(\cdot \mid \cdot, k)$ for $k \neq d_t$—even though those commands were not executed. This creates an *optimization-execution mismatch*. To address this, we propose dynamic skill-focused policy optimization. We first formalize the joint policy $\pi_\theta(\mathbf{a}|\mathbf{s})$ with two output heads: a discrete skill selector $\pi_\theta^d(d|\mathbf{s})$ and a continuous command generator $\pi_\theta^c(\mathbf{c}|\mathbf{s}, d)$. The skill selector samples $d \in \{1, \ldots, K\}$, while the command generator outputs $\mathbf{c}$, which is a union of different commands corresponding to each skill, $\mathbf{c} = \left(\mathbf{c}^1, \mathbf{c}^2, \ldots, \mathbf{c}^K\right)$, where $\mathbf{c}^k \in \mathbb{R}^{n_k}$ is the command for skill k. Once a skill d is selected, the low-level network processes only the corresponding command $\mathbf{c}^d$.

The action generation process begins with the skill selector which defines a categorical distribution over the K skills using softmax probabilities:

$$d \sim \text{Categorical}(p_1, \ldots, p_K),\ p_d = \pi_\theta^d(d|\mathbf{s}) = \frac{e^{z_d}}{\sum_{k=1}^{K} e^{z_k}}, \tag{8}$$

where z_d are the unnormalized logits. The command generator then provides the parameters for a multivariate normal distribution:

$$\mathbf{c}^d \sim \mathcal{N}(\mu^d, \Sigma^d), \quad \mu^d = \pi_\theta^c(\mathbf{s}, d). \tag{9}$$

Thus, the overall policy $\pi_\theta(a|s)$ can be decomposed as:

$$\pi_\theta(\mathbf{a}|\mathbf{s}) = \pi_\theta^d(d|\mathbf{s}) \cdot \prod_{k=1}^{K} \mathcal{N}(\mathbf{c}^k; \mu_\theta^k, \Sigma^k)^{\mathbb{I}(k=d)}, \tag{10}$$

where $\mathbb{I}(k = d)$ is an indicator function that selects the appropriate command $\mathbf{c}^k$ based on the selected skill index d. To theoretically respect the trust region of this joint policy, we analyze its KL divergence decomposition:

$$\begin{aligned} D_{KL}(\pi_\theta||\pi_{\text{old}}) = D_{KL}(\pi_\theta^d||\pi_{\text{old}}^d) \\ + \sum_{k=1}^{K} w_k(\mathbf{s}) D_{KL}(\pi_\theta^c(\cdot|k)||\pi_{\text{old}}^c(\cdot|k)), \end{aligned} \tag{11}$$

where $w_k(\mathbf{s}) = \pi_\theta^d(k|\mathbf{s})$ represents the probability of selecting skill k. This decomposition reveals a critical insight: the optimization of a skill's command distribution could be weighted by its usage probability w_k. Unlike heuristic gradient masking which treats selection as deterministic, DSF-PO strictly adheres to this bound. This introduces a confidence-aware regularization, scaling down gradients under uncertainty.

Accordingly, we reformulate the PPO importance ratio to incorporate this weighting. The *Dynamic Skill-Focused importance ratio* $r_t^{\text{DSF}}(\theta)$ is defined as:

$$r_t^{\text{DSF}}(\theta) = \underbrace{\frac{\pi_\theta^d(d_t|\mathbf{s}_t)}{\pi_{\theta_{\text{old}}}^d(d_t|\mathbf{s}_t)}}_{\text{Skill Ratio}} \cdot \prod_{k=1}^{K} \left(\underbrace{\frac{\mathcal{N}(\mathbf{c}_t^k; \mu_\theta^k, \Sigma^k)}{\mathcal{N}(\mathbf{c}_t^k; \mu_{\theta_{\text{old}}}^k, \Sigma^k)}}_{\text{Command Ratio}} \right)^{w_k(\mathbf{s}_t) \cdot \mathbb{I}(k=d_t)}. \tag{12}$$

The final surrogate loss function is then formulated as:

$$\mathcal{L}^{\text{surrogate}} = \mathbb{E}_t \left[\min\left(r_t^{\text{DSF}} \hat{A}_t, \text{clip}(r_t^{\text{DSF}}, 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right], \tag{13}$$

where $\hat{A}_t$ is the estimated advantage and ϵ is the clipping parameter.

We employ PPO with Asymmetric Actor-Critic [38] to train the high-level policy. The actor receives partial observations $\mathbf{o}_t$, while the critic accesses the full state $\mathbf{s}_t$, including the robot's base velocity and ball state.

2) Reward Shaping and Curriculum Learning: We present a comprehensive training strategy inspired by [5] to deal with the challenges of skill switching and training stability in high-level ball dribbling control. To strike a balance in skill transitions, the high-level policy operates at a lower inference frequency (e.g., 10 Hz) than low-level skills. Additionally, we incorporate carefully designed reward terms and curriculum learning tailored to terrains and commands, as detailed in the following.

Reward Design: To prevent the high-level policy from excessively focusing on low-level actions, the reward function is restricted to terms that are indispensable for stable learning, which consist of *five* components shown in Table I: 1) encouraging the robot to maintain its balance; 2) encouraging the robot to approach and orient toward the ball; 3) rewarding the consistent output of the same skill index d_t over time; 4) rewarding the proximity between the ball's actual and expected velocity; 5) encouraging the robot to use dribbling skills when approaching the ball. To ensure stable training, a bounded summation function is employed to aggregate all reward terms, with exponential kernels for weighted integration.

Curriculum Learning: To ensure stable from-scratch training of the high-level policy, we employ two curricula. The terrain curriculum progressively increases the steepness and ruggedness of challenging environments (e.g., stairs, ramps, and rough terrains). Simultaneously, the command curriculum expands the range of user commands the robot must follow.

We define $p^k_{\mathbf{c}^H,\mathbf{t}}$ as the joint distribution of the user commands $\mathbf{c}^H$ and terrain difficulty $\mathbf{t}$ used in the sampling process during the k-th episode. An efficient approach is to maintain independent distributions over $p_{\mathbf{c}^H}, p_{\mathbf{t}}$ such that $p_{\mathbf{c}^H,\mathbf{t}} = p_{\mathbf{c}^H} \cdot p_{\mathbf{t}}$. Referring to [1], we employ the *box adaptive curriculum update rule* with a reward-based approach. Specifically, we initialize $p^0_{\mathbf{c}^H,\mathbf{t}}$ as a uniform probability distribution over $(\mathbf{c}^H \in [-0.5, 0.5], \mathbf{t} \in \{0, 1\})$. The user commands and terrain difficulty for the robot are sampled independently at episode k: $\mathbf{c}^H \sim p^k_{\mathbf{c}^H}, \mathbf{t} \sim p^k_{\mathbf{t}}$. If the robot succeeds in this region, we expand the sampling distribution to include neighboring

TABLE I
REWARD TERMS FOR HIGH-LEVEL POLICY TRAINING

Reinforcement Learning		
Term	Expression	Weight
Projected Gravity	$\lvert\mathbf{g}_{\text{xy}}\rvert^2$	-5.0
Robot Ball Distance	$\exp\{-\delta_p\lvert\mathbf{b}-\mathbf{p}_{\text{FRHip}}\rvert^2\}$	4.0
Yaw Alignment	$\exp\{-\delta_\psi(e^2_{\text{rbcmd}}+e^2_{\text{rbbase}})\}$	4.0
Temporal Coherence Bonus	$\sum_{i=t-T}^{t}\mathbb{I}(d_i=d_{i-1})$	0.1
Transition Cost	$\mathbb{I}(d_t\neq d_{t-1})$	-0.005
Ball Velocity Norm	$\exp\{-\delta_n(\lvert\mathbf{v}^{\text{cmd}}\rvert-\lvert\mathbf{v}^b\rvert)^2\}$	8.0
Ball Velocity Angle	$1-(\psi_b-\psi_{\text{cmd}})^2/\pi^2$	8.0
Ball Velocity Error	$\exp\{-\delta_v\lvert\mathbf{v}^b-\mathbf{v}^{\text{cmd}}\rvert^2\}$	8.0
Dribbling Near Ball	$\mathbb{I}(\lvert\mathbf{b}-\mathbf{p}\rvert<d_{\max})\cdot\mathbb{I}(\mathbf{d_t}\in\{1,2\})$	1.0
Curriculum Learning		
Term	Expression	
$r_{\mathbf{c}^H}$	$\mathbb{I}(r_{\text{BallVelocityError}}>0.5)$	
$r_{\mathbf{t}}$	$\mathbb{I}(\mathbf{b}>d_{\text{th1}})\cdot\mathbb{I}(\lvert\mathbf{b}-\mathbf{p}\rvert<d_{\text{th2}})$	

regions, thereby increasing the task's difficulty. Suppose the robot receives rewards $r_{\mathbf{c}^H}$ and $r_{\mathbf{t}}$ for attempting to follow $\mathbf{c}^H$ and traverse the terrain with difficulty $\mathbf{t}$, respectively. We then apply the update rule:

$$p^{k+1}_{\mathbf{c}^H}\left(\mathbf{c}^H_{\text{n}}\right)\leftarrow\begin{cases}p^k_{\mathbf{c}^H}\left(\mathbf{c}^H_{\text{n}}\right) & \text{if } r_{\mathbf{c}^H}=1,\\ 1 & \text{otherwise.}\end{cases}\tag{14a}$$

$$p^{k+1}_{\mathbf{t}}\left(\mathbf{t}_{\text{n}}\right)\leftarrow\begin{cases}p^k_{\mathbf{t}}\left(\mathbf{t}_{\text{n}}\right) & \text{if } r_{\mathbf{t}}=1,\\ 1 & \text{otherwise.}\end{cases}\tag{14b}$$

Eq. (14) indicates the probability density on neighbors $\mathbf{c}^H_{\text{n}}$ of $\mathbf{c}^H$ and $\mathbf{t}_{\text{n}}$ of $\mathbf{t}$ are increased. In our implementation, $\mathbf{c}^H_{\text{n}}\in\{\mathbf{c}^H-0.1,\mathbf{c}^H+0.1\},\mathbf{t}_{\text{n}}\in\{\mathbf{t}+1\}$. The reward terms are listed in Table I.

IV. RESULTS

In this section, we design both simulation studies and real-world experiments to evaluate the effectiveness of the proposed method and compare its ball dribbling performance on rugged terrains with previous approaches.

A. Experiment Results

1) Learning Performance: To validate the effectiveness of DSF-PO, we perform a detailed ablation study. Training is conducted in Isaac Gym, leveraging 4096 parallel environments on an NVIDIA RTX 4090. We compare our full DSF-PO formulation against two baselines by selectively ablating its key mechanisms:

- Vanilla PPO: This baseline removes both the skill focus weight and the indicator function:

$$r_t^{\text{vanilla}}(\theta)=\frac{\pi^d_\theta(d_t\mid\mathbf{s}_t)}{\pi^d_{\theta_{\text{old}}}(d_t\mid\mathbf{s}_t)}\cdot\prod_{k=1}^{K}\frac{\mathcal{N}\left(\mathbf{c}^k_t;\mu^k_\theta,\Sigma^k\right)}{\mathcal{N}\left(\mathbf{c}^k_t;\mu^k_{\theta_{\text{old}}},\Sigma^k\right)}\tag{15}$$

- DSF-PO (Focus-Only): This variant ablates the skill focus weight but retains the indicator function:

$$r_t^{\text{Focus-Only}}(\theta)=\frac{\pi^d_\theta(d_t\mid\mathbf{s}_t)}{\pi^d_{\theta_{\text{old}}}(d_t\mid\mathbf{s}_t)}\cdot\frac{\mathcal{N}(\mathbf{c}^{d_t}_t;\mu^{d_t}_\theta,\Sigma^{d_t})}{\mathcal{N}(\mathbf{c}^{d_t}_t;\mu^{d_t}_{\theta_{\text{old}}},\Sigma^{d_t})}\tag{16}$$

TABLE II
HYPER-PARAMETERS FOR HIGH-LEVEL POLICY TRAINING

Hyperparameter	Value
Rollout buffer size	24 steps
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
PPO clipping ratio (ϵ)	0.2
Number of epochs per update	5
Minibatch size	4
Optimizer	Adam
Learning rate	1×10^{-3}
Entropy coefficient	0.01
Value function coefficient	1.0

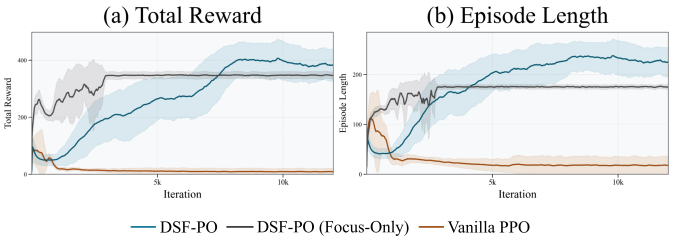


Fig. 3. Training curves of PPO with DSF-PO compared to vanilla PPO and DSF-PO with focus only. The shaded regions indicate the standard deviation over multiple runs.

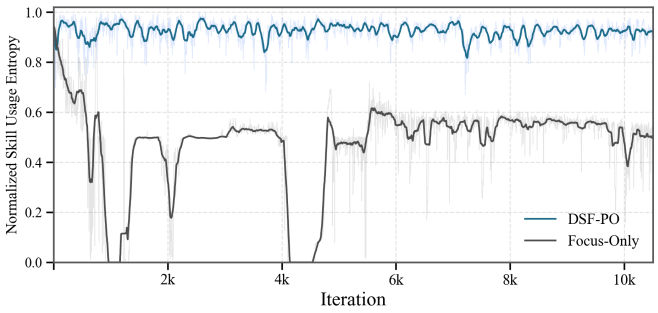


Fig. 4. **Normalized Skill Usage Entropy throughout training.** The Focus-Only baseline suffers from frequent entropy collapses, indicating premature commitment to single skills and loss of exploration. In contrast, DSF-PO maintains high entropy, demonstrating that confidence-aware updates effectively preserve skill diversity and prevent mode collapse during optimization.

All runs use PPO with standard legged-robot hyperparameters (Table II), identical across DSF-PO and other ablations. Fig. 3 shows the learning curves: the full DSF-PO consistently surpasses vanilla PPO, achieving higher return and longer episodes. Notably, although the Focus-Only ablation exhibits faster early improvement, it saturates at a suboptimal level. This limitation is explained by the training dynamics shown in Fig. 4: the baseline suffers from frequent entropy collapses, indicating that aggressive parameter updates drive the policy to prematurely commit to specific skills (mode collapse). In contrast, DSF-PO employs confidence-aware weighting to maintain consistently high entropy, which effectively preserves skill diversity and prevents the policy from locking into local optima.

2) Terrain Traversability: To evaluate the terrain traversability, we conduct two comparative studies. First, we compare the learned policy with DribbleBot [5] and

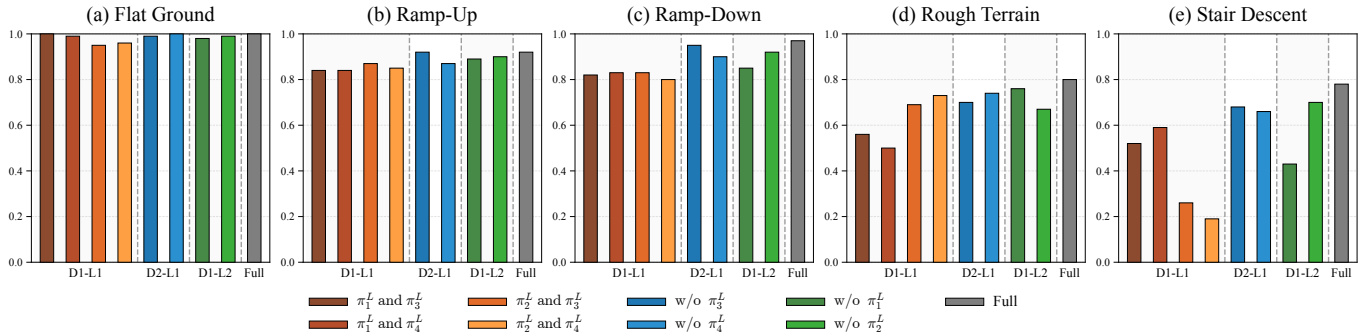


Fig. 5. **Low-level skill ablation across terrains.** Each panel shows per-terrain success rates based on 100 trials. Groups D1–L1, D2–L1, D1–L2, and Full denote the numbers of dribbling (D) and locomotion (L) skills: $(D, L) = (1, 1)$, $(2, 1)$, $(1, 2)$, $(2, 2)$.

TABLE III
BALL DRIBBLING SUCCESS RATES ACROSS DIFFERENT TERRAINS

Method	Flat Ground	Ramp-Up	Ramp-Down	Rough Terrain	Stair Descent
DribbleBot	**1.00**	0.89	0.90	0.53	0.61
DribbleBot-FT[1]	**1.00**	0.65	0.76	0.62	0.36
DribbleBot-S[2]	0.10	0.03	0.01	0.00	0.00
DexDribbler	**1.00**	0.91	0.92	0.22	0.13
DexDribbler-FT	**1.00**	0.90	0.93	0.30	0.08
DexDribbler-S	0.15	0.07	0.10	0.01	0.00
DribbleBot-Terrain	**1.00**	0.12	0.14	0.05	0.00
Ours	**1.00**	**0.92**	**0.97**	**0.80**	**0.78**

* Terrain settings: ramp-up $(m = 0.10)$; ramp-down $(m = -0.10)$; rough terrain $(\Delta h = 0.10\,\mathrm{m})$; stair descent $(h = 0.05\,\mathrm{m}, d = 0.50\,\mathrm{m})$.
[1] The suffix *FT* indicates the algorithm was fine-tuned on above terrains.
[2] The suffix *S* indicates the algorithm was trained from scratch on above terrains.

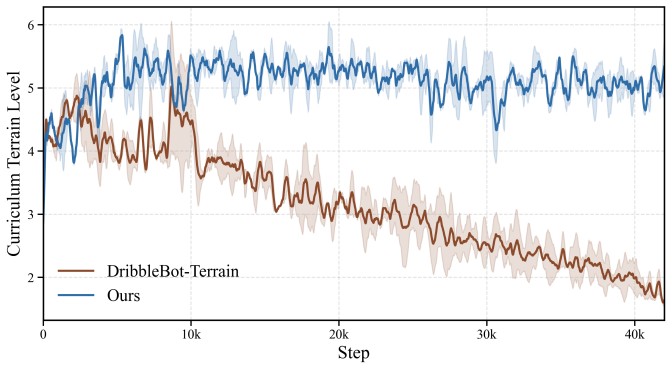


Fig. 6. **Training dynamics of the curriculum terrain level.** While our method consistently masters harder terrains, the baseline exhibits severe performance degradation.

DexDribbler [6]. Two baseline-oriented tasks are also included: i) fine-tuning the baselines on the environment introduced in Sec. III-B2, and ii) training the baselines from scratch under the same environment. Furthermore, to validate the necessity of the hierarchical architecture, we implement an enhanced monolithic baseline, denoted as DribbleBot-Terrain. To ensure a rigorous comparison, this baseline is explicitly trained using the exact same progressive terrain curriculum as our method and incorporates specialized reward terms (e.g., swing foot clearance and angular velocity penalties) to enhance robustness on rugged surfaces. All policies are tested under the same command: $\mathbf{c}_t = (1.0, 0.0)$. We conduct 100 tests per method on each terrain, where success is defined as the robot reaching the terrain boundary. As shown in Table III, most methods achieve 100% on flat ground, whereas performance diverges on harder terrains; our approach attains the best success rates on all non-flat settings. For the baselines, fine-tuning on the complex terrain yields at most marginal gains and sometimes substantial declines, and training from scratch on these terrains performs even worse than their flat-trained counterparts. As shown in Fig. 6, DribbleBot-Terrain exhibits severe training degradation, which translates into poor quantitative results. Failure analysis reveals that these outcomes are overwhelmingly dominated by locomotion falls rather than ball loss, which reveals a fundamental optimization conflict in monolithic policies between stability and manipulation. In contrast, our HRL framework resolves this conflict via temporal abstraction, allowing the high-level policy to switch to specialized recovery behaviors when necessary. These results indicate that a single monolithic RL policy is insufficient for robust operation on complex terrains.

Further qualitative analysis shows that baselines fail during stair descent due to locomotion instability. Specifically, DribbleBot suffers from poor posture, while DexDribbler experiences frequent falls. This suggests that manipulation is fundamentally limited by the underlying locomotion robustness. In contrast, our HRL framework is more robust, failing only at perceptual or physical limits like tracking loss or insufficient torque on steep slopes. However, stair ascent remains a challenge as the ball often gets stuck against vertical risers, requiring future "chipping" maneuvers.

3) Ablation Study: To investigate the impact of skill diversity on navigation performance, we perform an ablation study on low-level skill combinations. Combinations are grouped into four categories: i) D1–L1, one dribbling and one locomotion skill; ii) D2–L1, two dribbling and one locomotion skill; iii) D1–L2, one dribbling and two locomotion skills; and iv) Full, the proposed 2D–2L composition. Success rates for all nine combinations are evaluated across five terrains under the same protocol as above. As shown in Fig. 5, performance increases with low-level capacity: D1–L1 degrades on rough terrain and stairs; adding a second dribbling or locomotion skill (D2–L1/D1–L2) yields consistent gains, and the Full 2D–2L setting is most reliable across all terrains, indicating that capacity and complementary skills drive traversability.

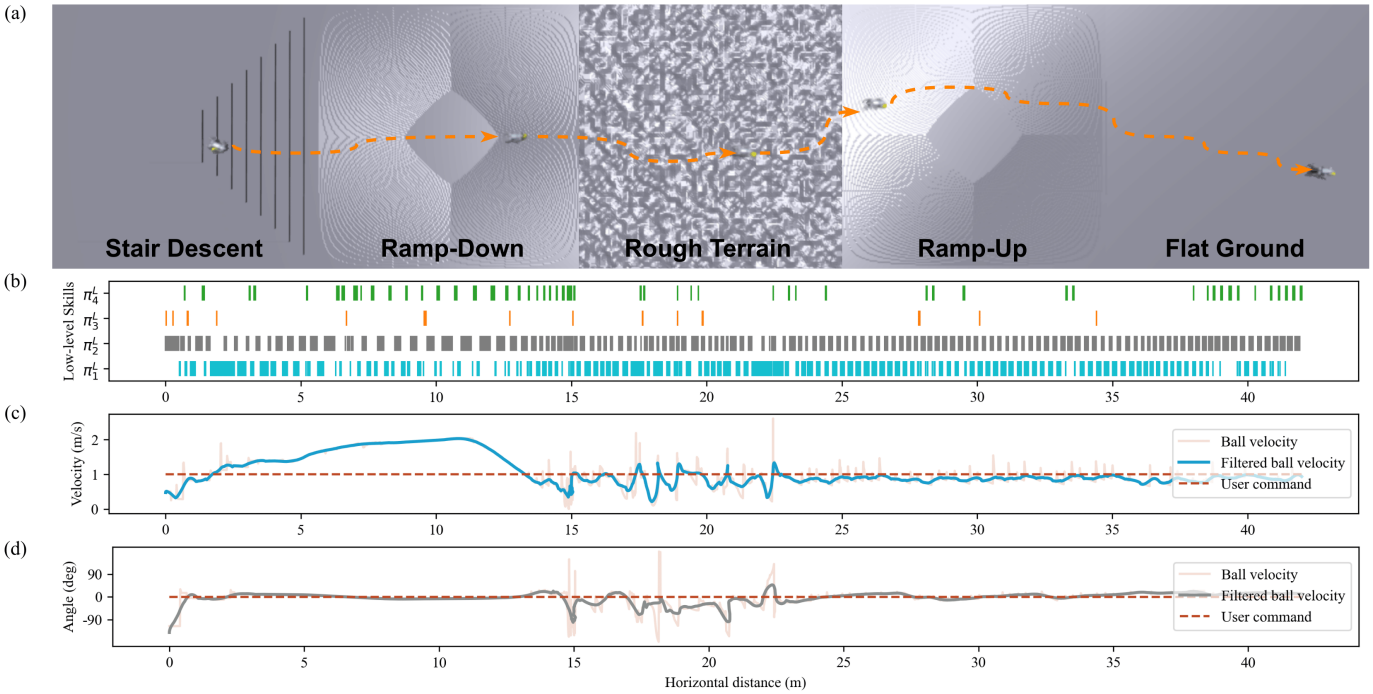
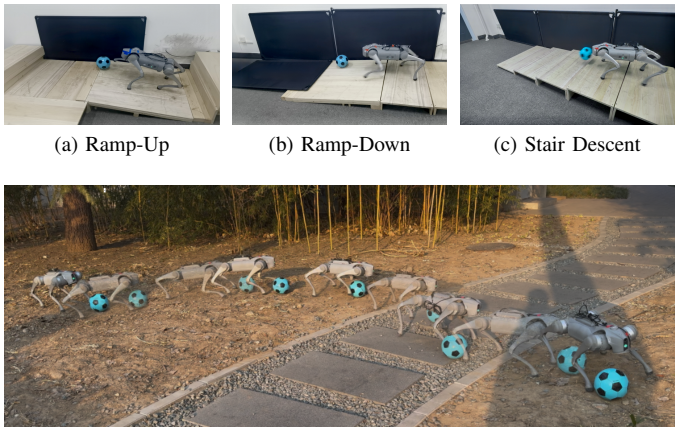


Fig. 7. **Cross-terrain dribbling performance evaluation.** (a) A trajectory schematic of the ball dribbling across five terrains in sequence. Each terrain measures 10 m per side. (b) Visualization of the invocation of different low-level skills, where each thin vertical line represents a single invocation. (c-d) Visualization of the ball's velocity magnitude and direction. The horizontal axis represents the robot's traveled horizontal distance.

TABLE IV
REAL WORLD BALL DRIBBLING PERFORMANCE EVALUATION

Method	Ramp-Up	Ramp-Down	Gravel	Stair Descent
DribbleBot	0/4	0/4	**4/4**	–
DexDribbler	**4/5**	1/5	**5/5**	–
Ours	**8/10**	**5/10**	**10/10**	**6/10**

4) Cross-Terrain Dribbling: As shown in Fig. 7, we evaluate the robot's dribbling ability across five terrains: stair descent, ramp-down, rough terrain, ramp-up, and flat ground. With a fixed command $\mathbf{c}_t^H = (1.0, 0.0)$, the robot successfully completes the task in 121 seconds. It maintains a stable direction except on rough terrain, where external disturbances cause deviations. Fig. 7(b) shows the invocation of low-level skills by the high-level policy across different terrains. The results indicate that dribbling skills are used most frequently, allowing the policy to adaptively switch between them based on terrain variations. Locomotion skills are invoked less frequently, mainly when the robot needs to reposition itself after drifting away from the ball. Fig. 7(c) and 7(d) illustrate the ball's velocity magnitude and direction, respectively. On downhill terrains (stair descent, ramp-down), the robot struggles to precisely control the ball's velocity due to gravitational acceleration. Velocity fluctuations are pronounced on rough terrain, mainly due to uneven surfaces. In contrast, on ramp-up and flat ground, the robot effectively regulates the ball's velocity, demonstrating greater stability and control.



(a) Ramp-Up (b) Ramp-Down (c) Stair Descent

(d) Cross-Terrain

Fig. 8. **Real-world deployment experiments on different terrains.** The indoor environment includes ramp-up, ramp-down, and stair descent, while the outdoor environment consists of a complex multi-terrain field.

B. Physical Deployment

We evaluate our method using a Unitree Go2 quadruped robot. We evaluate on four terrains (Fig. 8): ramp-up, ramp-down, stair descent (each step 50 cm wide, 5 cm high), and cross-terrain (mixed irregular ground, raised curbs, smooth stone pavement, soft gravel). The robot and ball start on flat ground and the objective is to drive the ball to the far end while maintaining balance. High-level commands are published via a remote controller. We conduct ten trials per terrain; data on successful terrain crossings are reported in Table IV. The baseline results are directly extracted from the original papers

to ensure a fair and consistent comparison.

Across all four terrains, our method attains the highest success rate, indicating stronger generalization. Qualitatively, the high-level policy adapts skill usage to terrain: on flat ground it maintains steady dribbling; on ramps it adjusts posture and limb coordination to counter slope while stabilizing ball control; on stairs it first kicks the ball down steps and then follows under stabilized gait; on rough outdoor ground it modulates stride length and stance width while switching skills to sustain stable dribbling. The main limitation of our policy lies in its reduced ability to control the initial ball speed on terrains of ramp-down and stair descent, making it difficult for the robot to keep pace with the ball.

V. Conclusion

In this paper, we propose a hierarchical RL framework that trains a high-level policy to coordinate dribbling and locomotion skills for dynamic ball manipulation on rugged terrains. We also introduce a dynamic skill-focused loss formulation to improve learning efficiency and convergence in mixed discrete-continuous action spaces. Simulation and real-world experiments show that our methods outperform previous works in generalizing across diverse terrains, achieving stable ball dribbling performance on rugged surfaces. Moreover, the framework is highly scalable, allowing the integration of more low-level skills to tackle complex tasks and diverse motion modalities. Future work will be devoted to extending low-level inputs to 3D to reduce positional ambiguity on uneven terrain, and to integrating additional multimodal skills within our hierarchical RL framework to enhance autonomy.

References

[1] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *The International Journal of Robotics Research*, 2024.

[2] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *CoRL*, 2023.

[3] Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath, "Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot," in *IROS*, 2022.

[4] S. Ha, J. Lee, M. van de Panne, Z. Xie, W. Yu, and M. Khadiv, "Learning-based legged locomotion: State of the art and future perspectives," *The International Journal of Robotics Research*, 2024.

[5] Y. Ji, G. B. Margolis, and P. Agrawal, "Dribblebot: Dynamic legged manipulation in the wild," in *ICRA*, 2023.

[6] Y. Hu, K. Wen, and F. Yu, "Dexdribbler: Learning dexterous soccer manipulation via dynamic supervision," in *IROS*, 2024.

[7] R. Huang, S. Zhu, Y. Du, and H. Zhao, "Moe-loco: Mixture of experts for multitask locomotion," in *IROS*, 2025.

[8] K. Jiang, Z. Fu, J. Guo, W. Zhang, and H. Chen, "Learning whole-body loco-manipulation for omni-directional task space pose tracking with a wheeled-quadrupedal-manipulator," *IEEE Robotics and Automation Letters*, 2025.

[9] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo, "Learning whole-body manipulation for quadrupedal robot," *IEEE Robotics and Automation Letters*, 2024.

[10] Y. Kim, H. Oh, J. Lee, J. Choi, G. Ji, M. Jung, D. Youm, and J. Hwangbo, "Not only rewards but also constraints: Applications on legged robot locomotion," *IEEE Transactions on Robotics*, 2024.

[11] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, 2022.

[12] I. M. A. Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," in *ICRA*, 2023.

[13] H. Zhang, H. Yu, L. Zhao, A. Choi, Q. Bai, Y. Yang, and W. Xu, "Learning multi-stage pick-and-place with a legged mobile manipulator," *IEEE Robotics and Automation Letters*, 2025.

[14] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, 2019.

[15] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, 2020.

[16] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *CoRL*, 2022.

[17] Q. Liu, P. Chen, K. Lin, K. Zhao, J. Ding, and Y. Li, "Sample-efficient backtrack temporal difference deep reinforcement learning," *Knowledge-Based Systems*, p. 114613, 2025.

[18] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, 2022.

[19] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *CoRL*, 2023.

[20] P. Arm, M. Mittal, H. Kolvenbach, and M. Hutter, "Pedipulate: Enabling manipulation skills using a quadruped robot's leg," in *ICRA*, 2024.

[21] X. Cheng, A. Kumar, and D. Pathak, "Legs as manipulator: Pushing quadrupedal agility beyond locomotion," in *ICRA*, 2023.

[22] X. Huang, Z. Li, Y. Xiang, Y. Ni, Y. Chi, Y. Li, L. Yang, X. B. Peng, and K. Sreenath, "Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning," in *IROS*, 2023.

[23] Z. He, K. Lei, Y. Ze, K. Sreenath, Z. Li, and H. Xu, "Learning visual quadrupedal loco-manipulation from demonstrations," in *IROS*, 2024.

[24] X. Huang, Q. Liao, Y. Ni, Z. Li, L. Smith, S. Levine, X. B. Peng, and K. Sreenath, "Hilma-res: A general hierarchical framework via residual rl for combining quadrupedal locomotion and manipulation," in *IROS*, 2024.

[25] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, 1999.

[26] O. Nachum, S. Gu, H. Lee, and S. Levine, "Near-optimal representation learning for hierarchical reinforcement learning," in *ICLR*, 2019.

[27] M. Hutsebaut-Buysse, K. Mets, and S. Latré, "Hierarchical reinforcement learning: A survey and open research challenges," *Machine Learning and Knowledge Extraction*, 2022.

[28] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, 2017.

[29] D. Jain, A. Iscen, and K. Caluwaerts, "Hierarchical reinforcement learning for quadruped locomotion," in *IROS*, 2019.

[30] W. Tan, X. Fang, W. Zhang, R. Song, T. Chen, Y. Zheng, and Y. Li, "A hierarchical framework for quadruped omnidirectional locomotion based on reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, 2023.

[31] J. Gehring, G. Synnaeve, A. Krause, and N. Usunier, "Hierarchical skills for efficient exploration," in *NeurIPS*, 2021.

[32] X. Yang, Z. Ji, J. Wu, Y.-K. Lai, C. Wei, G. Liu, and R. Setchi, "Hierarchical reinforcement learning with universal policies for multi-step robotic manipulation," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[33] K. N. Kumar, I. Essa, and S. Ha, "Cascaded compositional residual learning for complex interactive behaviors," *IEEE Robotics and Automation Letters*, 2023.

[34] N. Yokoyama, A. Clegg, J. Truong, E. Undersander, T.-Y. Yang, S. Arnaud, S. Ha, D. Batra, and A. Rai, "Asc: Adaptive skill coordination for robotic mobile manipulation," *IEEE Robotics and Automation Letters*, 2023.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] V. Makoviychuk *et al.*, "Isaac gym: High performance GPU based physics simulation for robot learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[37] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, 2022.

[38] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," in *Robotics: Science and Systems*, 2018.

APPENDIX A
ADDITIONAL RESULTS AND ANALYSIS

A. Robustness Analysis against Perception Failures

A critical challenge in real-world deployment is the discrepancy between the perfect perception available in simulation and the imperfect sensing on the physical robot. To evaluate the policy's robustness against out-of-view and occlusion, we design a simulator-side evaluation protocol that explicitly mimics physical perception limits.

1) Simulation of Perception Limits: We model perception availability $m_t \in \{0, 1\}$ as the intersection of geometric constraints and temporal data losses. The simulator provides a ball measurement only when $m_t = 1$, defined by:

$$m_t = m_t^{\text{fov}} \cdot m_t^{\text{GE}}. \tag{17}$$

Geometric Occlusion: We align the simulated sensor with the fisheye camera used in deployment. Let $\mathbf{p}_t^{\text{ball}} = [x_t, y_t]^\top$ denotes the ball position in robot's base frame. A measurement is valid ($m_t^{\text{fov}} = 1$) only if the ball falls within the effective field-of-view (FOV) cone defined by a half-angle $\theta_{\max}$ and a maximum reliable depth $d_{\max}$:

$$m_t^{\text{fov}} = \mathbb{I}\left(\frac{x_t}{\|\mathbf{p}_t^{\text{ball}}\|} \geq \cos\theta_{\max}\right) \cdot \mathbb{I}\left(\|\mathbf{p}_t^{\text{ball}}\| \leq d_{\max}\right), \tag{18}$$

where $\mathbb{I}(\cdot)$ is the indicator function. In our setup, we set $\theta_{\max} = 120°$ and $d_{\max} = 3.0\,\text{m}$.

Temporal Burst Loss: To capture bursty failures caused by motion blur or communication delays (which are temporally correlated), we employ a Gilbert-Elliott (GE) model. This implies a two-state Markov chain $z_t \in \{\text{G}, \text{B}\}$ with transition probabilities:

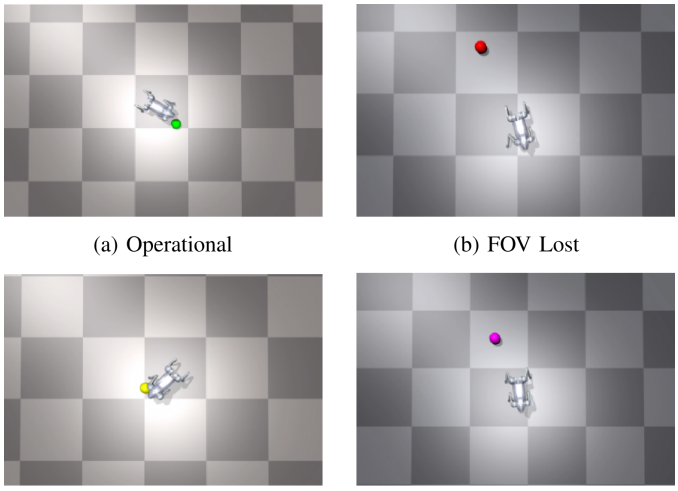
$$\begin{aligned} Pr(z_{t+1} = \text{B} \mid z_t = \text{G}) &= p_{\text{GB}}, \\ Pr(z_{t+1} = \text{G} \mid z_t = \text{B}) &= p_{\text{BG}}. \end{aligned} \tag{19}$$

The temporal mask is then defined as $m_t^{\text{GE}} = \mathbb{I}(z_t = \text{G})$. This formulation allows us to simulate realistic, consecutive frame drops rather than independent noise.

2) Predictive State Estimation: To bridge the gaps created by these simulated failures, we implement the same Kalman Filter (KF) used in the real robot following [6]. When $m_t = 0$, the measurement update is skipped, and the policy relies on the KF's constant-velocity prediction.

3) Evaluation Results and Visualization: We conduct stress tests where the robot commands are generated by a randomly operated joystick while subject to the aforementioned perception limits. As shown in Fig. 9 and the accompanying video, we visualize the perception status using color-coded ball to verify the system's behavior under different failure modes:

- **Green (Operational):** Both geometric and temporal checks pass ($m_t^{\text{GE}} = 1, m_t^{\text{fov}} = 1$). The policy receives the corrected observation.
- **Red (FOV Lost):** The ball moves out of the camera view ($m_t^{\text{GE}} = 1, m_t^{\text{fov}} = 0$). This tests the policy's ability to handle self-occlusion or rear-side dribbling.
- **Yellow (GE Lost):** The ball is within the FOV but data is lost due to temporal burst failure ($m_t^{\text{GE}} = 0, m_t^{\text{fov}} = 1$). This evaluates robustness against communication delays or motion blur.
- **Purple (Both Lost):** The system suffers from simultaneous geometric and temporal failures ($m_t^{\text{GE}} = 0, m_t^{\text{fov}} = 0$).



(a) Operational (b) FOV Lost (c) GE Lost (d) Both Lost

Fig. 9. Example snapshots of the perception availability model induced by the FOV gate and the Gilbert–Elliott (GE) loss process.

During intervals of Red, Yellow, or Purple indicators, the policy relies entirely on the predicted state. We observe that despite frequent and sometimes prolonged occlusion (e.g., the ball remaining in the Red/Purple state while behind the robot), the robot maintains stable dribbling behavior. The policy successfully anticipates the ball's trajectory based on the KF prediction and adjusts its turning rate to re-acquire the ball into the valid FOV. These results confirm that the combination of robustness via randomized training and continuity via state estimation is sufficient to handle intermittent ball loss in dynamic dribbling tasks.

B. Robustness Analysis on Ramp Traversal Limits

We conduct a quantitative stress test using the original pre-trained policy, which is trained on up to slope with 10%, to identify the performance boundaries of our method on ramp terrains. We evaluate success rates across a wide range of slope gradients—specifically ramp-up from 0% to 30% (0°-16.7°) and ramp-down up to 45% (24.2°)—with 10 000 trials performed in simulation for each setting. As illustrated in Fig. 10, our hierarchical framework significantly extends the robustness boundary compared to baselines; notably, at a 20% slope, our policy maintains a 60% success rate on ascent and over 75% on descent, whereas baseline performance degrades to near zero. Note that ramp-up performance is limited by the dribbling skill's capacity to generate sufficient torque. Specifically, at a 25% slope, the robot struggles to push the ball against gravity, leading to timeout failures (exceeding max episode length). Conversely, ramp-down performance is constrained by the locomotion skill. At extreme slopes (e.g., 35%), the high velocity required to catch the gravitationally

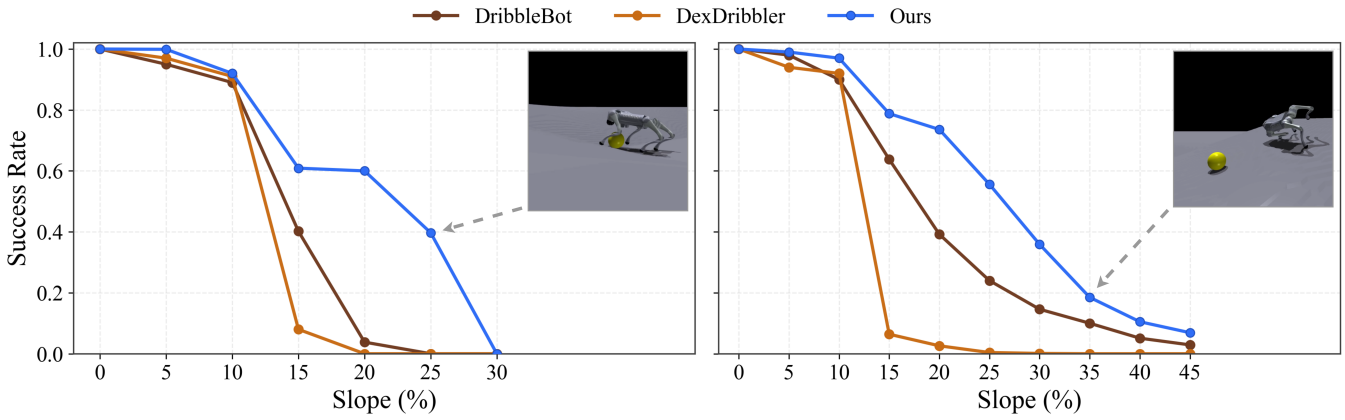


Fig. 10. **Quantitative robustness evaluation on ramp traversal limits.** Comparison of success rates on Ramp-Up (left) and Ramp-Down (right) scenarios. Insets visualize failure modes at physical limits: 1) insufficient torque to push the ball against gravity; 2) locomotion instability caused by excessive slope.

accelerated ball forces the policy into an out-of-distribution (OOD) state, resulting in loss of balance.

C. Training Process Analysis

To explain the performance disparity between DSF-PO and the *Focus-Only* baseline (masked-gradient PPO), we analyze the optimization dynamics through the lens of policy uncertainty and gradient magnitudes.

Premature Mode Collapse in Baselines As observed in Fig. 4, the *Focus-Only* baseline exhibits rapid early improvement but saturates at a suboptimal level. This transient advantage stems from aggressive parameter updates that ignore the selector's uncertainty. In the baseline setting, full-magnitude gradients are backpropagated to the active command head even when the high-level selector is effectively guessing (high entropy). This forces the policy to prematurely commit to specific skills to maximize immediate rewards, leading to premature mode collapse. Consequently, the policy loses the exploration capacity required to master complex transitions, getting trapped in local optima.

Confidence-Aware Weighting and Theoretical Justification In contrast, DSF-PO introduces a confidence-aware weighting mechanism via the skill probability term $w_{d_t}(s_t)$. This acts as a "soft hand-off": when the high-level policy is uncertain, the gradient update to the continuous command head is scaled down. This preserves skill diversity (higher entropy) throughout training, preventing the "thrashing" observed in the baseline.

This mechanism is grounded in the theoretical decomposition of the joint policy's KL divergence. For a hierarchical policy $\pi(a|s)$ composing a discrete selector π^d and continuous command heads π^c, the KL divergence decomposes as:

$$D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{old}}) = D_{\mathrm{KL}}\big(\pi^d_\theta \,\|\, \pi^d_{\mathrm{old}}\big) + \sum_{k=1}^{K} w_k(s)\, D_{\mathrm{KL}}\Big(\pi^c_\theta(\cdot \mid s,k) \,\|\, \pi^c_{\mathrm{old}}(\cdot \mid s,k)\Big). \tag{20}$$

Eq. (20) reveals that to respect the trust region of the joint policy, the optimization of a skill's command distribution must be weighted by its usage probability $w_k(s)$. The *Focus-Only* baseline violates this formulation by removing the weighting term, implicitly optimizing an objective inconsistent with the true policy distribution. DSF-PO restores this consistency, ensuring that skills specialize only when the high-level policy confidently commits to them.

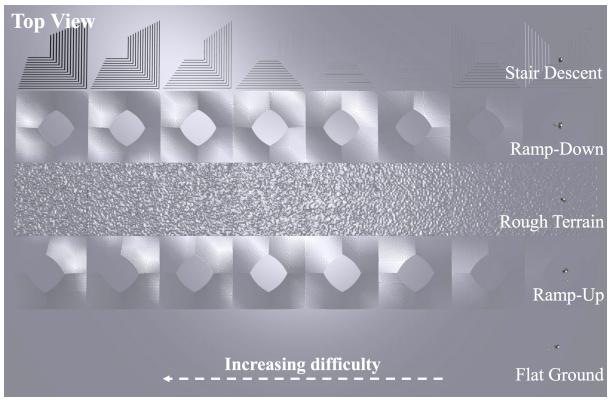


Fig. 11. **Top-view visualization of the progressive terrain curriculum.** From right to left, the curriculum difficulty increases across terrain segments including flat ground, ramp-up, ramp-down, rugged terrain, and stair descent. The same curriculum layout is used for both our method and the baseline to ensure a fair comparison.

D. Performance Analysis on the End-to-End Baseline

We implement an enhanced monolithic baseline based on DribbleBot, denoted as **DribbleBot-Terrain**, which is explicitly trained for rugged terrains. To ensure a rigorous comparison, this baseline incorporates two key enhancements over the original implementation:

- **Enhanced Terrain Rewards:** As detailed in Table V, we augment the reward function with three specialized terms to enhance robustness on rugged terrain: a *Swing Foot Clearance Penalty* to enforce high-stepping behaviors critical for stair climbing, a *Terrain-aware Speed Penalty* to restrict unsafe velocities on irregular surfaces, and a *Roll/Pitch Angular Velocity Penalty* to minimize body oscillation and prevent tipping on steep slopes.

TABLE V
REWARD TERMS OF DRIBBLEBOT-TERRAIN

Original Reward Terms		
Term	Expression	Weight
Projected Ball Velocity	$\exp\left(-\delta_v \|\mathbf{v}^b - \mathbf{v}^{\text{cmd}}\|^2\right)$	0.5
Robot Ball Distance	$\exp\left(-\delta_p \|\mathbf{b} - \mathbf{p}_{\text{FRHip}}\|^2\right)$	4.0
Yaw Alignment	$\exp\left(-\delta_\psi (e^2_{\text{rbcmd}} + e^2_{\text{rbbase}})\right)$	4.0
Ball Velocity Norm	$\exp\left(-\delta_n (\|\mathbf{v}^{\text{cmd}}\| - \|\mathbf{v}^b\|)^2\right)$	4.0
Ball Velocity Angle	$1 - (\psi_b - \psi_{\text{cmd}})^2 / \pi^2$	4.0
Swing Phase Schedule	$(1-\boldsymbol{\kappa}) \exp\left(-\delta_{\text{cf}} \|\mathbf{f}^{\text{foot}}\|^2\right)$	4.0
Stance Phase Schedule	$\boldsymbol{\kappa} \exp\left(-\delta_{\text{cv}} \|\mathbf{v}^{\text{foot}}_{\text{xy}}\|^2\right)$	4.0
Joint Limit Violation	$\mathbb{1}_{q_i > q_{\max} \vee q_i < q_{\min}}$	-10.0
Joint Position Deviation	$\|\mathbf{q} - \mathbf{q}^{\text{default}}\|^2$	-0.05
Joint Torque	$\|\boldsymbol{\tau}\|^2$	-0.0001
Joint Velocity	$\|\dot{\mathbf{q}}\|$	-0.0001
Joint Acceleration	$\|\ddot{\mathbf{q}}\|$	-2.5e-7
Hip/Thigh Collision	$\mathbb{1}_{\text{collision}}$	-5.0
Projected Gravity	$\|\mathbf{g}_{\text{xy}}\|^2$	-5.0
Action Smoothing	$\|\mathbf{a}_{t-1} - \mathbf{a}_t\|^2$	-0.1
Action Smoothing 2	$\|\mathbf{a}_{t-2} - 2\mathbf{a}_{t-1} + \mathbf{a}_t\|^2$	-0.1
Additional Reward Terms		
Term	Expression	Weight
Swing Foot Clearance Penalty	$\frac{1}{N_{\text{swing}}} \sum_{i \in \text{swing}} [h_{\text{tgt}} - h_i]^2_+$	-8.0
Speed Penalty	$\mathbb{I}_{\|\mathbf{v}_{\text{cmd}}\| > v_g} \cdot [v_{\text{obs}} - \bar{v}]^2_+$	-1.5
Roll/Pitch Angular Velocity Penalty	$\|\boldsymbol{\omega}_{xy}\|^2$	-0.2

- **Adaptive Terrain Curriculum:** To rule out optimization difficulties caused by sparse rewards, we utilize the exact same progressive terrain curriculum as our HRL method, ensuring the baseline is gradually exposed to increasing difficulty levels (visualized in Fig. 11).

E. Modality-Skill Consistency Analysis

To provide a more granular analysis of the high-level policy's decision-making logic, we assess whether the high-level skill selection aligns with the contact modality indicated by the impulse-based contact flag $i_t \in \{0, 1\}$: dribbling skills $\mathcal{D} = \{1, 2\}$ are expected under contact (i_t=1) and locomotion skills $\mathcal{L} = \{3, 4\}$ are expected under non-contact (i_t=0).

From the trajectory in Fig. 7, we construct a confusion matrix between the contact label i_t and the predicted skill category $d_t \in \{\mathcal{D}, \mathcal{L}\}$. The matrix is row-normalized and summarized by two scalar metrics reported in Table VI: balanced accuracy (BAcc) and Matthews correlation (MCC). BAcc captures average correctness under class imbalance (good if ≥ 0.80, very strong if ≥ 0.90), while MCC (-1 to 1) reflects overall agreement (strong if ≥ 0.70). In our data, $\text{BAcc} = 0.866$ and $\text{MCC} = 0.727$, indicating strong, well-balanced alignment: the selector reliably switches to $\mathcal{D}$ upon contact and maintains $\mathcal{L}$ otherwise.

APPENDIX B
DEPLOYMENT DETAILS

Hardware and Inference: Our system is deployed on a Unitree Go2 quadruped robot. A downward-facing fisheye camera with a 240° field of view is mounted on the head for ball tracking, which replaces the built-in head LiDAR. All neural network inference and vision processing run onboard on an NVIDIA Jetson Orin NX. Policies are transferred to the physical robot in a zero-shot manner.

TABLE VI
ROW-NORMALIZED CONFUSION MATRIX AND METRICS OF MODALITY–SKILL CONSISTENCY

Actual i_t	**Predicted** d_t		BAcc	MCC
	$\mathcal{D}$ (dribbling)	$\mathcal{L}$ (locomotion)		
1 (contact)	0.897	0.103	0.866	0.727
0 (no contact)	0.166	0.834		

Vision System: We utilize a YOLOv11 detector to identify the ball in the fisheye frames. Pixel coordinates are converted to metric distances using the equidistant fisheye model $r = f\theta$. To ensure robust 2D ball localization, we employ two geometric solvers (one based on apparent diameter and another on the angle–range relationship) and fuse their outputs using a Kalman filter.

Low-level Control: The joint-level control is implemented via a PD controller. For most tasks, we set k_p=20 and k_d=0.5. However, for the rough-terrain locomotion skill π^L_4, the gains are increased to k_p=40 and k_d=1.0 to enhance passability and torque response on irregular surfaces.

APPENDIX C
NETWORK IMPLEMENTATION DETAILS

The actor network architecture consists of the following components:

- **Feature Extractor:** A three-layer MLP with $[512, 256, 128]$ units and ELU activations.
- **Skill Head π^d:** A linear layer followed by a *softmax* function, producing a 4D categorical distribution for sampling the skill index d_t.
- **Command Head π^c:** A linear layer with *tanh* activation that outputs the mean $\mu_t \in (-1, 1)$ of a 5D normal distribution $\mathcal{N}(\mu_t, \Sigma)$, from which the low-level commands $\mathbf{c}^L_t$ are sampled. The first two dimensions correspond to dribbling commands, and the remaining three correspond to locomotion commands.