

3 比赛任务概述

本赛道由腾讯开悟平台与杭州宇树科技有限公司共同定制，旨在考核单智能体强化学习算法开发、奖励函数设计以及具身智能运动控制能力。

1. 复赛：四足仿真运动控制挑战

- 任务目标：在仿真环境中，控制 Unitree Go2 四足机器人通过复杂地形。
- 核心挑战：地形包含波浪、台阶、障碍物等，要求算法具备极高的稳定性。
- 评价指标：由成功次数、通关速度、姿态稳定性等指标加权得出。

2. 决赛：四足强化学习竞速挑战

- 任务目标：将模型部署到 Unitree Go2 上，在现实搭建的复杂赛道中完成竞速。
- 核心挑战：解决 Sim-to-Real 中的 Reality Gap。现实中的摩擦力、电机响应特性、传感器噪声与仿真环境存在差异，同时赛道地形复杂多样，模型需具备泛化能力和抗干扰能力。
- 评价指标：在完成规定赛道动作的前提下，用时越短排名越高。

4 奖励函数设计

在强化学习框架中，奖励函数的设计直接决定了策略的行为模式与收敛效果。本队采用 Legged Gym 经典的奖励架构作为基础，并根据复赛仿真环境与决赛真实场景的差异，进行了针对性的迭代与剪枝。本队的设计原则从初期的高性能探索逐渐向后期的高鲁棒性迁移转变。

本队保留了通用的基础奖励项，适用于各种规模和动态参数的四足机器人，包括：

- 指令追踪：鼓励基座线速度 (v_{xy}) 和角速度 (ω_z) 贴近指令。
- 基座稳定性：惩罚基座的垂直速度 (v_z) 和横滚/俯仰角速度 (ω_{xy})，以减少机身震荡。
- 动作正则化：包括对关节扭矩 (τ)、关节速度 (\dot{q})、动作平滑度 ($a_t - a_{t-1}$) 以及关节功率的惩罚，以通过最小化能量消耗来诱导自然且低冲击的步态。

此类设计在业内已较为成熟 [1]，故不再赘述，重点阐述本队伍针对赛题特性的改进与 Sim-to-Real 适配。在备赛过程中，本队发现仿真环境的高分策略往往难以直接迁移至真实机器人。为此对奖励函数进行了如下调整：

4.1 摒弃的奖励项

在复赛阶段，为了通过复杂崎岖地形，曾引入以下奖励，但在 Sim-to-Real 过程中被移除：

- 足端摆动轨迹 (Swing Trajectory)：复赛中曾实现 `reward.swing_trajectory`，旨在奖励足端在摆动相达到特定的峰值高度，以跨越障碍。然而在实机部署中，强制规定轨迹高度导致机器人频繁出现踏空或踩脚现象，最终被移除。
- 激进的前进速度 (Raw Forward Velocity)：复赛中为了追求竞速直接奖励前进速度，但这与决赛需要精细且多变的指令调节矛盾。故转而使用基于指令的追踪奖励以换取可控性。

4.2 新增与优化的奖励项

对称接触奖励 (Symmetric Contact)

为了强制机器人学习出稳定的 Trot 步态，本队实现了 `reward.symmetric_contact` 的简单对角腿同步性奖励。当且仅当对角线的两条腿（左前-右后或右前-左后）同时处于接触或腾空状态时给予奖励：

$$r^{\text{sym}} = \mathbb{I}_{\text{moving}} \cdot (\mathbb{I}_{LF}^c \odot \mathbb{I}_{RH}^c) + (\mathbb{I}_{RF}^c \odot \mathbb{I}_{LH}^c) \quad (1)$$

该项奖励有效地引导机器人形成对称的步态模式，提升了运动的稳定性和效率。

足端滑行惩罚 (Feet Regulation)

本队注意到基于强化学习的四足机器人，其摆动腿末端执行器倾向于沿最短轨迹移动，并始终靠近地面。这会导致运动既不美观，也不适合非结构化地形。相比之下，基于最优控制的方法通常会规划平滑曲线，使得足部垂直抬起和落地。为了使机器人实现平滑而优雅的运动，本队使用了一个足部调节奖励：

$$r^{\text{fr}} = - \sum_{i \in \text{feet}} \|\mathbf{v}_{xy,i}^{\text{foot}}\|^2 \cdot \mathbb{I}_{\text{contact},i} \quad (2)$$

其中 \mathbf{v}^{foot} 为足部速度， $\mathbb{I}_{\text{contact}}$ 为接触指示器。该奖励有效地减少了足端在接触地面时的水平速度，减少了足端滑行等现象。

髋关节内收约束 (Hip Adduction Penalty)

在 Sim-to-Real 初期，本队注意到机器人倾向于将髋关节向内收缩、将脚放置在更接近中心线的位置，以最小化质心的扭矩。然而，这种行为增加了腿部碰撞的风险、降低了静止和行走时的稳定性。为缓解这一问题，本队引入一个 `_reward_dof_error_named` 惩罚函数，针对髋关节进行内收惩罚：

$$r^{\text{hip}} = \mathbb{I}_{x\text{moving}} \cdot \|q_{\text{hip}} - q_{\text{nominal}}\|^2 \quad (3)$$

其中 $\mathbb{I}_{x\text{moving}}$ 为机器人仅在 x 方向上有指令输入时的指示器， q_{nominal} 为髋关节的默认位置。该项奖励促使机器人保持适当的腿部外展角度，提升了整体的稳定性和运动表现。

经过反复多次的迭代和权重调节，最终的奖励函数项及其权重设计如下表所示，其中红色为本次优化的关键项：

Reward Term	Equation	Weight
Lin. velocity tracking	$\exp(-4\ \mathbf{v}_{xy}^{\text{cmd}} - \mathbf{v}_{xy}\ _2^2)$	1.0
Ang. velocity tracking	$\exp(-4(\omega_z^{\text{cmd}} - \omega_z)^2)$	0.5
Lin. velocity (z)	v_z^2	-0.5
Ang. velocity (xy)	$\ \boldsymbol{\omega}_{xy}\ _2^2$	-0.05
Joint acceleration	$\ddot{\mathbf{q}}^2$	-2.5×10^{-7}
Joint power	$\ \boldsymbol{\tau}\ \dot{\mathbf{q}}^T$	-2×10^{-5}
Joint Torque	$\ \boldsymbol{\tau}\ _2^2$	-0.0001
Base height	$(h^{\text{des}} - h)^2$	-1
Action rate	$\ \mathbf{a}_t - \mathbf{a}_{t-1}\ _2^2$	-0.01
Action smoothness	$\ \mathbf{a}_t - 2\mathbf{a}_{t-1} - \mathbf{a}_{t-2}\ _2^2$	-0.01
Collision	$n_{\text{collision}}$	-1
Joint limit	$n_{\text{limitation}}$	-2
Stand still	$\ \mathbf{v}_{xy}\ _2^2$	-2
Termination	\mathbb{I}_{fail}	-10
Feet regulation	r^{fr}	-0.02
Symmetric contact	r^{sym}	0.05
Hip Adduction Error	r^{hip}	-0.5

表 1：奖励函数设计与权重

5 强化学习算法

5.1 Baseline

本队首先在赛事方提供的基线模型上进行改进。Baseline 由三个核心子网络组成：历史编码器 History Encoder、策略网络 Actor 和价值网络 Critic 以处理部分可观测性问题。

5.1.1 网络结构设计

- 历史编码器 (History Encoder)：由于单帧观测无法完全推断环境属性（如摩擦系数、负载），我们将过去 H 步的关节状态与感知信息作为输入，通过多层感知机 MLP 将其压缩为一个低维的潜向量 Latent Vector。该向量旨在显式编码机器人的动力学特征。
- 策略网络 (Actor)：接收由编码器输出的潜在向量以及当前的指令，输出关节位置的目标控制量。
- 价值网络 (Critic)：拥有特权信息，接收包含环境物理参数如地形高度图、摩擦力真值等的完整状态信息。

具体网络参数配置如下表所示。所有隐藏层均采用 **ELU** 激活函数，Critic 网络的隐藏层后额外添加 LayerNorm 以稳定训练。

表 2: Network Architecture of the Baseline Model

Module	Input Representation	Hidden Layers	Output
History Encoder	Trajectory History ($H \times S_{obs}$)	[256, 128]	Latent Vector z_t
Actor (Policy)	Latent $z_t \oplus$ Command c_t	[256, 128, 64]	Action a_t
Critic (Value)	Privileged Observations O_{priv}	[512, 256, 128]	Value Estimate $V(s_t)$

5.1.2 损失函数与优化目标

采用 PPO 算法进行策略更新。为强化编码器对物理世界的理解，在标准 PPO 损失的基础上，增加一项线速度预测辅助损失 (Linear Velocity Prediction Loss)。总损失函数 L 定义如下：

$$L = L_{surr} + c_1 L_{value} + c_2 L_{vel} - c_3 L_{entropy} \quad (4)$$

其中：

- L_{surr} 为 PPO 的截断替代目标损失 Clipped Surrogate Loss。
- L_{value} 为价值函数的均方误差损失，同样采用了截断机制以限制梯度更新幅度。
- $L_{vel} = \|\hat{v}_{xy} - v_{xy}^{gt}\|^2$ 是辅助损失，要求编码器的输出能预测当前机器人的真实线速度，强迫潜在向量 z_t 包含运动学特征。
- $L_{entropy}$ 为策略熵，用于鼓励探索。

5.1.3 算法流程

Algorithm 1 Baseline Training

Require: Policy π_θ , Value V_ϕ , Encoder E_ψ , Buffer \mathcal{D}

```

1: Initialize learning rate  $\alpha$ , target KL  $d_{targ}$ 
2: for iteration = 1, ...,  $N$  do
3:   Rollout Phase:
4:   for step = 1, ...,  $T$  do
5:     Encode history:  $z_t = E_\psi(h_t)$ 
6:     Sample action:  $a_t \sim \pi_\theta(\cdot | z_t, c_t)$ 
7:     Execute  $a_t$ , observe  $s_{t+1}, r_t, d_t$ 
8:     Store transition  $(s_t, h_t, a_t, r_t, O_{priv})$  in  $\mathcal{D}$ 
9:   end for
10:  Compute GAE returns  $\hat{A}_t$  using  $V_\phi$ 
11:  Update Phase:
12:  for epoch = 1, ...,  $K$  do
13:    for minibatch in  $\mathcal{D}$  do
14:      Compute latent  $z = E_\psi(h)$  and predict vel  $\hat{v} = \text{Head}(z)$ 
15:      Calculate  $L_{total} = L_{ppo}(\pi) + L_{val}(V) + \|\hat{v} - v_{gt}\|^2 + H(\pi)$ 
16:      Update parameters  $\theta, \phi, \psi$  via Adam
17:    end for
18:  end for
19: end for

```

5.2 自监督潜变量表示学习 SLR

为了进一步提升模型在未知环境下的泛化能力，摆脱对显式特权信息如地面摩擦系数、负载质量等的依赖，本队复现了 **SLR (Self-learning Latent Representation)** 算法 [2]。

该方法的核心思想是：智能体不应依赖人类手动定义的物理参数，而应通过自身的本体感知历史 Proprioceptive History，在马尔可夫决策过程 MDP 的引导下，自监督地学习出对运动控制最有用的环境潜变量表征 Latent Representation。

5.2.1 网络结构设计

SLR 框架包含四个核心网络模块：编码器 Encoder、策略网络 Actor、价值网络 Critic 和转移模型 Transition Model。

- **Encoder (ϕ)**: 输入过去 H 步的历史观测 o_t^H ，输出当前的潜变量 z_t 。
- **Actor (π)**: 输入当前观测 o_t 和潜变量 z_t ，输出动作 a_t 。注意：在更新 Actor 时，对 z_t 施加梯度截断，防止策略网络的更新直接干扰潜变量的物理含义。
- **Critic (V)**: 输入 o_t 和 z_t ，预测状态价值。Critic 的梯度会反向传播给 Encoder，引导 z_t 包含有助于价值最大化的信息。
- **Transition Model (μ)**: 模拟环境的动力学演变。输入当前 z_t 和动作 a_t ，预测下一时刻的潜变量 \tilde{z}_{t+1} 。

本队复现时采用的具体参数如下表所示，所有隐藏层均采用 **ELU** 激活函数：

表 3: Network Architecture for SLR Implementation

Module	Inputs	Hidden Layers	Outputs
Encoder	History Obs o_t^H	[256, 128]	Latent z_t
Actor	Obs o_t , Latent sg[z_t]	[512, 256, 128]	Action a_t
Critic	Obs o_t , Latent z_t	[512, 256, 128]	Value V_t
TransModel	Latent z_t , Action a_t	[256, 128]	Pred Latent \tilde{z}_{t+1}

5.2.2 潜变量优化：三元组损失 Triplet Loss

为了确保 Encoder 学习到的 z_t 具有时序一致性和物理意义，引入了三元组损失 \mathcal{L}_{trip} 。该损失函数要求 Transition Model 预测的下一时刻潜变量 \tilde{z}_{t+1} 应当与 Encoder 实际生成的下一时刻潜变量 z_{t+1} 尽可能接近，同时与随机采样的其他时刻潜变量 z_{t+n} 保持距离：

$$\mathcal{L}_{trip}(z_{t+1}, \tilde{z}_{t+1}, z_{t+n}) = \max \left(\|z_{t+1} - \tilde{z}_{t+1}\|_2^2 - \|z_{t+1} - z_{t+n}\|_2^2 + m, 0 \right) \quad (5)$$

其中 $m = 1.0$ 为设定的边界阈值。

5.2.3 算法流程

Algorithm 2 Self-learning Latent Representation (SLR) Training

Require: Adaptation module ϕ , Transition model μ , Policy π , Replay Buffer D

- 1: Randomly initialize networks ϕ, μ, π
 - 2: Initialize empty replay buffer D
 - 3: **for** iteration = 1, 2, ... **do**
 - 4: $o_0 \leftarrow \text{env.reset}()$
 - 5: **Rollout Phase**
 - 6: **for** step $t = 0, 1, \dots, T$ **do**
 - 7: Generate latent: $z_t \leftarrow \phi(o_t^H)$
 - 8: Sample action (stop grad on z): $a_t \leftarrow \pi(o_t, \text{sg}[z_t])$
 - 9: Step environment: $o_{t+1}, r_t \leftarrow \text{env.step}(a_t)$
 - 10: Store tuple (o_t, a_t, r_t, o_{t+1}) in D
 - 11: **end for**
 - 12: **Update Phase**
 - 13: Compute next latent: $z_{t+1} \leftarrow \phi(o_{t+1}^H)$
 - 14: Predict next latent: $\tilde{z}_{t+1} \leftarrow \mu(z_t, a_t)$
 - 15: Sample negative sample: $z_{t+n} \leftarrow \text{RandomSample}(D)$
 - 16: Compute Triplet Loss \mathcal{L}_{trip} (Eq. 5) and PPO Loss \mathcal{L}_{ppo}
 - 17: Update π, μ, ϕ by optimizing total loss: $\mathcal{L}_{total} = \mathcal{L}_{ppo} + \alpha \mathcal{L}_{trip}$
 - 18: Empty Buffer D
 - 19: **end for**
-

5.3 并行教师-学生强化学习框架 (CTS)

虽然 SLR 在一定程度上提升了模型的泛化能力，但由于缺乏显式的物理参数指导，训练过程较为缓慢，且在复杂地形下的表现仍不尽如人意。而在本赛道平台的约束下，又难以实现传统教师-学生

两阶段训练 Two-stage Teacher-Student 的完整流程。

为了上述问题，本队复现最终并采用了 **CTS (Concurrent Teacher-Student)** 框架 [3]。该框架令教师组（拥有特权信息）和学生组（仅有本体感知）在同一个强化学习进程中并行训练。学生不仅通过监督学习模仿教师的潜在特征，还通过间接参与 PPO 的梯度更新，从而在非对称信息下实现更鲁棒的控制效果。

5.3.1 状态与观测空间定义

在 CTS 框架中将智能体分为两组，分别接受不同的输入信息：

- 本体感知观测 (o_t): 学生和教师均可获得的信息，包括基座角速度、重力向量、关节位置 (q)、关节速度 (\dot{q})、指令 (v^{cmd}, ω^{cmd}) 以及上一帧动作 (a_{t-1})。
- 特权状态 (s_t): 仅教师可获得的信息，在 o_t 基础上增加了基座线速度 (v_t)、地形高度采样点 (h_t)、足端接触力、归一化的域随机化参数等。

5.3.2 网络架构设计

CTS 采用非对称 Actor-Critic 架构，核心在于共享 Actor 和 Critic 网络，但根据身份使用不同的编码器：

- 特权编码器 (E_{priv}): 教师专用，将 s_t 映射为 32 维潜向量 z_t 。
- 本体感知编码器 (E_{prop}): 学生专用，将过去 H 步的历史观测序列 $o_{t-H:t}$ 映射为 32 维潜向量 z_t 。
- 共享策略 (π)与价值 (V): 无论输入来源于哪个编码器，均通过同一套 MLP 输出动作和价值估计。

本队复现时采用的具体参数如下表所示，所有隐藏层均采用 **ELU** 激活函数：

表 4: Network Architecture of CTS Framework

Module	Inputs	Hidden Layers	Outputs
Privileged Encoder	Privileged State s_t	[512, 256]	Latent z_t^t
Proprioceptive Encoder	Obs History $o_{t-H:t}$	[512, 256]	Latent z_t^s
Shared Actor	Latent $z_t \oplus o_t$	[512, 256, 128]	Action a_t
Shared Critic	Latent $z_t \oplus s_t$	[512, 256, 128]	Value V_t

5.3.3 并行训练实现

分组策略

为保证教师在训练中起到主导作用，本队将并行环境中的智能体按 3:1 比例分配：

$$\text{Teacher Mask} = (\text{env_indices} \mod 4 \neq 3)$$

潜变量球归一化

在将编码器的输出输入到共享网络之前，对潜在向量进行 L_2 归一化，将其映射到一个单位超球面 (Unit Hypersphere) 上。这缓解了因输入维度差异导致的数值不稳定性，同时迫使历史编码器学习教师潜在特征的方向而非模长，带来更快的收敛速度。

双重优化目标

训练包含两个独立的优化过程：

- 1. PPO 更新 (Shared Actor-Critic & Privileged Encoder):** 教师和学生收集的轨迹共同用于更新共享的 Actor 和 Critic 网络，以最大化累积奖励：

$$\mathcal{L}_{ppo} = \mathbb{E}_{\text{teacher} \cup \text{student}} [\min(r_t A_t, \text{clip}(r_t, 1 - \epsilon, 1 + \epsilon) A_t)] \quad (6)$$

在 PPO 更新阶段，学生侧的潜向量 $z_{student}$ 进行梯度截断，不反向传播至本体感知编码器。

- 2. 重构更新 (Proprioceptive Encoder Only):** 本体感知编码器通过监督学习进行更新，目标是使学生的潜向量 $z_{student}$ 尽可能逼近教师的潜向量 $z_{teacher}$ ：

$$\mathcal{L}_{rec} = \|E_{priv}(s_t) - E_{prop}(o_{t-H:t})\|^2 \quad (7)$$

此更新仅利用学生组的数据，使用独立的优化器 `reconstruction_optimizer`，对教师侧的潜变量不进行梯度传播。

超参数设置

表 5: Hyper Parameters for CTS Training

PPO	
Batch size	4096×24
Mini-batch size	4096×6
Number of epochs	5
Clip range	0.2
Entropy coefficient	0.01
Discount factor	0.99
GAE discount factor	0.95
Desired KL-divergence	0.01
Learning rate	adaptive
Proprioceptive Encoder	
Batch size	1024×24
Mini-batch size	1024×6
Number of epochs	5
Learning rate	1×10^{-3}

5.3.4 算法流程

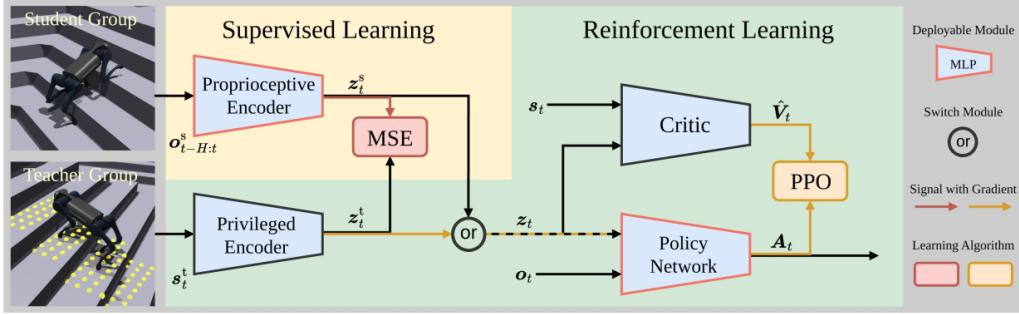


图 1: CTS Framework Overview [3]: The teacher and student agents share the same policy and value networks but utilize different encoders. The training process involves two concurrent updates: PPO update using data from both groups and reconstruction update from students group for the proprioceptive encoder.

Algorithm 3 Concurrent Teacher-Student (CTS) Training

Require: Teacher Mask M_T , Encoders E_{priv}, E_{prop} , Shared Policy π , Buffer \mathcal{D} , Buffer \mathcal{D}

1: Initialize PPO optimizer (for E_{priv}, π, V) and Recon optimizer (for E_{prop})

2: **for** iteration = 1, ..., N **do**

3: **Rollout Phase:**

4: **for** env $i \in \{0 \dots N_{envs}\}$ **do**

5: **if** $M_T[i]$ is True **then**

6: $z_i = E_{priv}(s_{t,i})$ {Teacher}

7: **else**

8: $z_i = E_{prop}(o_{t-H:t,i})$ {Student}

9: **end if**

10: $a_i \sim \pi(\cdot | o_{t,i}, z_i)$

11: Step env and store data to \mathcal{D}

12: **end for**

13: **Update Phase:**

14: **for** epoch = 1, ..., K **do**

15: **1. PPO Update:**

16: Calculate \mathcal{L}_{ppo} using data from BOTH groups.

17: Update E_{priv}, π, V . (Student latent z is detached here)

18: **2. Reconstruction Update:**

19: Calculate $\mathcal{L}_{rec} = \|E_{priv}(s_t) - E_{prop}(o_{t-H:t})\|^2$ using Student data.

20: Update E_{prop} using Recon optimizer.

21: **end for**

22: **end for**

6 Sim to Real

物理引擎 Isaac Gym 与真实物理世界之间仍存在不可忽视的 Reality Gap, 在将仿真评估表现优异的策略部署到 Unitree Go2 时遇到了一系列挑战。本节将详细阐述 Sim 与 Real 的主要差异, 以及采取的算法调优与工程改进。

6.1 部署差异与调优

在前几次部署中观察到策略表现出以下非预期的行为，并针对性进行了修正：

1. 基座高度过高：

- 现象：仿真中策略倾向于保持较高的站立姿态约 0.35m 以获得更好的越障能力，但高重心导致受到扰动时更易失稳。
- 改进：经过多次尝试，将训练中目标基座高度 h^{des} 降至 0.32m，显著提升了稳定性。

2. 交叉腿现象：

- 现象：仿真策略为了极小化能量消耗，倾向于将四肢向内收缩，导致左右腿在快速运动中发生物理碰撞，导致灾难性的摔倒。
- 改进：如第四章所述，引入了强约束的髋关节内收惩罚，迫使左右腿在运动时保持足够的间距，缓解了交叉腿现象。

3. 原地转向不稳：

- 现象：在执行原地转向指令时，策略表现出基座与足端转向不同步的问题。
- 改进：这是仿真中使用 heading 角速度指令导致的。后续在仿真斜坡地形中增加线速度为零、角速度恒定的指令形式，使转向速度和协调性得到提升。

6.2 域随机化

为了弥补仿真与现实世界之间的差距，训练时进行了广泛的随机化。在赛事方提供的基础上，增加了质心位置 y 方向和 z 方向的扰动、随机推力等。最终采用的域随机化参数及其范围如下表所示：

表 6: Domain Randomization

Randomization Term	Range	Unit
Link mass	$[0.8, 1.2] \times \text{nominal value}$	Kg
Payload mass	$[-1, 3]$	Kg
CoM of base	$[-7.5, 7.5] \times [-5, 5] \times [-5, 5]$	cm
Friction	$[0.05, 2.75]$	-
Restitution	$[0.0, 1.0]$	-
Joint K_p	$[0.8, 1.2] \times \text{nominal value}$	N-rad
Joint K_d	$[0.8, 1.2] \times \text{nominal value}$	N-rad/s
Motor Strength K_d	$[0.8, 1.2] \times \text{nominal value}$	N
Action delay	$[0, 20]$	ms
Push	$[-1, 1] \times [-1, 1]$	m/s
External Force	$[-30, 30] \times [-30, 30] \times [-30, 30]$	N

6.3 部署代码优化

为了保证控制的实时性与安全性，对部署代码做了如下优化：

1. **线程控制与精确定时：**多线程机制在处理密集计算时可能引入不可预测的延迟抖动，故显式限制 PyTorch 使用单线程推理，显著降低了延迟波动；同时，为了保证控制频率严格稳定在 50Hz，在使用简单的 `time.sleep()` 的基础上预留 1ms 余量进行忙等待，将控制周期的抖动控制在 0.01ms 以内。
2. **安全扭矩裁剪：**策略偶尔会输出极端的关节位置指令，导致 PD 控制器计算出极大的扭矩。参考仿真环境代码中步进时根据最大扭矩输出动作进行裁剪，在部署代码中保留该项处理：在下发指令前根据当前关节速度反算理论扭矩，并进行软裁剪在安全范围内以防硬件损伤。
3. **指令映射：**针对遥控器摇杆回中时可能存在的微小漂移，我们在代码中设置了 `lin.vel.deadband` 等死区参数。同时，建立了从摇杆数值 $[-1, 1]$ 到物理速度 $[-v_{max}, v_{max}]$ 的映射，使低速段控制更加细腻、高速段响应更加灵敏。

通过上述算法-仿真-工程三位一体的改进，成功在 Unitree Go2 上实现了稳定、高效的运动控制。

7 现实场地搭建训练

针对比赛要求的五种特定赛道表面（海绵斜坡、密集小斜坡、20cm 台阶、对角线障碍、碎石地），由于成本与空间限制，采取了校园环境替代与 Sim2Sim 相结合的策略。

7.1 校园环境替代

我们深入分析了五种赛道的物理特性，并寻找了如下替代方案进行测试：

1. 海绵斜坡
 - 特性分析：地面的非刚性形变使脚部陷入导致接触点与低摩擦力。
 - 替代方案：利用厚垫铺设在倾斜木板上；还在校园内的松软草坡进行了测试。
2. 密集小斜坡
 - 特性分析：连续的高频起伏和障碍地形引起机身剧烈震动，考验状态估计的鲁棒性。
 - 替代方案：选取校园绿地中的急坡和密集石块地形上作为测试场。
3. 20cm 台阶
 - 特性分析：考验越障高度与落足点精度。
 - 替代方案：利用了教学楼的标准楼梯，高度在 15-18cm。为了模拟 20cm 的状态，用堆叠的泡沫砖和木板搭建了简易台阶。重点测试了磕碰后恢复的问题，即当前脚踢到台阶边缘时，策略能否迅速调整抬腿高度。
4. 对角线障碍

- 特性分析：非对称的障碍物会破坏对角步态的稳定性，要求机器人具备良好的步态调整能力。
- 替代方案：在开阔地面上随机布置长条木方并摆放成与前进方向呈 45° 夹角。

5. 碎石地

- 特性分析：不规则的接触面与动态摩擦系数，且石块可能随受力发生滚动。
- 替代方案：在卵石路、石板路等进行测试。



图 2: 测试环境示意图

7.2 Sim2Sim

在进行真机部署前，本队还增加了 Sim2Sim 验证环节：使用 MuJoCo 物理引擎尝试搭建了与比赛赛道相同的地形模型，并利用 unitree_mujoco 项目进行测试。MuJoCo 的接触动力学模型与 Isaac Gym 存在差异，且更接近真实世界。将训练好的策略在上述环境中测试，基本能够顺利通过五种赛道，说明策略没有过拟合，具有较高的真实迁移成功率。

通过上述组合式训练场地与验证方法，在有限的条件下最大程度地模拟了决赛环境，确保了最终模型的能力。

8 训练效果分析

8.1 训练数据与仿真评估

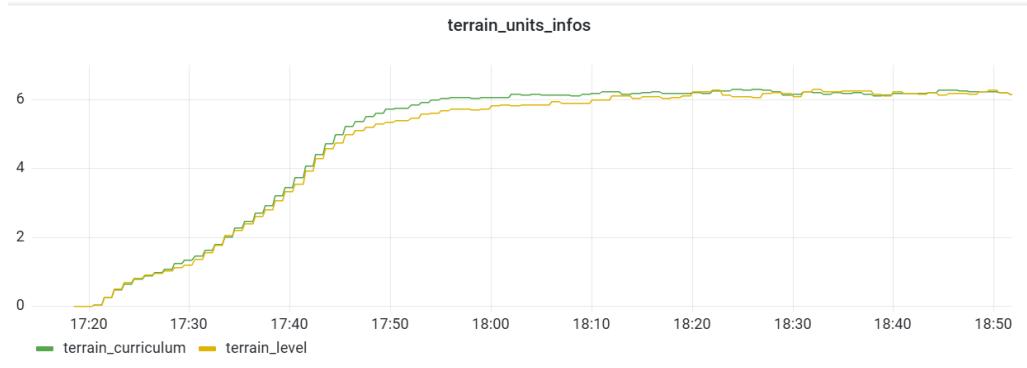


图 3: Training Curve: 绿线为教师组; 黄线为学生组

训练过程中地形等级的提升情况如图所示，它记录了训练过程中每个时刻所有智能体的平均地形等级，作为机器人整体移动能力的统计度量。这显示，在 CTS 中在大约 5000 步就基本收敛，与学生的同步训练并不会削弱教师的性能；学生表现略低于教师，但由于 Baseline 和 SLR。

8.2 基于现实反馈的模型优化

在从仿真到现实的迭代过程中，发现单纯依赖仿真高分并不意味着真机表现优异。通过部署-记录-重训的闭环，针对性地解决了以下问题：

1. 翻越障碍物能力提升

早期的 Baseline 和 SLR 策略在面对台阶时，踢到台阶边缘时需要较长时间才能感知高度变化，反应滞后。通过 CTS 架构的并行训练，学生网络 E_{prop} 被强制要求实时重构教师网络 E_{priv} 的潜在空间 z_t ，锻炼从历史关节轨迹中隐式推断地形特征的能力。优化后，机器人在接触台阶后调整步态的时间大大缩短。

2. 抗扰动能力的提升

早期模型中，观察到重心不稳容易跌倒。我们在训练中增加 y 方向随机质心位置扰动，最高 30N 随机推力和最高 1m/s 随机速度扰动。结果显示，模型在接受侧向 20N 冲击或脚底突然打滑时，能迅速通过调整落足点恢复平衡，无需人工干预。

3. 速度跟踪精度

无论在仿真评估还是实际部署，在平坦路面上模型的线速度跟踪误差低至 $0.1m/s$ 内。这得益于学生与教师策略网络共享，直接参与 RL 梯度更新，得到更适合本体感知输入的控制策略。

参考文献

- [1] Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., & Hutter, M. (2022). *Learning agile and dynamic motor skills for legged robots*. Science Robotics, 7(65), eabg3221.
- [2] Chen, S., Wan, Z., Yan, S., Zhang, C., Zhang, W., Li, Q., ... & Farrukh, F. U. D. (2024). *SLR: Learning Quadruped Locomotion without Privileged Information*. 8th Conference on Robot Learning (CoRL).
- [3] Wang, H., Luo, H., Zhang, W., & Chen, H. (2024). *CTS: Concurrent Teacher-Student Reinforcement Learning for Legged Locomotion*. IEEE Robotics and Automation Letters.
- [4] Cheng, X., Shi, K., Agarwal, A., & Pathak, D. (2024). *Extreme Parkour with Legged Robots*. 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE.
- [5] Zhuang, Z., Fu, Z., Wang, J., Atkeson, C., Schwertfeger, S., Finn, C., & Zhao, H. (2023). *Robot Parkour Learning*. 7th Annual Conference on Robot Learning (CoRL), PMLR.