MAPL: Multi-Objective Preference Learning for Robot Locomotion

Xiyue Chen[1], Muhan Lin[2], Shuyang Shi[3] and Joseph Campbell[2]

Abstract— **Reward design remains a major bottleneck in reinforcement learning for robot locomotion, where successful policies often depend on carefully tuned, task-specific reward functions. Preference-based reinforcement learning offers an alternative, but existing LLM-based methods typically ask for a single overall judgment between behaviors, making it difficult to capture the multiple competing objectives that underlie high-quality locomotion. We present Multi-Objective AI-Informed Preference Learning (MAPL), a framework that learns locomotion rewards from high-level natural language objectives rather than manually engineered reward equations. MAPL prompts a large language model to compare trajectories independently along semantically meaningful criteria, using generic language descriptions that are terrain-invariant and require little domain expertise. These objective-wise preferences are used to train a multi-head preference scoring model, whose outputs are aggregated to form a scalar reward for policy optimization. Across four quadruped locomotion environments, MAPL trains policies using only LLM-generated preferences and achieves performance comparable to or better than expert-designed rewards, while eliminating task-specific reward engineering.**

I. Introduction

Reinforcement learning (RL) has led to major advances in legged locomotion, enabling quadruped robots to learn agile and robust behaviors across challenging terrain [1]–[4]. In practice, however, these successes still depend heavily on carefully engineered reward functions. Designing such rewards is often tedious and brittle: practitioners must manually balance many competing objectives, such as command tracking and stability, and small misspecifications can lead to poor local optima or reward hacking [5]. As a result, reward design remains one of the largest practical barriers to applying RL for complex robot locomotion.

Preference-based learning (PbRL) offers an alternative less reliant on domain-knowledge. Rather than requiring the designer to fully specify a reward function, human judgments are used to identify preferential behaviors [6]–[8]. Recent work replaces human annotators with LLMs, using them to either directly generate reward code [9]–[11] or to produce pair-wise rankings over behaviors which are used to train reward a model [12]–[14]. This is a promising direction as it reduces human effort and opens the possibility of specifying desired behavior in natural language.

However, existing LLM-based preference learning methods typically require the model to produce a single overall ranking between behaviors. This makes it difficult to elicit reliable preferences for locomotion, since the LLM must implicitly trade off several distinct behavioral criteria within a single judgment. In this work, we show that LLM-based reward learning becomes substantially more effective when these criteria are ranked and modeled separately.

We propose **M**ulti-Objective **A**I-Informed **P**reference **L**earning (MAPL), **a framework that learns rewards from LLM feedback over multiple, decomposed objectives.** MAPL queries an LLM to rank trajectories independently according to three objectives: velocity tracking, smoothness, and stability. Crucially, these objectives are specified entirely in terms of *high-level natural language*. The resulting rankings are then used to train a model to infer rewards for each objective independently, after which they are linearly combined to yield a single scalar reward signal. This decomposition allows the LLM to evaluate each behavioral criterion separately, rather than requiring a single global judgment that implicitly mixes several trade-offs at once. In practice, this produces more reliable supervision and more optimal downstream policies, especially when the LLM is more accurate on some criteria than others. We evaluate our method on quadruped locomotion across four environments: flat ground, uneven terrain, stairs, and obstacle traversal. Our results show that MAPL can train policies without *any* environment reward, using only the reward model learned from LLM-generated preference rankings, and achieves performance equal to or better than policies trained with expert-designed rewards. Moreover, a *single generic prompt* is used to train optimal policies across all four terrain types, completely eliminating the task-specific reward engineering and domain expertise normally required. These results suggest that LLM-based reward learning can move beyond coarse preference supervision and become a practical tool for scalable locomotion training.

II. Related Work

A. Reward Design for Legged Robot

RL enables robots to acquire robust and agile behaviors through trial-and-error optimization guided by reward functions [1]–[4]. However, the reliance on expert prior knowledge and carefully engineered reward terms significantly limit the efficiency of RL in robot learning. Recent studies have explored leveraging LLMs to automatically generate reward functions for robot tasks [9]–[11]. Nevertheless, these methods need to deal with LLM hallucination and its limited concepts about numerical reward values. Therefore, these approaches still require iterative reward function revision by LLMs and substantial manual tuning to achieve reliable performance in complex robot tasks.

[1] XC is with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, USA chen4513@purdue.edu

[2] ML and JC is with the Department of Computer Science, Purdue University, West Lafayette, IN 47906, USA lin2265, joecamp@purdue.edu

[3] independent researcher

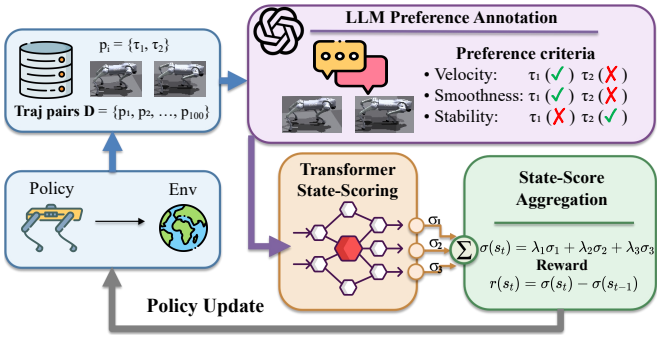

Fig. 1: MAPL consists of two iterative steps. **Scoring Model Training:** (1) Sample trajectory pairs from replay buffer. (2) Query LLM to obtain multi-objective pairwise preferences. (3) Train a transformer-based preference scoring model for each objective. **Policy Training:** (1) Sample state-action pairs by rolling out policy. (2) Compute scores for each state using the preference scoring model. (3) Aggregate scores to compute the potential-difference reward. (4) Update the policy using state-action-reward triplets.

B. Preference-based RL in Robot Learning

Preference-based RL aims to alleviate the challenges of hand-crafted reward design by learning reward functions from human preferences over pairwise trajectory comparisons [6], [13], [15], [16]. Rather than specifying explicit reward terms, RL from human feedback (RLHF) trains parametrized reward functions that better align with human notions of task success [6]. While preference labels require extensive human efforts, more recent work, represented by RL from AI feedback (RLAIF), replaces human annotators with pretrained models to provide scalable, automated feedback. [17]–[19] Some works start to apply this framework to robot learning. For robot manipulation tasks, PrefCLM and RL-SaLLM-F [20], [21] prompt LLMs to generate preference over state–action trajectory pairs, while RL-VLM-F [14], [22] employs vision–language models (VLMs) to provide preferences over visual observations of states.

Although RLAIF has achieved impressive results in robot manipulation, the application of PbRL to robot locomotion remains relatively unexplored because legged locomotion require optimizing multiple objectives simultaneously. LAPP [23] extends RLAIF to robot locomotion, achieving faster convergence and improving overall performance. However, it still relies on the summation of trained rewards and carefully engineered rewards to bootstrap and stabilize policy training.

DAPPER [24] introduces a VLM-based framework to promote stable locomotion through automated trajectory evaluation and active learning to reduce annotation cost. Nevertheless, VLM-based feedbacks are still expensive to acquire and struggles to capture certain continuous control objectives, such as precise velocity tracking, particularly in robot systems where single-frame visual observations provide limited dynamic information.

In contrast, MAPL addresses these limitations by integrating multiple pretrained preference-scoring models, each specializing in distinct locomotion aspects such as velocity tracking, stability, and motion smoothness. This structured multi-objective feedback formulation enables reward learning for subtle policy details while reducing dependence on hand-crafted rewards.

Algorithm 1 MAPL

Require: Buffer B, Preference Scoring Model σ_ψ, Policy π_θ, Scoring Model Update Frequency T_P

Populate B with L initial trajectories sampled from π
for $t = 1, 2, \ldots$ **do**
  if $t \% T_P == 0$ **then**
    Collect trajectories $\mathcal{D}$ sampled from π
    Generate preference rankings for each $\tau \in \mathcal{D}$
    Update scoring model parameters ψ using Eq. 4
  end if
  Rollout policy for one step to obtain s_t, a_{t-1}, s_{t-1}
  Calculate reward, $r_t = \sigma_\psi(s_t) - \sigma_\psi(s_{t-1})$
  Update policy parameters θ
end for

III. Preliminaries

We consider the standard Markov Decision Process (MDP) reinforcement learning setting, where an agent sequentially interacts with an environment [25]. At each timestep t, the agent receives an observation o_t from the current state s_t and selects an action a_t according to the policy π. After executing the action, the environment returns a reward r_t and transitions to the next state s_{t+1}. The objective of the agent is to maximize the expected return, defined as the discounted sum of rewards:

$$R = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k), \tag{1}$$

where γ is the discount factor.

Our work builds upon the PbRL framework, where the agent learns a reward function from preference feedback instead of relying on handcrafted reward signals. In traditional PbRL, a trajectory τ is defined as a sequence of states:

$$\tau = \{(s_t, a_t), (s_{t+1}, a_{t+1}), \ldots, (s_H, a_H)\}, \quad H \geq 1. \tag{2}$$

During training, pairs of trajectories are sampled, and human annotators or AI models provide preference labels over trajectory pairs. The collected preference signals are then used to train a reward model that outputs scalar rewards, effectively replacing the environment reward. In our method, we sample trajectory pairs (τ^0, τ^1) during training, where each trajectory is downsampled so that it contains M evenly-distributed states. This allows us to capture long-horizon trends within a compact trajectory. The preference label is defined as

$$y = \begin{cases} 0, & \tau^0 \succ \tau^1, \\ 1, & \tau^1 \succ \tau^0, \\ 0.5, & \tau^0 \sim \tau^1. \end{cases}$$

We follow the Bradley–Terry model [26] to compute the likelihood of $\tau^1 \succ \tau^0$ for a pair of trajectories:

$$P_\psi(\tau^1 \succ \tau^0) = \frac{\exp\left(\sum_{t=1}^{H} \sigma_\psi(s_t^1)\right)}{\sum_{i\in\{0,1\}} \exp\left(\sum_{t=1}^{H} \sigma_\psi(s_t^i)\right)}. \tag{3}$$

We train a transformer-based reward model [27] by minimizing the pairwise loss

$$\mathcal{L}^{(i)} = -\mathbb{E}_{(\tau^0,\tau^1,y)\sim\mathcal{D}}\Big[\mathbb{I}\{y=(\tau^0\succ\tau^1)\}\log P_\psi[\tau^0\succ\tau^1] + \mathbb{I}\{y=(\tau^1\succ\tau^0)\}\log P_\psi[\tau^1\succ\tau^0]\Big]. \tag{4}$$

We denote the output as the *preference score*, $\sigma_\psi(s_t)$.

IV. METHODOLOGY

Figure 1 provides an overview of MAPL, which consists of two iterative steps, *preference scoring model training* and *policy training*, which operate at different frequencies. The full MAPL framework is shown in Alg. 1.

A. Locomotion Objectives

First, we consider a generic quadruped locomotion task that requires balancing multiple behavioral objectives under dynamic and contact-rich interactions. We identify three semantically grounded and task-agnostic criteria that are sufficient to characterize high-quality locomotion across diverse terrains: velocity tracking, stability, and motion smoothness. We adopt the same three criteria across all terrain types without any task-specific modifications. These criteria are defined entirely in natural language, and do not contain any domain-specific engineering. Definitions are given below.

Velocity Tracking. Measures how accurately the robot executes commanded linear and angular velocities.

> *Prompt*: At each timestep, compare the robot's actual x–y linear velocity to the commanded x–y velocity. Smaller deviations correspond to better tracking, and preference should decrease smoothly as deviations increase. Larger errors should reduce preference more strongly, but no single timestep should completely invalidate a trajectory.

Stability. Measures how well the robot is able to maintain an upright position, avoiding excessive roll and pitch.

> *Prompt*: The base height should stay very close to `height` throughout the trajectory. Even small deviations should noticeably reduce preference, and larger deviations should rapidly make the trajectory much worse, regardless of other factors. The robot base should exhibit minimal vertical motion. Roll and pitch angular velocities should remain small and smooth.

Smoothness. Measures the smoothness of control signals.

> *Prompt*: We prefer the trajectory with smaller overall action difference. Trajectories with fewer or weaker stumble events are strictly preferred over those with more frequent or more severe stumbles. If one trajectory exhibits clear stumble behavior while the other does not, the non-stumbling trajectory should be preferred, even if its action smoothness is slightly worse.

B. Multi-Objective Preference Generation

Existing RLAIF approaches collapse multiple behavioral objectives into a single preference ranking. However, this forces the LLM to reason over multiple criteria simultaneously, leading to complex prompt engineering, noisy supervision signals, and gradient interference, especially when some objectives are more reliably estimated than others.

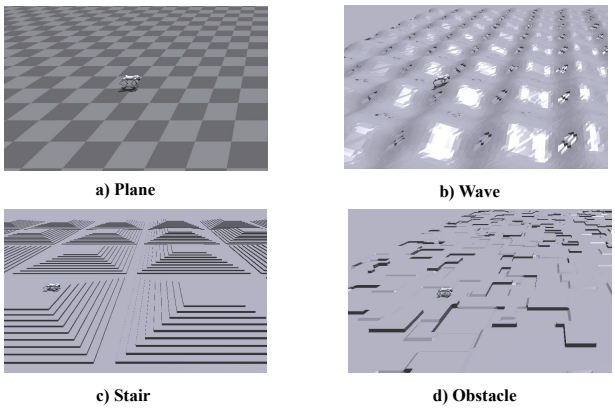

Fig. 2: Evaluation terrains where the quadratic robots are trained to walk.

Our work is based on the insight that LLMs elicit more accurate preference rankings if each objective is ranked and modeled independently. To that end, we define a prompt in which the LLM produces a separate preference ranking for each objective defined above. As specified in Sec. III, the prompt is given a complete trajectory which has been downsampled to consist of M evenly-distributed states. To account for stochasticity and better capture the mode of the LLM sampling distribution, we query preference rankings multiple times and take the majority answer.

C. Preference Scoring Model Training

In the preference scoring model training step, we train a transformer-based preference scoring model σ, parameterized by ψ, consisting of shared parameters and a separate output head for each objective, where the i-th head produces a preference score $\sigma_\psi^{(i)}(s)$, where $i \in \{\text{velocity}, \text{stability}, \text{smoothness}\}$. We adopt a pairwise ranking loss following the Bradley–Terry formulation in Equation 3. The scoring model is updated at a slower frequency than the policy—in our experiments $100\times$ slower—such that we avoid computationally expensive LLM calls until the policy's state visitation distribution has sufficiently evolved.

D. Policy Training

In the policy training step, state-action pairs are sampled by rolling out the policy in the environment. Overall preference scores are computed for each state as a linear combination of per-objective preference scores,

$$\sigma_\psi(s_t) = \sum_{i=1}^{N} \lambda_i \, \sigma_\psi^{(i)}(s_t), \tag{5}$$

where λ is a coefficient determining the relative weight of each objective. This formulation allows us to control objective balancing through a low-dimensional weight vector rather than through fine-tuning a high-dimensional reward

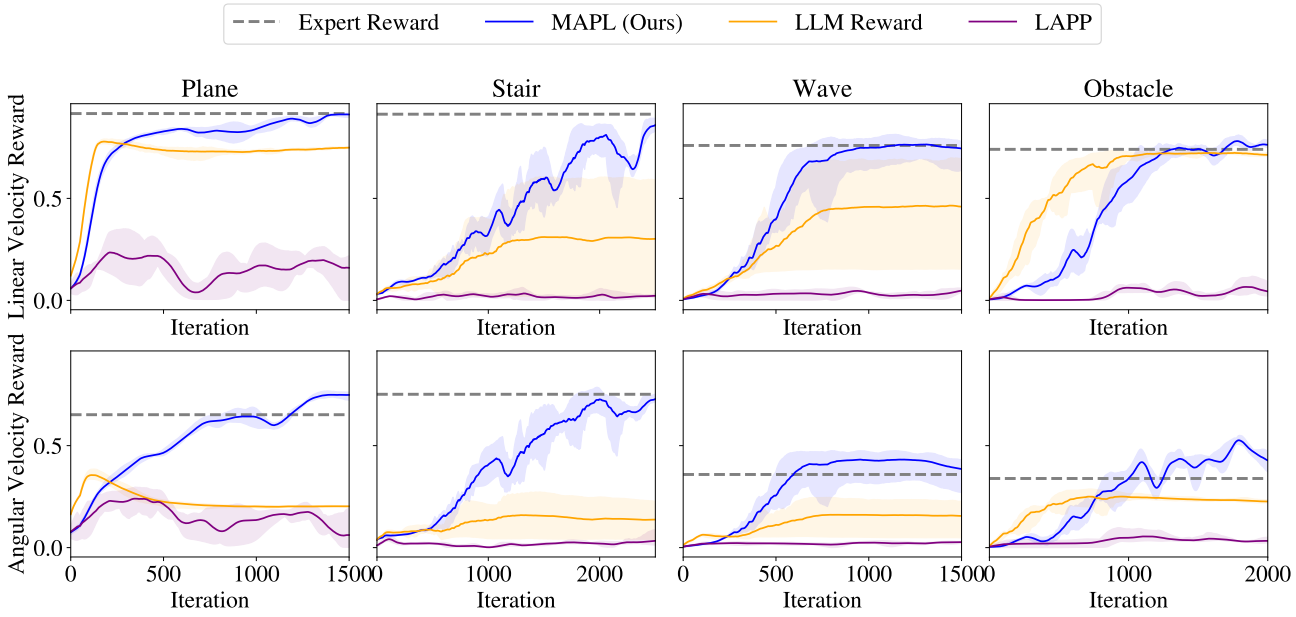

Fig. 3: Velocity training curves of MAPL and baseline methods across diverse terrains. All results are averaged over five independent random seeds. The solid lines indicate the mean performance, while the shaded regions represent the standard error across runs. A moving average with a window size of 100 is applied to improve visualization clarity of the learning trends.

model. This design decouples reward representation learning from objective balancing and improves training stability. While λ can be automatically computed by optimizing the policy's expected reward over a set of rollouts, in practice we treat it as an additional hyperparameter that is manually specified.

We prioritize velocity tracking and stability by assigning them large weights. By contrast, smoothness is terrain-dependent. In simple terrain types such as a flat plane, we assign a small weight value to avoid overly restrictive motion regularization. In more challenging terrain such as stairs, we assign a larger weight to avoid undesirable events such as stumbling and abrupt contact transitions.

Rather than directly using the preference score as the reward signal for policy optimization, as in prior works [28], [29] we adopt a potential difference formulation as the final reward to improve sample efficiency and policy return:

$$r(s_t) = \sigma_\psi(s_t) - \sigma_\psi(s_{t-1}). \tag{6}$$

V. Experiments and Results

A. Experiment Setup

We evaluate MAPL by comparing downstream policy performance to baseline approaches, where the goal is to train a legged quadruped to run at a specified velocity across varied terrain.

Environments. Our environments are implemented based on the Legged-Gym framework, which provides standardized terrain generation and locomotion task configurations [30]. Evaluation terrains include a flat plane, hills, pyramidal stairs, and randomly-sized obstacles. We conduct all experiments on the Unitree Go2 quadruped robot using the Isaac Gym simulator. See Figure 2 for the visualization of these simulated environments.

Metrics. We measure performance with two metrics:

- **Linear Velocity Reward**.

$$r^{v_{xy}} = \exp\left(-\frac{\left\|\mathbf{v}_{xy}^{\text{cmd}} - \mathbf{v}_{xy}^{\text{base}}\right\|_2^2}{\sigma_v}\right) \tag{7}$$

- **Angular Velocity Reward**.

$$r^{v_\theta} = \exp\left(-\frac{\left(v_\theta^{\text{cmd}} - v_\theta^{\text{base}}\right)^2}{\sigma_\theta}\right) \tag{8}$$

The parameters $\mathbf{v}^{\text{cmd}}$ denote the desired velocities, $\mathbf{v}^{\text{base}}$ represent the measured base velocities of the quadruped robot; σ_v and σ_θ are scaling hyperparameters that control the tolerance to velocity tracking errors. In all environments, both σ are set up to be 0.25. Both rewards are Gaussian-shaped tracking terms encouraging accurate velocity control for locomotion. They issue higher values when the measured linear or angular velocity of the robot closely matches the commanded velocity, and exponentially penalize deviations.

Curriculum and Difficulty-Weighted Evaluation. For uneven terrains (e.g., stairs, waves, and obstacles), we adopt a curriculum learning strategy. The terrain difficulty is gradually increased only after the policy reaches a predefined success threshold, indicating stable locomotion under the current difficulty level. To ensure fair comparison across baselines trained under varying terrain difficulties, we report a difficulty-weighted tracking reward during evaluation. Specifically, the velocity reward is scaled proportionally to

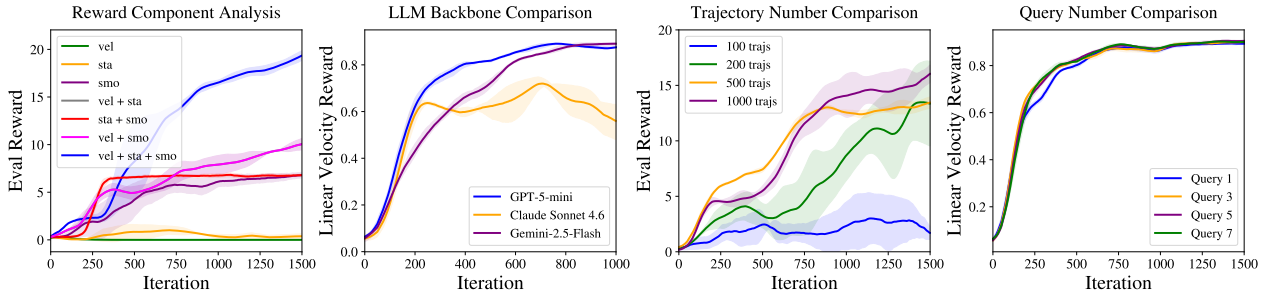

Fig. 4: All results are averaged over three independent random seeds for the ablation studies. Solid lines denote the mean training performance, and shaded regions indicate the standard error across runs. We conduct 4 abolation studies which includes: 1) Contribution analysis of each reward component in the MAPL framework to evaluated locomotion performance where vel, sta, and smo denote velocity, stability, and smoothness, respectively. 2) Comparison of different large language model backbones for velocity tracking within MAPL. 3) Sensitivity analysis with respect to the size of the trajectory replay buffer. 4) Sensitivity analysis with respect to the number of LLM queries per trajectory ranking. Eval reward represents expert reward values.

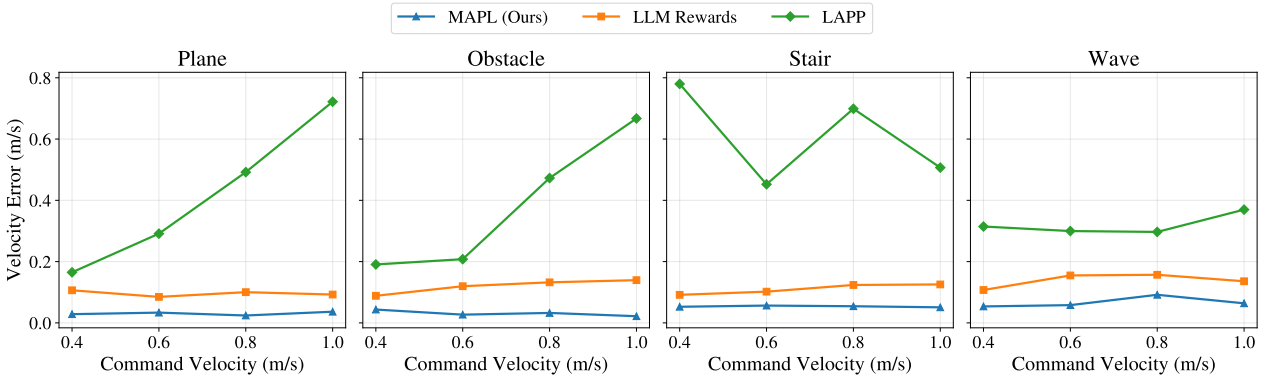

Fig. 5: Velocity tracking error of the trained policies across different methods. The error is computed under given forward command velocities, where velocities are averaged over 20-step intervals during a 500-step rollout.

Term	Equation	Weight
$r^{cmd}_{v_{xy}}$: v_{xy} tracking	$e^{\left(-\frac{\|\mathbf{v}_{xy}-\mathbf{v}^{cmd}_{xy}\|^2}{\sigma_v}\right)}$	1.0
$r^{cmd}_{\omega_z}$: yaw tracking	$e^{\left(-\frac{(\omega_z-\omega^{cmd}_z)^2}{\sigma_\omega}\right)}$	0.5
z velocity	v_z^2	-2.0
roll-pitch velocity	$\|\boldsymbol{\omega}_{xy}\|^2$	$-5e^{-2}$
joint torques	$\|\tau\|^2$	$-1e^{-5}$
joint accelerations	$\|\ddot{q}\|^2$	$-2.5e^{-7}$
thigh/calf collision	$1_{\text{collision}}$	-1.0
joint limit violation	$1_{q_i>q_{max}\,\|\|\,q_i<q_{min}}$	-10
feet air time	$\sum_{foot} t_{air}$	1.0
action smoothing	$\|a_{t-1}-a_t\|^2$	$-1e^{-2}$

TABLE I: The hand-designed expert reward function consists of a linear combination of the above terms.

the normalized difficulty level:

$$\tilde{r} = r \cdot \frac{d+1}{D_{\max}}, \tag{9}$$

where d denotes the current terrain difficulty level, $D_{\max}$ is the maximum difficulty level, and r represents the raw velocity tracking reward. This scaling ensures that policies capable of handling higher-difficulty terrains receive proportionally higher evaluation rewards. When the highest difficulty level is reached ($d = D_{\max} - 1$), the scaling factor becomes 1, and no additional weighting is applied.

Baselines. We compare against the following baselines:

- **LAPP** [23]. Representative preference learning method based on a single preference signal. It employs a transformer-based reward model and uses carefully designed prompts with in-context examples to guide the LLM toward producing accurate feedback. Unlike the original version, for fair comparison we use a variant which does *not* incorporate the standard environment reward.
- **LLM Reward**. Representative code-based baseline, in which an LLM is prompted to generate code for a reward function given the same prompts as MAPL.
- **Expert Reward**. Hand-designed reward function created by a domain expert, defined in Table I. Represents upper-bound performance in policy training.

All baselines use the PPO-based [31] RSL-RL as the underlying reinforcement learning algorithm, following the network architecture and hyperparameter settings reported in prior work [32]. For the preference aggregation of MAPL, we set $\lambda_{\text{velocity}}, \lambda_{\text{stability}}, \lambda_{\text{smoothness}}$ ratio to be 1.5, 1.0, and 0.1 for flat and wave terrains, and 1.5, 1.0, 0.25 for stair and obstacle terrains. Unless specified otherwise, all rankings are produced with GPT-5-mini.

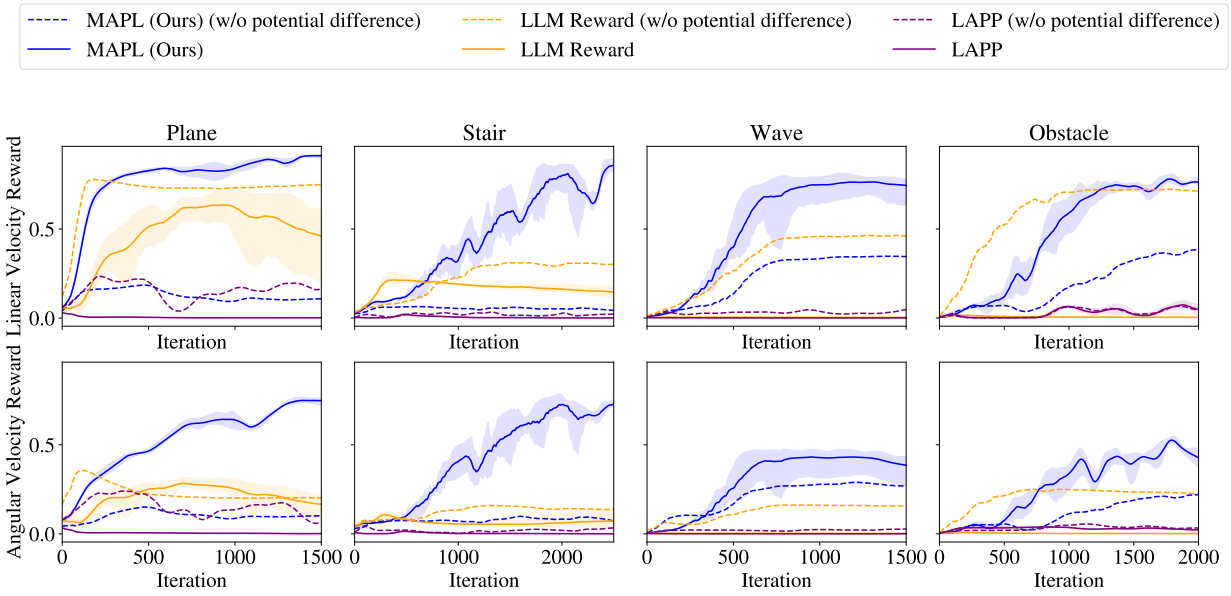

Fig. 6: Velocity learning curves of all methods with and without the potential difference. The results are running for 5 seeds, and the moving average with window size 100 are applied in the figure to improve the readability.

B. Locomotion Policy Performance

We next evaluate how the proposed reward modeling approach performs compared to baselines with GPT-5-mini in all environments. We collected 500 trajectory pairs for training each preference-based reward model. Each rewarding method is employed to train 5 RL policies with random seeds and initializations for each method. Training performance is measured by the residual error between commanded and actual velocities. Figure 3 shows the resulting learning curves. The dashed gray line denotes the PPO baseline with well-tuned rewards, which is expected to be the upper-bound performance.

Across all terrains, our method (blue) consistently achieves higher final tracking rewards while LLM reward plateaus at lower returns, even if the policy training speed is limited by multi-criterion rewards and sometimes becomes lower than that of LLM Rewards. Without the assistance of well-tuned rewards, LAPP rewards alone fail in learning useful policies. Our trained reward models reach the converging policy returns of well-tune expert rewards in all scenarios and even outperform them over some terrains.

C. Preference Component Ablation

To understand the contribution and necessity of each preference component in the reward assembly, we conduct an ablation study. Keeping other settings unchanged, we train locomotion policies using rewards constructed from different subsets of the proposed preference terms, velocity, stability, and smoothness, as well as their combinations. Each reward model is trained using rankings over the same 2000 trajectory pairs to ensure convergence under diverse preference objectives. Figure 4 compares the learning curves over training iterations.

Without the robustness requirements, velocity preference alone fails to produce meaningful locomotion policies, whose performance remains near zero. The stability-only variant controls the legged robot slightly better, but fails to achieve obvious returns. The smoothness-only variant helps the policy capture more details but still converge at low returns.

Pairwise combinations significantly improve learning but remain inoptimal. Since the smoothness-only preference independently learns meaningful policies, it is reasonable to observe that combining it with other preference scores yields the greatest improvement in policy training compared to using any single preference alone. Compared to the returns with smoothness-only preference, adding velocity variant to measure the locomotion outcome increases training speed and returns simultaneously. Adding stability preference to the smoothness one boosts the learning speed significantly, but the conservative strategy leads to similar converging returns. Without the information from smoothness preference, velocity + stability fails in leading to any improvement.

The full reward achieves the best performance, nearly 2× higher than the best two-term variant with reduced variance. velocity drives forward motion, stability enforces physically robust body dynamics, and smoothness mitigates control oscillations. Removing any component disrupts this balance, degrading both convergence speed and returns. These results validate our multi-objective preference-based reward design.

D. Trajectory and Query Sensitivity

We further investigate the sensitivity of our method to the size of the trajectory buffer in the plane environment (top-left in Fig. 4) while keeping all other settings the same. While MAPL rewards are used to train policies, ”Eval Rewards” here used to measure policy training returns is the

expert reward function. As shown in the figure, using 1000 trajectories yields the highest evaluation reward. When the trajectory number is reduced to 500, the performance remains competitive, implying the sample efficiency of MAPL. We also study the query number of one trajectory ranking required by MAPL with GPT-5-mini (bottom-right in Fig. 4) over the Plane terrain while keeping other settings the same. Due to hallucination, an LLM may produce inconsistent responses to repeated queries of the same question when it is uncertain, which constitutes a critical source of errors. A common mitigation strategy is to query the LLM multiple times and use majority voting as the final answer. By comparing training results under different numbers of queries per trajectory ranking, we evaluate the robustness of MAPL to such LLM-induced ranking errors. The results show that the policy trained with a single query per ranking converges to nearly the same return within almost the same training steps as policies trained with multiple queries, although its early-stage learning is slightly slower. When the number of queries is greater than or equal to three, the learning curves become almost identical, where increasing the number of queries yields no noticeable performance improvement. These findings indicate that MAPL is robust to LLM inconsistent ranking answers and does not require a large number of queries per ranking.

E. Potential Difference Reward Modeling

To further boost the training speed and final returns of locomotion policies, MAPL has utilized the technique of potential difference. To demonstrate its effectiveness, Figure 6 compares training performance of all methods with and without potential-difference shaping across four locomotion tasks while keeping all other settings same as previous sections. Across all environments, removing the potential difference from MAPL consistently results in substantial degradation in both final performance and convergence speed. In contrast, incorporating potential difference enables MAPL to outperform all baselines, as detailed in Section V-B. Notably, integrating potential difference into LLM Reward or LAPP often leads to early plateauing or unstable training. This suggests that these methods fail to utilize potential-based shaping, likely due to poorly structured rewards induced by LLM hallucinations or single-criterion preference models.

F. Robot Performance under Different Rewards

To demonstrate the actual effects of different rewards over the robot, we compare the distributions of body height offset, roll, and pitch under different reward designs. In this experiment, 20 steps are randomly sampled by rolling out the policy trained with different rewards in 16 randomly-initialized Plane environment objectives respectively. We record the roll, pitch and robot body height at each step for each method, which are plotted in Figure 7. Given that zero height offset and near-zero roll and pitch imply stable standing, points closer to the origin indicate better robot performance. As expected, expert rewards produce a relatively concentrated distribution around the origin. However, our reward achieves

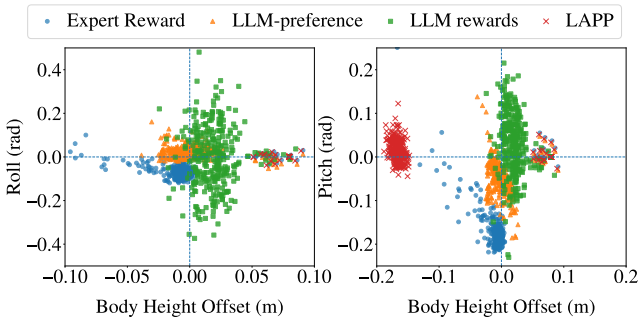

Fig. 7: The robot height, roll, pitch distribution over 20 random steps rolled out by policies trained with different rewards in 16 randomly initialized

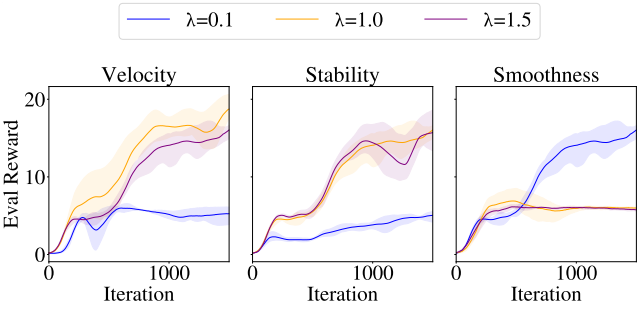

Fig. 8: Evaluation rewards under different λ settings for each preference objective. Results are averaged over three random seeds with a smoothing window of 100. The shaded areas indicate the standard error.

a tight clustering even closer to the origin, demonstrating improved stability even without expert-crafted terms and manual tuning. In contrast, LLM rewards without preference modeling exhibit higher variance due to LLM hallucination and its limited numerical concepts. Meanwhile, LAPP shows clear bias away from the origin, indicating unstable posture. This is consistent with its low training returns and ill-posed rewards without the assistance of well-tuned rewards, which are discussed in previous sections. Consistent with Section V-B, these results illustrate how our method not only matches but surpasses expert-designed rewards, LLM generated rewards and single-preference rewards. They validate the effectiveness of structured preference aggregation and potential-difference shaping for properly rewarding robust and balanced locomotion behaviors.

G. Preference Aggregation Weight Sensitivity

We analyze the sensitivity of reward quality to the aggregation weight λ for each preference criterion by comparing the resulting policy performance. As shown in Figure 8, for velocity and stability, large weights with distinct scales lead to high but similar final returns, consistent with the dominant role of these two criteria in shaping the policy. On the other hand, consistent to the fact that smoothness plays a less important role, high λ values achieve similarly low training returns. These results suggest that, the critical thing about λ setting is identifying which preference criteria have stronger or weaker influence on the policy as discussed in Section IV-D, which is intuitive. As long as more influential objectives receive relatively larger weights, preference scores from multiple aspects can be combined reliably without extensive hyperparameter tuning.

	Overall	Velocity	Stability	Smoothness
SOSQ	63%	×	×	×
MOSQ	×	67%	67%	98%
MOMQ (MAPL)	×	88%	76%	98%
LAPP	68%	×	×	×

TABLE II: Ranking accuracy of different preference-query strategies. SOSQ denotes a single prompt that combines all preference criteria and produces one overall ranking. MOSQ denotes a single prompt that outputs separate rankings for different objectives. MOMQ (MAPL) uses multiple objective-specific prompts and multiple queries to obtain separate rankings. LAPP uses a manually designed prompt to generate a single overall ranking.

H. Velocity Error Analysis

We evaluate velocity tracking performance by measuring the error between commanded and actual forward velocities under different command magnitudes. Specifically, we fix lateral and yaw commands to zero and vary the forward velocity from 0.4 to 1.0 m/s.

As shown in Fig. 5, MAPL consistently achieves the lowest velocity error across all terrains, including plane, obstacle, stair, and wave environments. Notably, MAPL maintains low velocity error as the commanded velocity increases, indicating strong generalization and stable tracking performance even under more aggressive commands.

In contrast, LLM-based reward methods have moderate velocity errors, suggesting limited capability in accurately capturing velocity tracking objectives. LAPP shows significantly larger errors and a clear degradation trend, particularly on plane and obstacle terrains where the error grows rapidly as command velocity increases. This indicates that single-objective preference signals fail to provide sufficient supervision for precise control.

I. Ranking Accuracy Analysis

To demonstrate the strength of aggregating multi-objective preference scores from trained scoring models, we analyze the ranking accuracy of different preference-query strategies by comparing their predicted rankings with the ground-truth rewards by rolling out 100 trajectory pairs. For the single-objective strategies, the ranking is evaluated using the full expert reward function as the reference. For the multi-objective strategies, the complete expert reward function is decomposed into three components corresponding to velocity, stability, and smoothness, and the ranking accuracy is evaluated separately for each objective.

As shown in Table II, our method (MOMQ), which employs multiple objective-specific prompts and multiple queries, achieves the most stable and accurate rankings across different objectives compared with the other strategies.

VI. CONCLUSIONS

In this work, we introduced MAPL, a multi-objective AI-informed preference learning framework that learns locomotion rewards aggregating multi-criterion LLM feedback rather than manually engineered reward functions. By querying an LLM to rank trajectories along velocity tracking, smoothness, and stability, MAPL learns objective-wise preference models and then combines them linearly under a potential-difference formulation to provide a stable training signal. Across four locomotion environments, including flat ground, uneven terrain, stairs, and obstacle traversal, MAPL trains policies using only LLM preference supervision and achieves performance comparable to or better than expert-designed rewards. Importantly, a single generic prompt is sufficient to train policies across all terrains, highlighting MAPL ability to reduce the need for reward engineering. Meanwhile, the reward-component ablation study and LLM trajectory ranking accuracy analysis support the necessity of multi-objective preference modeling.

Despite these significant results, several limitations remain. The current framework aggregates objectives using a linear combination, which improves interpretability but may limit expressiveness. Future work could explore nonlinear aggregation mechanisms that better capture interactions among locomotion objectives. Finally, while LLM-preference-based reward models provide scalable supervision, integrating multimodal foundation models, such as vision-language and tactile-aware models, could further enable reward learning from visual perception and physical interaction signals. These directions could potentially extend the approach to more complex robot behaviors such as loco-manipulation.

REFERENCES

[1] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[2] 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.

[3] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*. PMLR, 2023, pp. 22–31.

[4] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 572–587, 2024.

[5] J. Skalse, N. Howe, D. Krasheninnikov, and D. Krueger, "Defining and characterizing reward gaming," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9460–9471, 2022.

[6] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in neural information processing systems*, vol. 30, 2017.

[7] J. Park, Y. Seo, J. Shin, H. Lee, P. Abbeel, and K. Lee, "Surf: Semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning," *arXiv preprint arXiv:2203.10050*, 2022.

[8] K. Lee, L. Smith, and P. Abbeel, "Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training," *arXiv preprint arXiv:2106.05091*, 2021.

[9] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv:2310.12931*, 2023.

[10] Z. K. Heng, Z. Zhao, T. Wu, Y. Wang, M. Wu, Y. Wang, and H. Dong, "Boosting universal llm reward design through heuristic reward observation space evolution," *arXiv preprint arXiv:2504.07596*, 2025.

[11] Y. Zeng, Y. Mu, and L. Shao, "Learning reward for robot skills using large language models via self-alignment," *arXiv preprint arXiv:2405.07162*, 2024.

[12] X. Wang, K. Lee, K. Hakhamaneshi, P. Abbeel, and M. Laskin, "Skill preferences: Learning to extract and execute robotic skills from human feedback," in *Conference on robot learning*. PMLR, 2022, pp. 1259–1268.

[13] R. Wang, D. Zhao, Z. Yuan, T. Shao, G. Chen, D. Kao, S. Hong, and B.-C. Min, "Primt: Preference-based reinforcement learning with multimodal feedback and trajectory synthesis from foundation models," *arXiv preprint arXiv:2509.15607*, 2025.

[14] Y. Wang, Z. Sun, J. Zhang, Z. Xian, E. Biyik, D. Held, and Z. Erickson, "Rl-vlm-f: Reinforcement learning from vision language foundation model feedback," *arXiv preprint arXiv:2402.03681*, 2024.

[15] A. Wilson, A. Fern, and P. Tadepalli, "A bayesian approach for policy learning from trajectory preference queries," *Advances in neural information processing systems*, vol. 25, 2012.

[16] H. Lee, S. Phatale, H. Mansoor, T. Mesnard, J. Ferret, K. Lu, C. Bishop, E. Hall, V. Carbune, A. Rastogi *et al.*, "Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback," *arXiv preprint arXiv:2309.00267*, 2023.

[17] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon *et al.*, "Constitutional ai: Harmlessness from ai feedback," *arXiv preprint arXiv:2212.08073*, 2022.

[18] T. Yu, H. Zhang, Q. Li, Q. Xu, Y. Yao, D. Chen, X. Lu, G. Cui, Y. Dang, T. He *et al.*, "Rlaif-v: Open-source ai feedback leads to super gpt-4v trustworthiness," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 19 985–19 995.

[19] L. Li, Z. Xie, M. Li, S. Chen, P. Wang, L. Chen, Y. Yang, B. Wang, and L. Kong, "Silkie: Preference distillation for large visual language models," *arXiv preprint arXiv:2312.10665*, 2023.

[20] S. Tu, J. Sun, Q. Zhang, X. Lan, and D. Zhao, "Online preference-based reinforcement learning with self-augmented feedback from large language model," *arXiv preprint arXiv:2412.16878*, 2024.

[21] R. Wang, D. Zhao, Z. Yuan, I. Obi, and B.-C. Min, "Prefclm: Enhancing preference-based reinforcement learning with crowdsourced large language models," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2486–2493, 2025.

[22] S. Venkataraman, Y. Wang, Z. Wang, N. S. Ravie, Z. Erickson, and D. Held, "Real-world offline reinforcement learning from vision language model feedback," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 13 452–13 459.

[23] P. Jian, X. Wei, Y. Liu, S. A. Moore, M. M. Zavlanos, and B. Chen, "Lapp: Large language model feedback for preference-driven reinforcement learning," *arXiv preprint arXiv:2504.15472*, 2025.

[24] Y. Kadokawa, J. Frey, T. Miki, T. Matsubara, and M. Hutter, "Dapper: Discriminability-aware policy-to-policy preference-based reinforcement learning for query-efficient robot skill acquisition," *IEEE Robotics & Automation Magazine*, 2026.

[25] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[26] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[27] C. Kim, J. Park, J. Shin, H. Lee, P. Abbeel, and K. Lee, "Preference transformer: Modeling human preferences using transformers for rl," *arXiv preprint arXiv:2303.00957*, 2023.

[28] M. Lin, S. Shi, Y. Guo, B. Chalaki, V. Tadiparthi, E. M. Pari, S. Stepputtis, J. Campbell, and K. Sycara, "Navigating noisy feedback: Enhancing reinforcement learning with error-prone language models," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

[29] G. Swamy, C. Dann, R. Kidambi, Z. S. Wu, and A. Agarwal, "A minimaximalist approach to reinforcement learning from human feedback," *arXiv preprint arXiv:2401.04056*, 2024.

[30] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on robot learning*. PMLR, 2022, pp. 91–100.

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

[32] C. Schwarke, M. Mittal, N. Rudin, D. Hoeller, and M. Hutter, "Rsl-rl: A learning library for robotics research," *arXiv preprint arXiv:2509.10771*, 2025.

APPENDIX

A. Reward Model Hyperparameter

The hyparameter of Transformer-based reward model are tuned manually. The details are listed below.

Hyperparameter	Transformer Reward Model
Embedding Dimension	256
Number of Layers	2
Number of Heads	4
Feedforward Dimension	512
Dropout	0.1
Batch Size	32
Epochs	80
Learning Rate Schedule	$[4 \times 10^{-5}, 20]$, $[3 \times 10^{-5}, 70]$, $[8 \times 10^{-5}, 80]$

TABLE III: Training hyperparameters of the Transformer-based reward model. The learning rate schedule is presented as the learning rate value along with the corresponding epoch it is applied to.

We adopt the same PPO hyperparameter to Legged gym [30]:

Hyperparameter	Value
Batch size	98304 (4096 $\times$ 24)
Mini-batch size	24576 (4096 $\times$ 6)
Number of epochs	5
Clip range	0.2
Entropy coefficient	0.01
Discount factor γ	0.99
GAE discount factor λ	0.95
Desired KL-divergence kl^*	0.01
Learning rate α	adaptive*

TABLE IV: Training hyperparameters for PPO for all of our environment, (*) means we use the adaptive learning rate based on the KL-divergence

B. System Prompt for Reward Type

1) Velocity Preference Prompt:

You are a robotics engineer specializing in analyzing and comparing the trajectories of a Unitree Go2 Robot Dog. You will be given two trajectories of the Unitree Go2 Robot Dog, and you need to decide which trajectory is better.
A trajectory will include the following information:
1) "The base linear velocity" (m/s): Current Unitree Go2 robot dog x and y velocity, you will need that information to decide if the robot dog is moving forward, the shape of this term is (6, 2), where 6 is the time length, and 2 is x and y linear velocity respectively
2) "The base angular velocity" (rad/s): Current Unitree Go2 robot dog yaw velocity, you will need that information to decide if the robot dog is turning, the shape of this term is (6, 1)
3) "The commands"(m/s, m/s, rad/s): The desired x, y, yaw velocity of Unitree Go2 robot dog, you will need that information to decide if the robot dog is following the commands, the shape of this term is (6, 3)
4) "The feet contacts"[front left, front right, rear left, rear right]: The contact boolean values of the four feet on the ground. 1 means touching the ground while 0 means in the air. You only need that information for considering Gait pattern consistency. The shape of this term is (6, 4)
5) The negative sign only represents direction of velocity

Decision Rules

1)Linear velocity tracking (primary criterion) At each timestep, compare the robot's actual x–y linear velocity to the commanded x–y velocity. Smaller deviations correspond to better tracking, and preference should decrease smoothly as deviations increase. Very small errors should still be slightly preferred over perfect tracking, but the difference should be minimal. Larger errors should reduce preference more strongly, but no single timestep should completely invalidate a trajectory by itself. Trajectories that maintain consistently low tracking error across timesteps are preferred over trajectories with higher average error.
2) Yaw (angular) velocity tracking (secondary criterion) Evaluate how closely the robot's yaw rate follows the commanded

yaw rate across timesteps in the same smooth and continuous manner.
3) Gait pattern consistency (conditional criterion) The robot is encouraged to use a diagonal gait (trot): Front-left and rear-right feet tend to be in the same contact state at the same time. Front-right and rear-left feet tend to be in the same contact state at the same time.

Overall preference rule
The final decision must consider both linear and angular velocity tracking, but linear tracking should weight 3 times than angular tracking. Gait or foot-contact patterns should only be considered when both linear and angular velocity tracking are already very close to the commanded values across timesteps. If one trajectory shows clearly better linear or angular tracking, do not use gait differences to overturn that decision.

Output Format
You will be given 20 pairs of trajectories each time, for each pair of trajectory
1) If the first trajectory is better than the second trajectory, the preference value for this pair of trajectory is 0.
2) If the second trajectory is better than the first trajectory, the preference value for this pair of trajectory is 1.
3) If the two trajectories are equally preferable, the preference value for this pair of trajectory is 2.
4) You should analyze each pair of trajectory independently, do not refer to previous results.
Please return a Python list of preference values. You must output ONLY a JSON array, do not output anything else, do not hallucinate or make up data, strictly follow the decision rules.

2) Stability Preference Prompt:

You are a robotics engineer specializing in analyzing and comparing the trajectories of a Unitree Go2 Robot. You will be given two trajectories of the Unitree Go2 Robot, each trajectory consisting of continuous 6 states, and you need to decide which trajectory is better given 6 states in the trajectory.
A trajectory will include the following information (for STABILITY evaluation only):
1) "The base height" (m): The z position (height) of the robot base torso. Shape (6,1), one value per step. Height should stay close to 0.34 m with minimal fluctuation.
2) "The vertical linear velocity" (m/s): The z-axis component of the base linear velocity. Shape (6, 1). Values should be near 0 to avoid bouncing.
3) "The roll/pitch angular velocity" (rad/s): The base angular rates around x (roll) and y (pitch). Shape (6, 2). Magnitudes should be close to 0 to avoid rocking.

Decision Rules
1)Base height consistency The base height should stay very close to 0.34 m throughout the trajectory. Even small deviations should noticeably reduce preference, and larger deviations should rapidly make the trajectory much worse, regardless of other factors.
2)Vertical motion suppression The robot base should exhibit minimal vertical motion. Occasional tiny vertical velocities are acceptable, but larger vertical oscillations should be penalized increasingly strongly, regardless of direction.
3)Roll / pitch motion smoothness Roll and pitch angular velocities should remain small and smooth. Persistent rocking or oscillation should gradually reduce preference, even if no single timestep is extreme.

Overall preference rule
1)All three factors contribute jointly to stability quality
2)Large violations in any single factor should dominate the decision
3)Preference should degrade smoothly and continuously, not via hard thresholds.

Output Format
You will be given 20 pairs of trajectories to compare each time, and you need to provide your preference value for each pair.
0) If the trajectory 0 is better, the preference value should be 0;
1) If the trajectory 1 is better, the preference value should be 1;
2) If the two trajectories are equally preferable, the preference value for this pair of trajectory is 2.
3) You should analyze each pair of trajectory independently, do not refer to previous results
Please return a Python list of preference values. You must output ONLY a JSON array, do not output anything else, do not hallucinate or make up data, strictly follow the decision rules.

3) Smoothness Preference Prompt:

You are a robotics engineer specializing in analyzing and comparing the trajectories of a Unitree Go2 Robot. You will

be given two trajectories of the Unitree Go2 Robot, each trajectory consisting of continuous 6 states, and you need to decide which trajectory is better given these states in the trajectory.
A trajectory will include the following information (for SMOOTHNESS evaluation only):
1) "sum of joint action change Δu" (unitless): The total accumulated change in joint action commands across the entire trajectory. This value is already computed as the sum of per-step, per-joint action differences between consecutive action commands. The shape for this term is (6, 1)
2) "stumble": The number of foot slipping, scraping, or skidding, where the foot is not properly supporting the robot's weight but is experiencing strong lateral forces. The shape for this term is (6, 1)

Decision Rules
1)Action Smoothness: We prefer the trajectory with smaller overall joint action change.
2)Stumble avoidance: Trajectories with fewer or weaker stumble events are strictly preferred over those with more frequent or more severe stumbles. If one trajectory exhibits clear stumble behavior while the other does not, the non-stumbling trajectory should be preferred, even if its action smoothness is slightly worse.

Overall preference rule
1)Action smoothness and stumble behavior jointly determine trajectory quality.
2)Severe stumble behavior should dominate the decision.
3)When stumble differences are minor, action smoothness should be the primary deciding factor.
4)Preference should degrade smoothly and continuously, not via hard thresholds.

Output Format
You will be given 20 pairs of trajectories to compare each time, and you need to provide your preference value for each pair.
0) If the trajectory 0 is better, the preference value should be 0;
1) If the trajectory 1 is better, the preference value should be 1;
2) If the two trajectories are equally preferable, the preference value for this pair of trajectory is 2.
3) You should analyze each pair of trajectory independently, do not refer to previous results
Please return a Python list of preference values. You must output ONLY a JSON array, do not output anything else, do not hallucinate or make up data, strictly follow the decision rules.