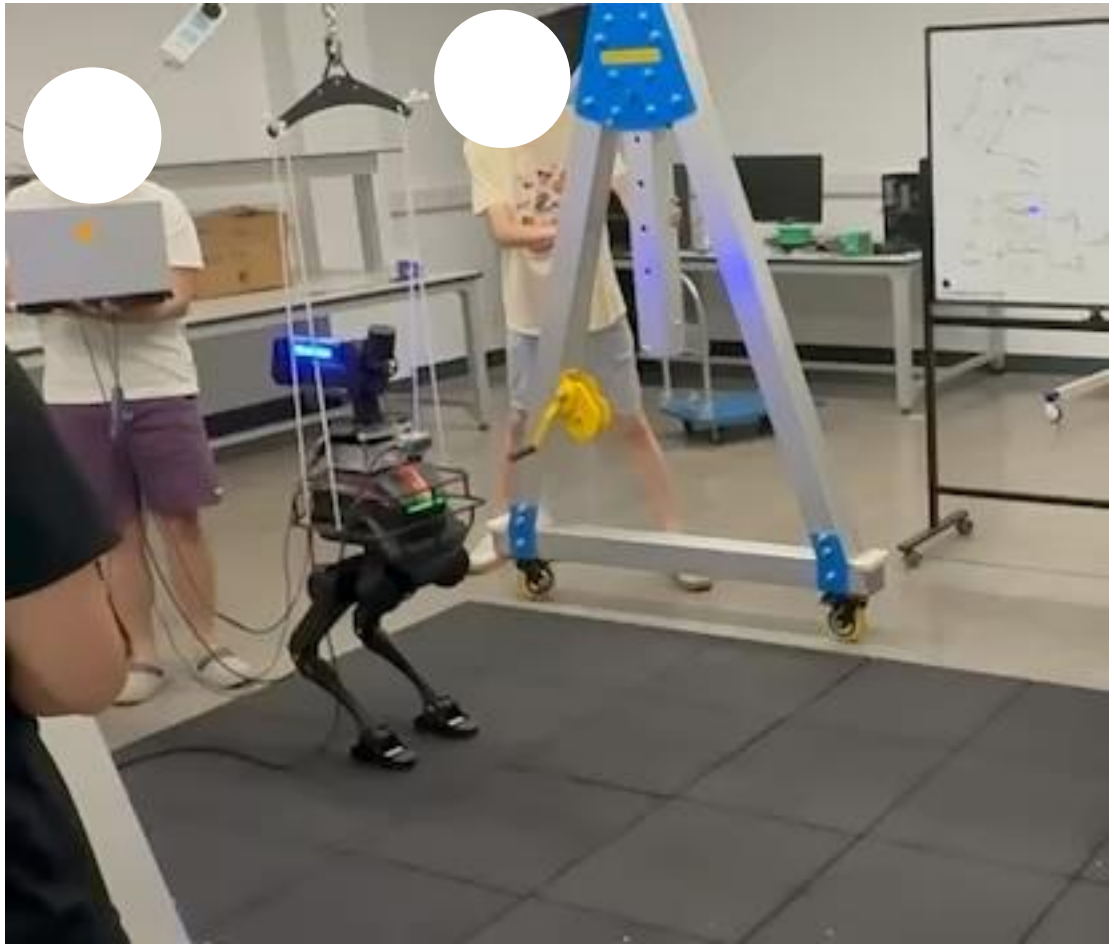
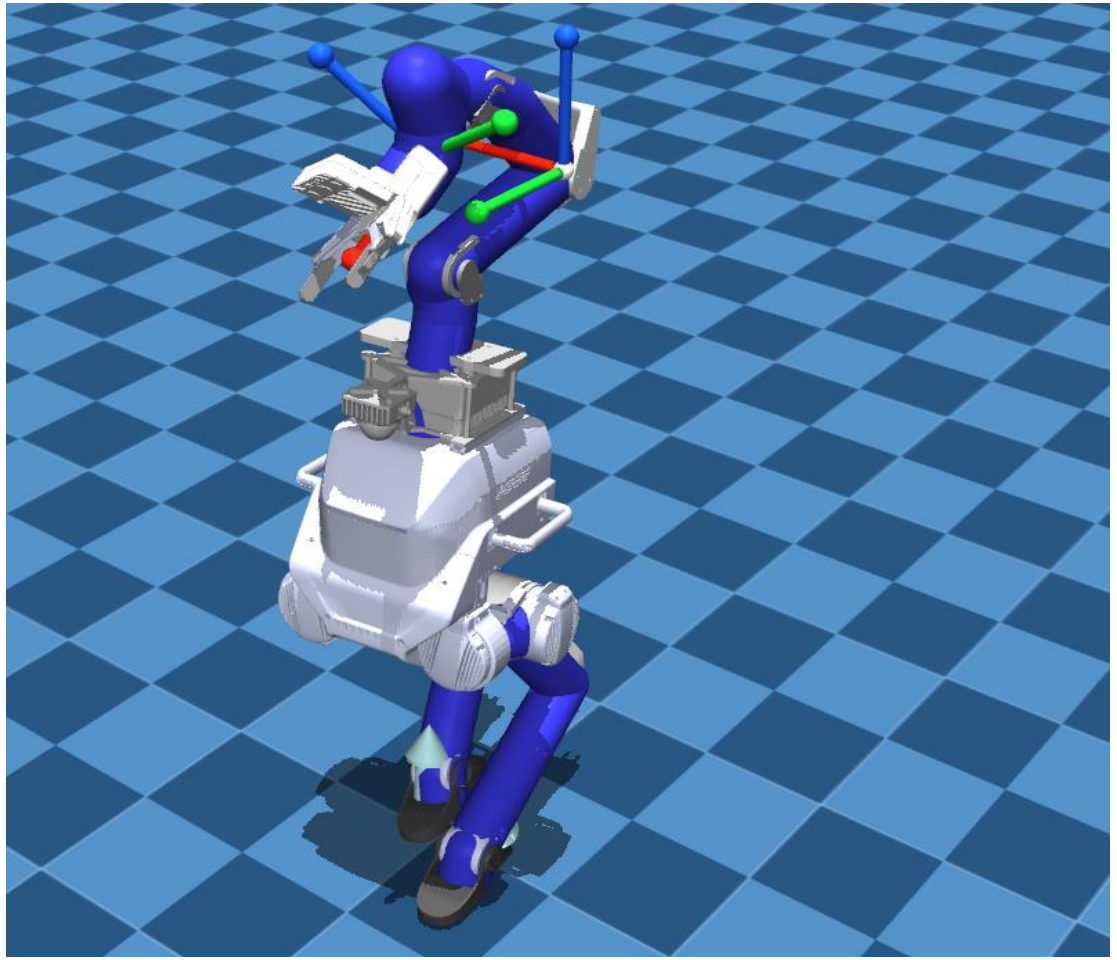
Training and Deployment of the Tron1 Whole-Body Control Strategy

6D End-Effector Pose Tracking Based on RFM/PPO

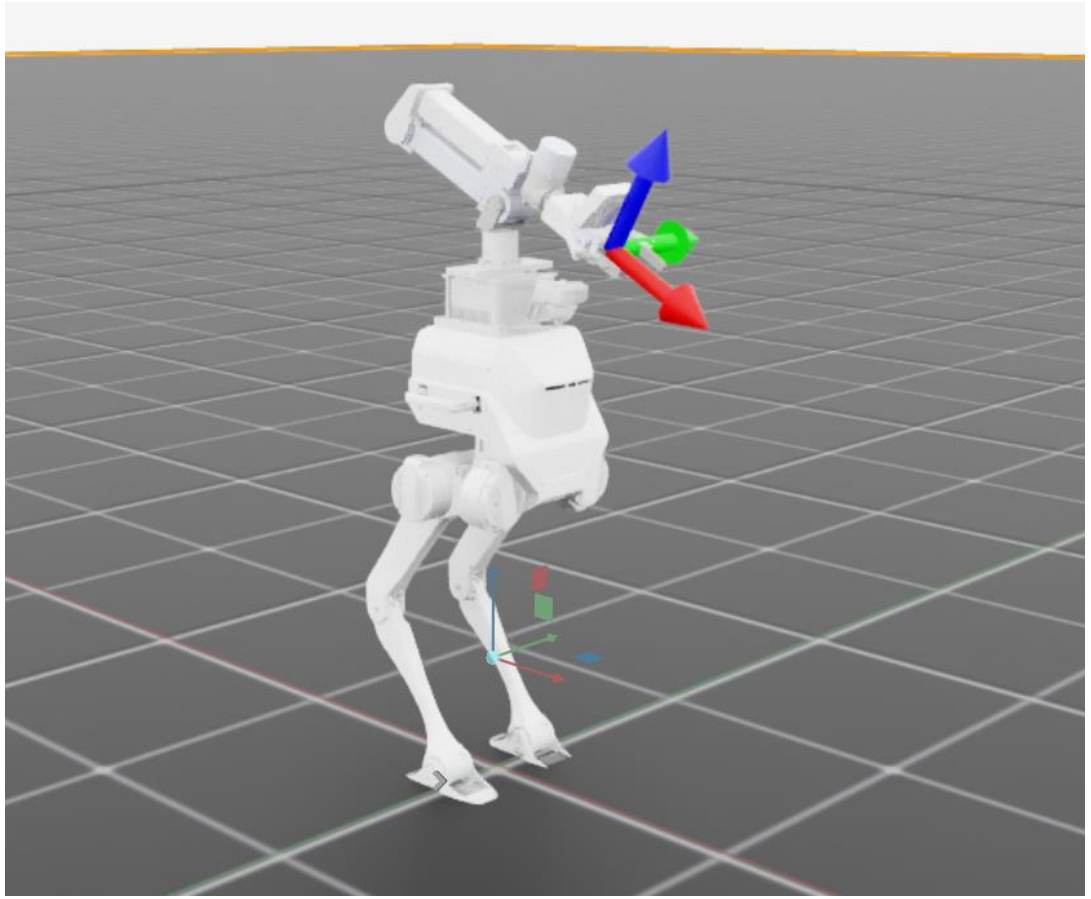
Chen Jing · 2026-07-25



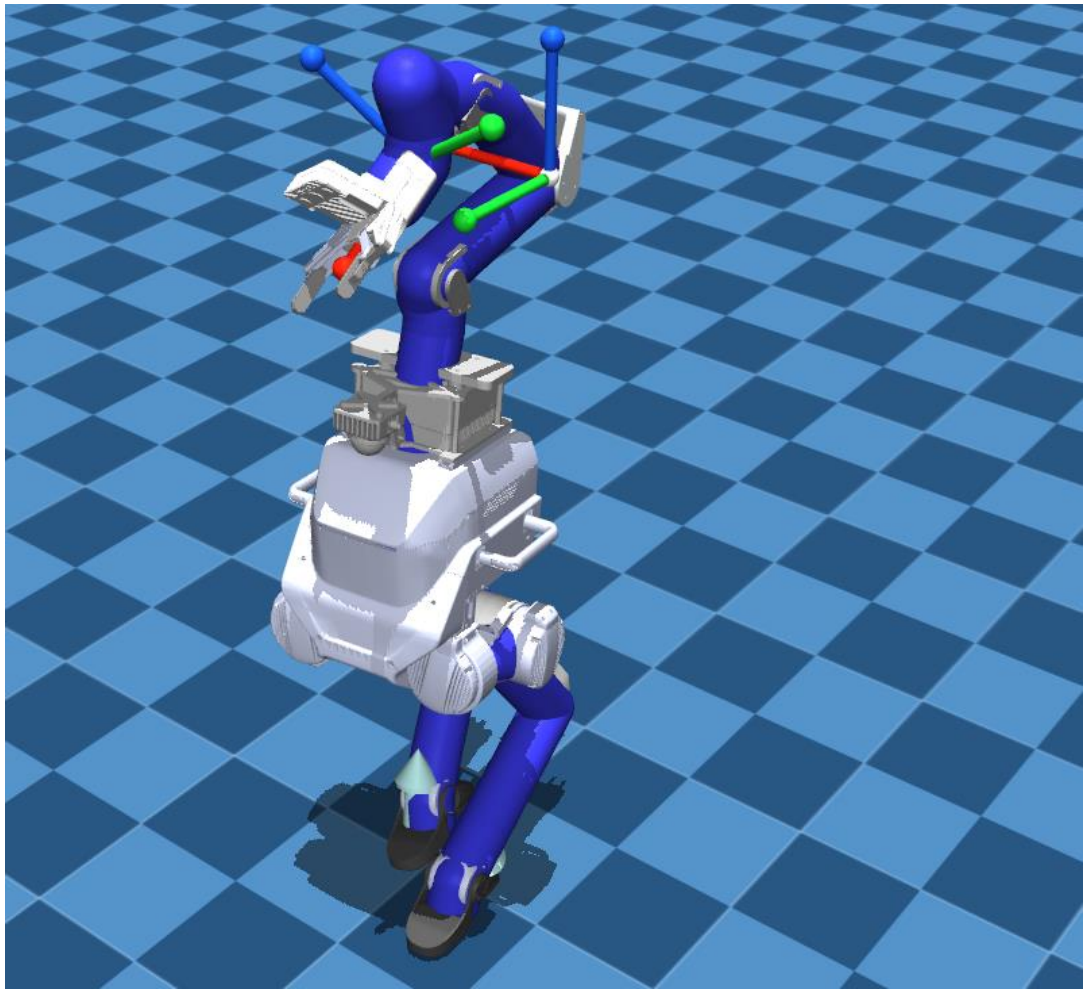


Newly refined URDF

1. EEF (End-Effector frame) on the Joint6
2. Corrected the URDF and related params for **R5** arm



Previously 0.15m forward J6 in x-direction

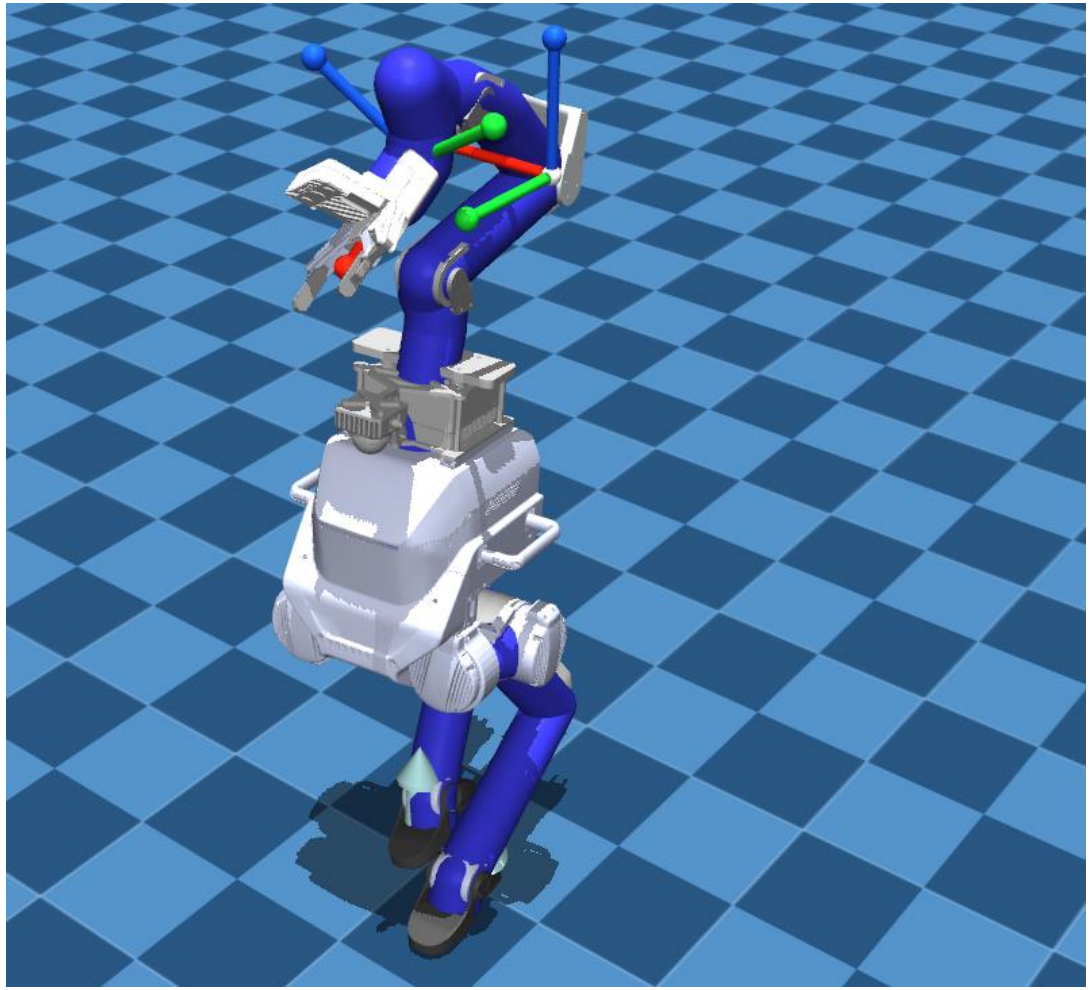


Newly Refined simply on J6

For the same chassis or end-position jitter, positioning it at J6 rather than extending it an additional 0.15 m results in stronger coupling between position and orientation, making the training signal more sensitive.

Newly refined URDF

1. EEF (End-Effector frame) on the Joint6
2. Corrected the URDF and related params for **R5** arm



Overview

Robot: LIMX SF-Tron1A with ARX R5 arm

Task: World-frame 6D end-effector pose tracking with a unified whole-body policy

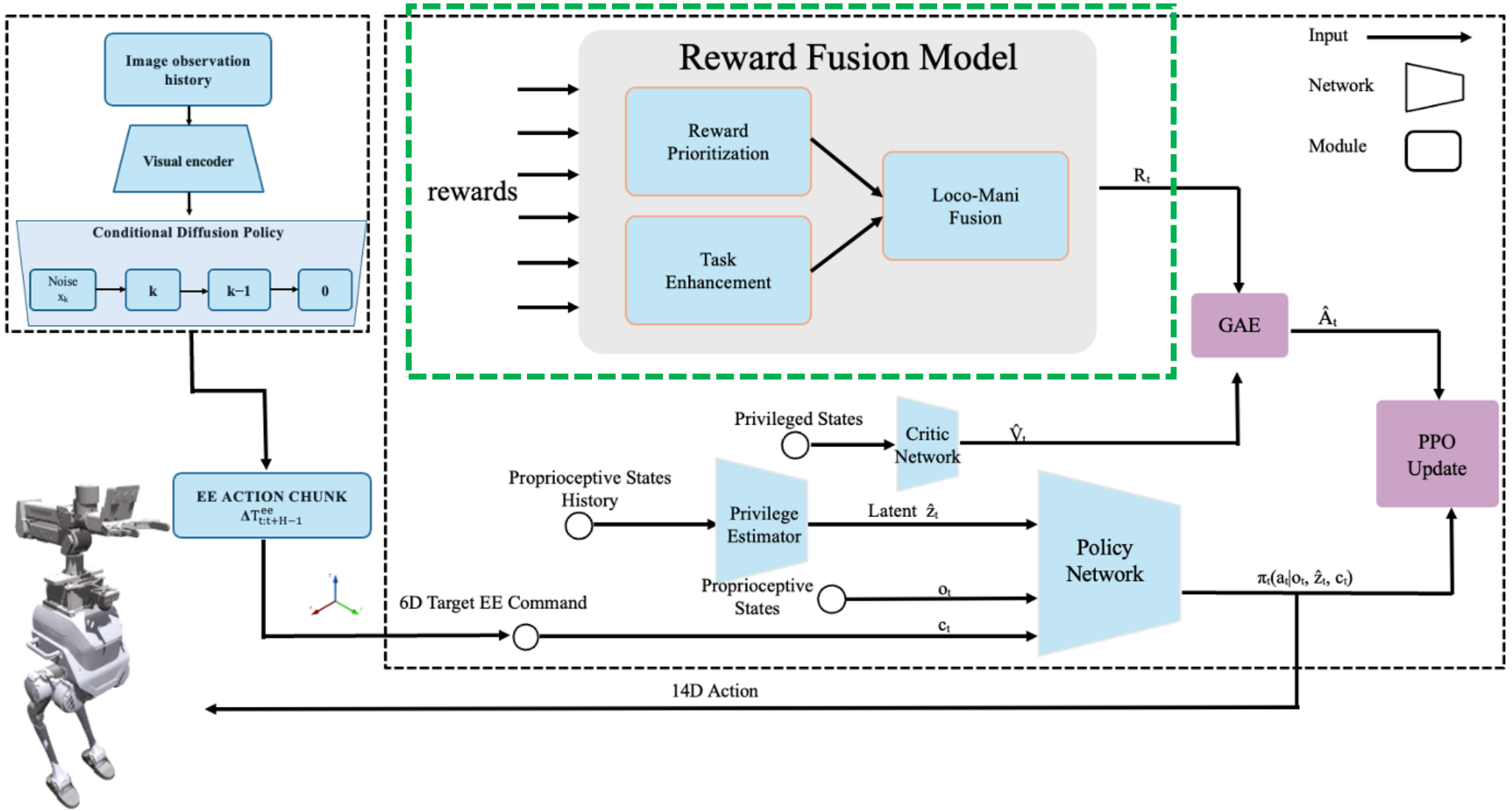
Training: Isaac Lab/Sim, RFM-style reward shaping + PPO

Evaluation: Isaac Sim (training) → MuJoCo (sim2sim) → physical robot (sim2real)

This week: Reward tuning, robustness comparison, and deployment validation

RFM (Reward Fusion Model): A nonlinear reward fusion mechanism

Since whole-body loco-manipulation is clearly phased, it is necessary to determine under what conditions a particular reward takes effect and to what extent.



Critic Network:

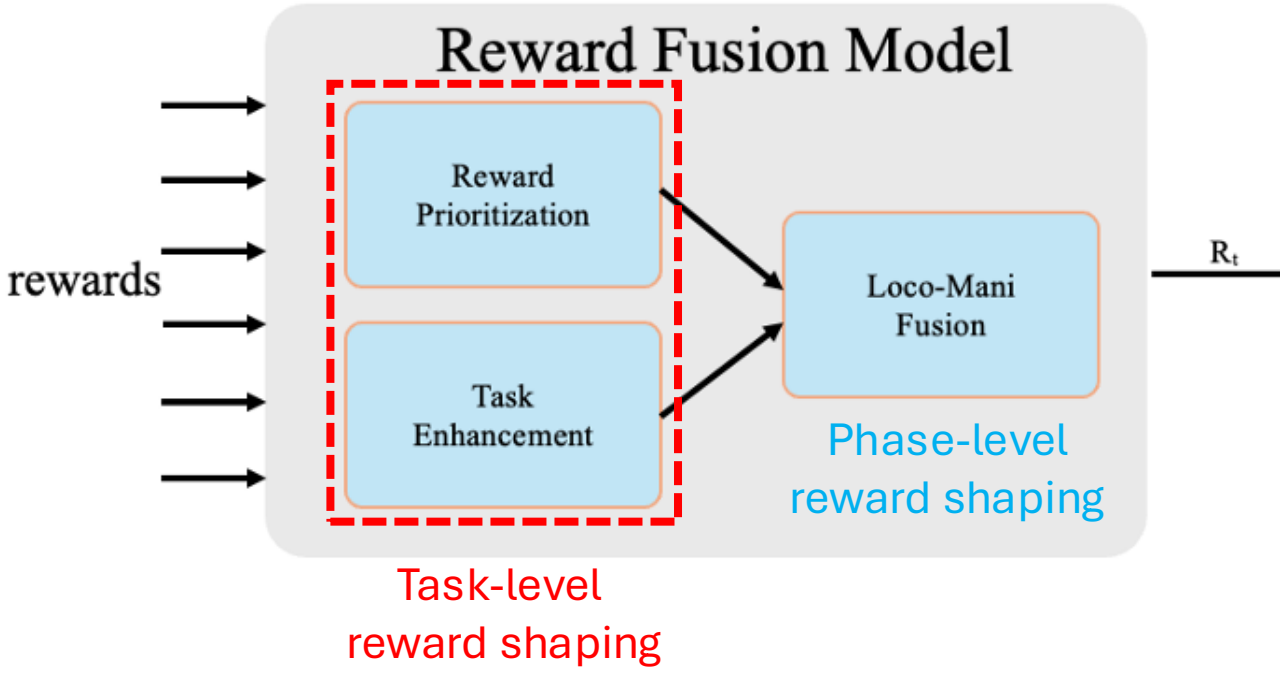
Provide a long-term value estimate $\hat{V}_t$

GAE:

Determining whether this step is better or worse than expected, creating advantage $\hat{A}_t$

RFM (Reward Fusion Model): A nonlinear reward fusion mechanism

Since whole-body loco-manipulation is clearly phased, it is necessary to determine under what conditions a particular reward takes effect and to what extent.



Loco-Mani Fusion:

Under the same unified strategy, different reward items should be enabled at different stages of the mission.

1. Expected Phase Gating L
2. Real-time State Gating G

Task Enhancement:

Converting the original error into a distinguishable learning signal, clearly showing different errors to policy.

For example, $H(e) = \underbrace{e^{-e/\sigma^2}}_{\text{broad-range term}} + \underbrace{e^{-5e/\sigma^2}}_{\text{micro-enhancement term}}$

Make sure the position/orientation reward clearly converted errors at different scales into quantifiable, computable standards.

Reward Prioritization:

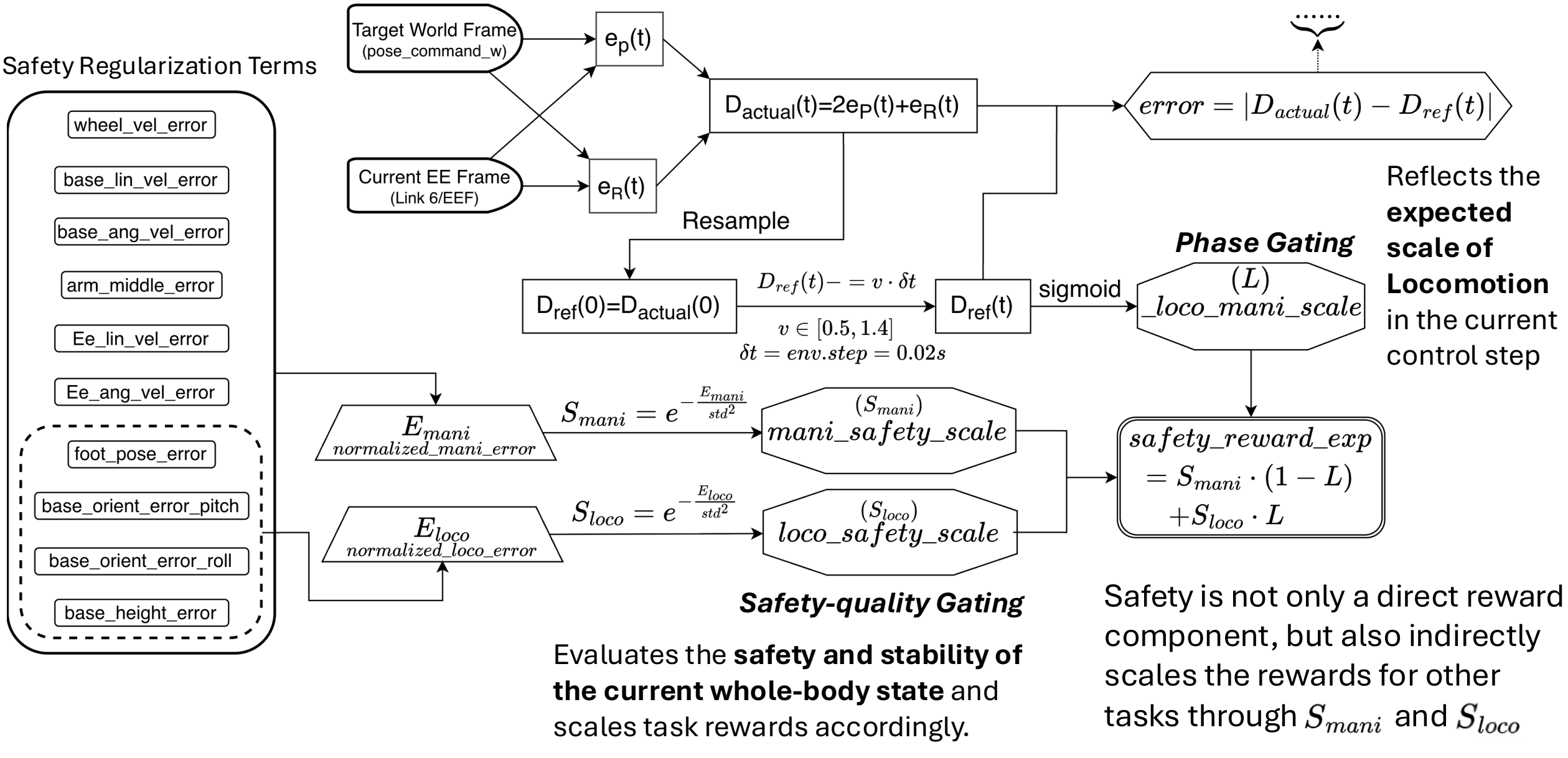
The value of the results of a reward depends on whether other underlying conditions are sufficiently good enough.

For example, $r_R = H(e_R) \underbrace{e^{-e_p/0.5}}_{} (1-L)(S_{\text{mani}} + 0.4)$

Shows that orientation accuracy is considered a valid mission result only when the position quality is high.

At the same time, **safety-quality gating** also determine how much should the Task Enhancement term should be remained……

Practical Implementation of RFM: loco_mani_scale and Safety Rewards

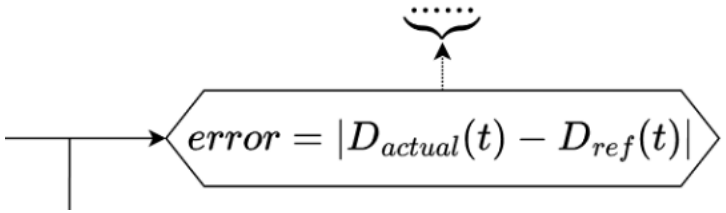


Reflects the **expected scale of Locomotion** in the current control step

Evaluates the **safety and stability of the current whole-body state** and scales task rewards accordingly.

Safety is not only a direct reward component, but also indirectly scales the rewards for other tasks through S_{mani} and S_{loco}

Locomotion-biased Schedule-following Reward `track_EE_reference_exp`



$$D_{\text{ref}} \to 0 \quad \Rightarrow \quad L = \text{sigmoid}\left(5(D_{\text{ref}} - 1)\right) \to 0.0067 \approx 0$$

$$D_{\text{actual}} \leq 0.5 \quad \Rightarrow \quad \bar{r}_{\text{ref}} = 1$$

$$r_{\text{reference}} = \exp\left(\frac{\overbrace{\max\{\text{error} - 0.5, 0\}}^{\text{A tolerance of } \pm 0.5}}{\sigma^2}\right) \cdot L \cdot \underbrace{(S_{\text{loco}} + 0.4)}_{\text{Locomotion safety gating}}$$

Role of L: As the robot enters the manipulation phase, $D_{ref} \to 0$ and the reference kernel becomes insensitive within its 0.5 dead zone.

Role of gating: Determine how much of this reward to "save" for this control step based on current locomotion security level.

Safety-aware Loco-manipulation Progress Reward `track_EE_pb`

$$G_{\text{walk}} = \text{sigmoid}\left(\frac{\mu}{\text{decay\_length}}(e_{\text{P}} - 0.5)\right), \ \mu = 0.5, \ \text{decay\_length} = 0.25$$

$$\delta e_{\text{P}} = \max\{b_{\text{P}} - e_{\text{P}}(t), 0\}, \ \delta e_{\text{R}} = \max\{b_{\text{R}} - e_{\text{R}}(t), 0\}$$

$$r_{\text{pb}} = \left(2 \cdot \delta e_{\text{P}} \cdot \underbrace{\exp\left(-\frac{b_{\text{P}}}{0.5}\right)}_{} + \delta e_{\text{R}} \cdot \underbrace{\exp\left(-\frac{b_{\text{R}}}{0.5}\right)}_{}\right) \cdot (S_{\text{loco}} + 0.4) \cdot \underbrace{(0.25 + 0.75 G_{\text{walk}})}_{G_{\text{progress}}}$$

$$\begin{aligned} e_p > 0.5\,\text{m}: &\quad G_{\text{walk}} \to 1, \quad \text{favoring long-range locomotion;} \\ e_p = 0.5\,\text{m}: &\quad G_{\text{walk}} = 0.5; \\ e_p < 0.5\,\text{m}: &\quad G_{\text{walk}} \to 0, \quad \text{favoring near-target pose refinement.} \end{aligned}$$

The smaller the historical minimum error; the greater the reward for progress.

When far from the target, progress is important, so "retain" as much of the reward as possible

When closer, the continued desire for progress can lead to lunging, shuffling, and arm swinging, so this reward component needs to be reduced

Manipulation-gated Position Precision Reward `track_EE_position_exp`

$$r_{\mathrm{P}} = \left(\underbrace{\exp\left(-\frac{e_{\mathrm{P}}}{\sigma^2}\right)}_{\text{Broad-range position tracking}} + \underbrace{\exp\left(-\frac{5 \cdot e_{\mathrm{P}}}{\sigma^2}\right)}_{\text{Near-target precision enhancement}} \right) \cdot (1 - L) \cdot (S_{\mathrm{mani}} + 0.4)$$

Provides smooth approach guidance even when errors are significant

noticeable when the error is very small, enhancing the final fine positioning.

Role of (1-L):
Ensure the reward is retained only when entering the manipulation phase. Reduce its effect during the locomotion phase.

Role of mani-safety gating:

Apart from considering error and phase terms, whether the robot's whole-body control is safe during this control step.
If so, this reward will be remained.

Position-conditioned Orientation Precision Reward `track_EE_orientation_exp`

$$r_{\mathrm{R}} = \left(\underbrace{\exp\left(-\frac{e_{\mathrm{R}}}{\sigma^2}\right)}_{\text{Broad-range orientation tracking}} + \underbrace{\exp\left(-\frac{5 \cdot e_{\mathrm{R}}}{\sigma^2}\right)}_{\text{Orientation precision enhancement}} \right) \cdot \underbrace{\exp\left(-\frac{e_{\mathrm{p}}}{0.5}\right)}_{\text{Position-priority gating}} \cdot (1 - L) \cdot (S_{\mathrm{mani}} + 0.4)$$

Why +0.4?

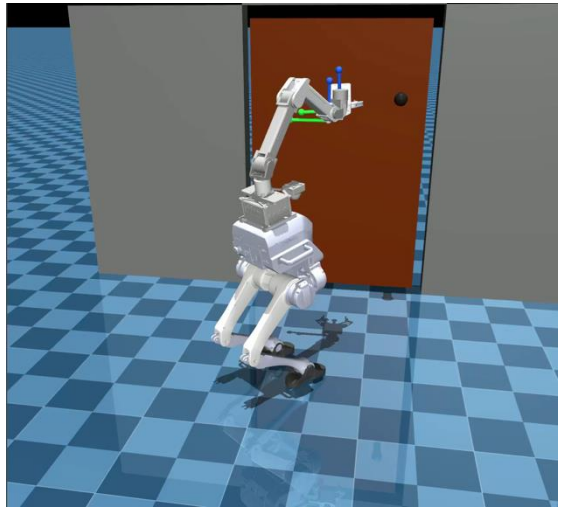
This reward is retained by the gate only when the position error is already relatively small (i.e., upon entering the manipulation phase).

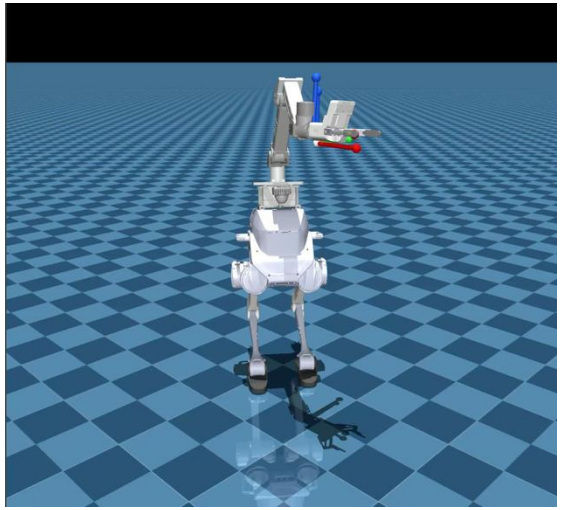
To ensure that, during the early stages of reinforcement learning when the safety scale is still very small, the reward term can still convey effective learning signals to the policy

Supplementary Non-Phased Stability Rewards

- `action_rate_l2` Penalizes abrupt step-to-step changes in joint-position commands, improving **motion smoothness**.
- `body_ee_alignment` Keeps selected arm joints **close to their neutral configuration**, avoiding excessive arm twisting.
- `feet_contacts_reg` Rewards stable double-foot contact **near the target**, reducing unnecessary stepping & switching
- `foot_flat_l2` Penalizes the **tilt** of contacting feet, **encouraging flat and stable ground support**.
- `foot_slip_l2` Penalizes **horizontal foot motion during ground contact**, suppressing foot slippage.

Strange slip in sim2sim

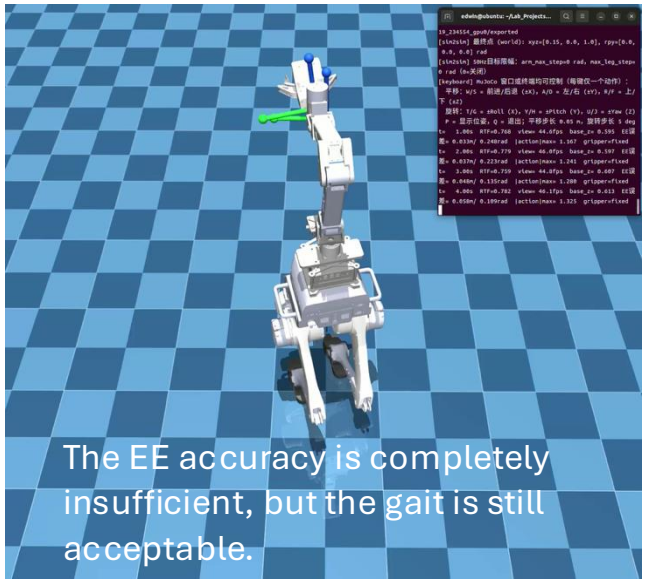




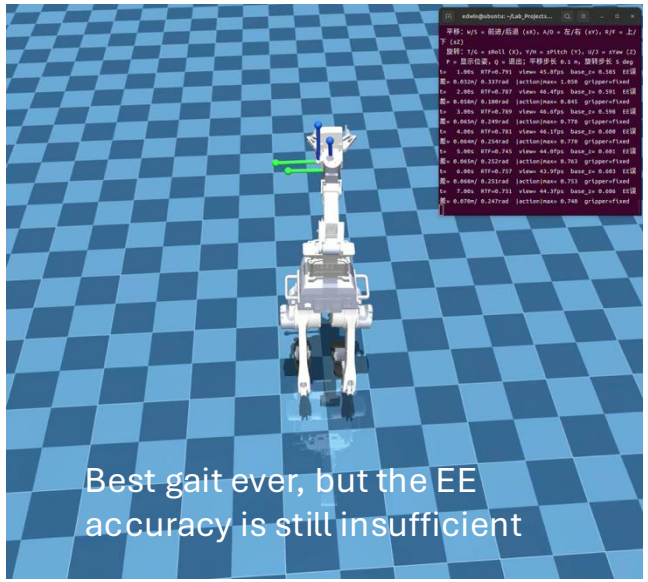
- `legs_min_separation` Penalizes **insufficient lateral separation** between the two legs, preventing **leg crossing** and overly **narrow support**.
- `base_height` Penalizes **deviations** from the desired base height, maintaining a **stable whole-body posture**.

0719: Four GPUs training for comparison

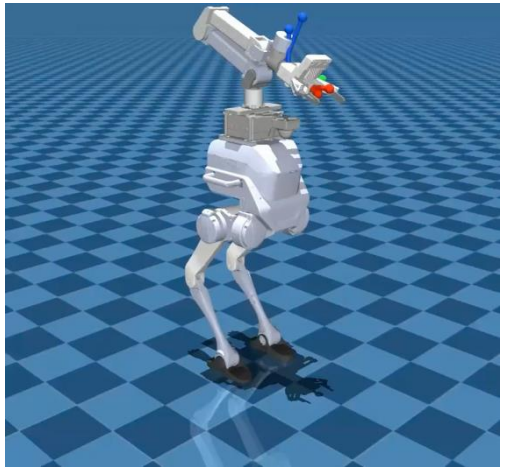
GPU0



GPU1



GPU2



GPU3

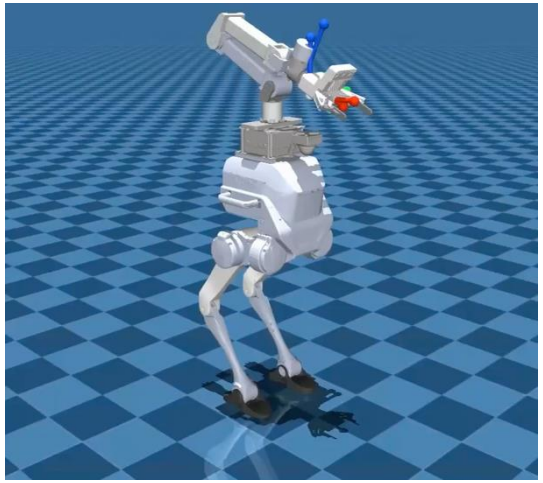


Table 1: Reward-weight configurations for four parallel training runs on July 19.

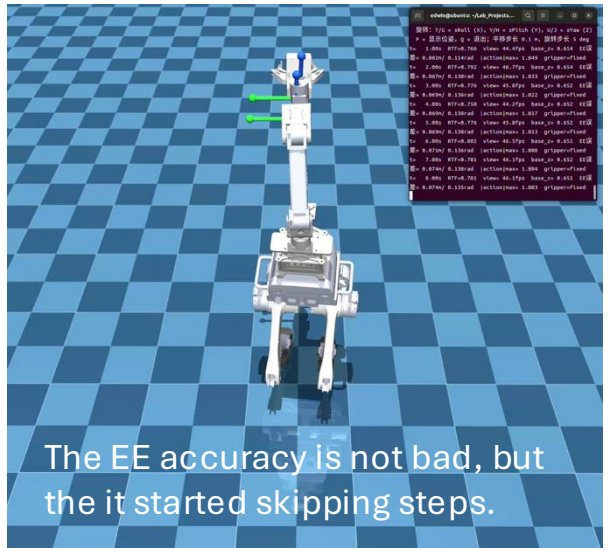
Reward term	GPU 0	GPU 1	GPU 2	GPU 3
safety_weight	3.0	3.0	3.0	3.0
track_EE_position_exp	3.0	3.0	3.0	3.0
track_EE_orientation_exp	3.0	3.0	3.0	3.0
track_EE_pb	10.0	15.0	10.0	15.0
track_EE_reference_exp	5.0	5.0	10.0	10.0
action_rate_l2	-1.0	-1.0	-1.0	-1.0
body_ee_alignment	-1.5	-1.5	-1.5	-1.5
feet_contacts_reg	0.5	0.5	0.5	0.5
foot_flat_l2	-2.0	-2.0	-2.0	-2.0
foot_slip_l2	-10.0	-10.0	-10.0	-10.0
legs_min_separation	-10.0	-10.0	-10.0	-10.0
base_height	NULL	NULL	NULL	NULL

Conclusion

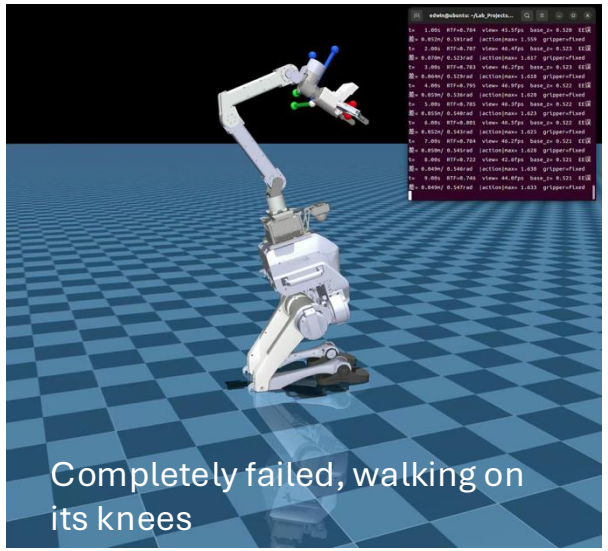
- An **excessively high safety weight** may cause the robot to learn **not to move**, resulting in a **local optimum** by sacrificing manipulation accuracy.
- Excessively **high reference weights** can cause robot to perform large movements during the initial, unstable phase of locomotion, leading to **training failure**.
- Setting the **progress weight to a moderately high** value can **improve gait to some extent**.

0720: Four GPUs training for params comparison

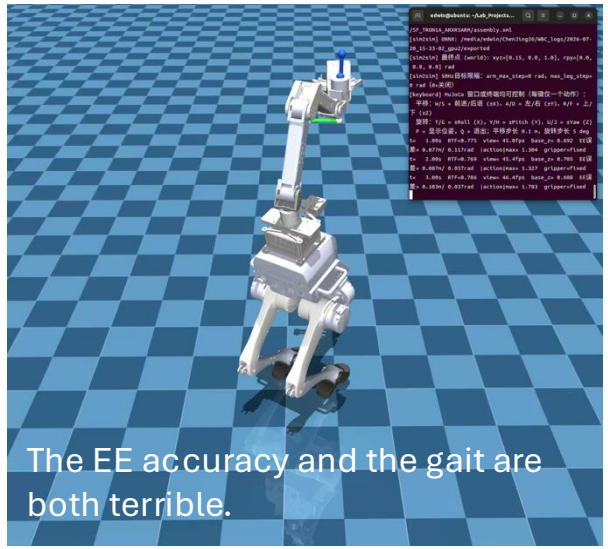
GPU0



GPU1



GPU2



GPU3

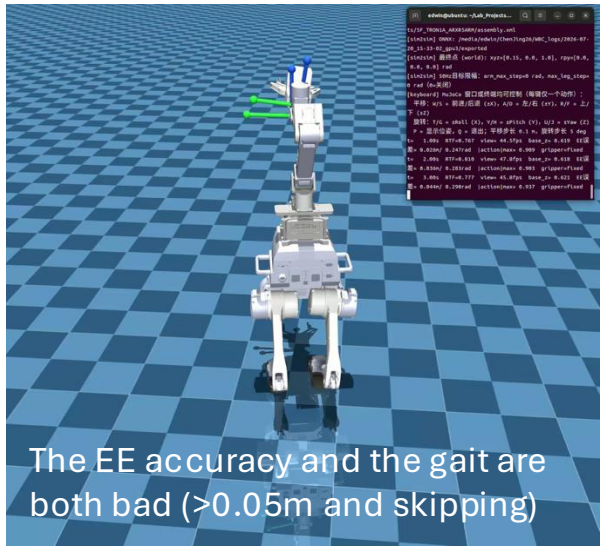


Table 2: Reward-weight configurations for four parallel training runs on July 20.

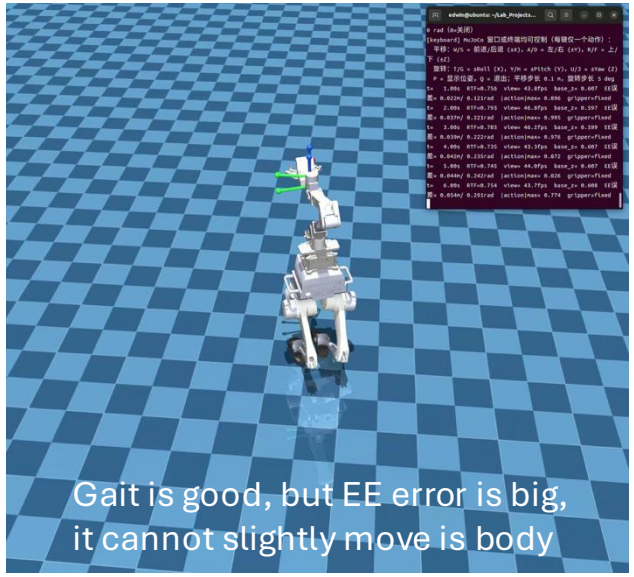
Reward term	GPU 0	GPU 1	GPU 2	GPU 3
safety_weight	1.0	1.0	1.0	1.0
track_EE_position_exp	4.0	4.0	4.0	2.0
track_EE_orientation_exp	5.0	5.0	5.0	3.0
track_EE_pb	15.0	15.0	15.0	20.0
track_EE_reference_exp	1.0	2.0	3.0	5.0
action_rate_l2	-1.0	-1.0	-1.0	-1.0
body_ee_alignment	-1.5	-1.5	-1.5	-1.5
feet_contacts_reg	1.5	1.5	1.5	1.5
foot_flat_l2	-2.0	-2.0	-2.0	-2.0
foot_slip_l2	-10.0	-10.0	-10.0	-10.0
legs_min_separation	-10.0	-10.0	-10.0	-10.0
base_height	NULL	NULL	NULL	NULL

Conclusion

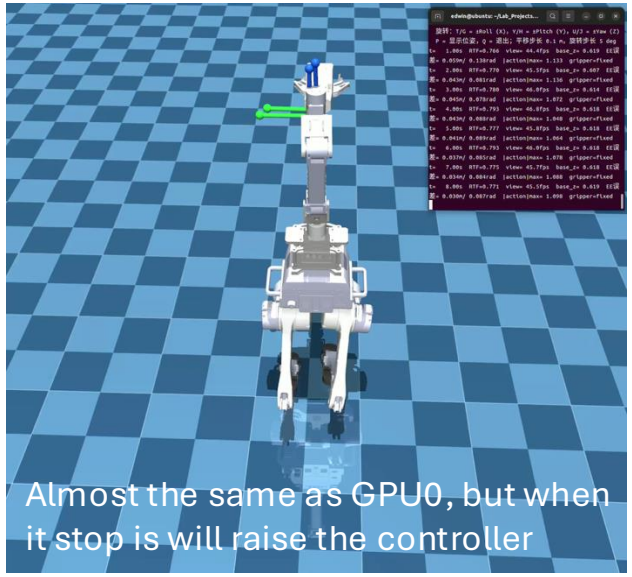
- While increasing the manipulation reward, the reference must not fall below 5.0.
- Reducing the safety bonus will make the robot's movements more agile, but it also carries the risk of overdoing it.
- Feet_contact_reg cannot effectively change the slip issue

0721: Four GPUs training for params comparison

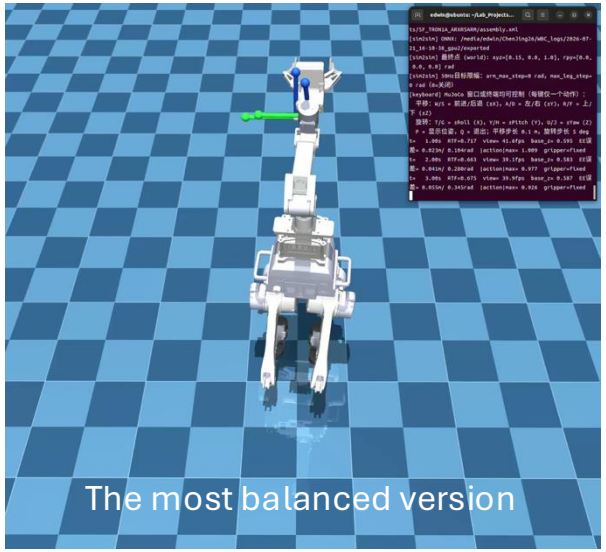
GPU0



GPU1



GPU2



GPU3

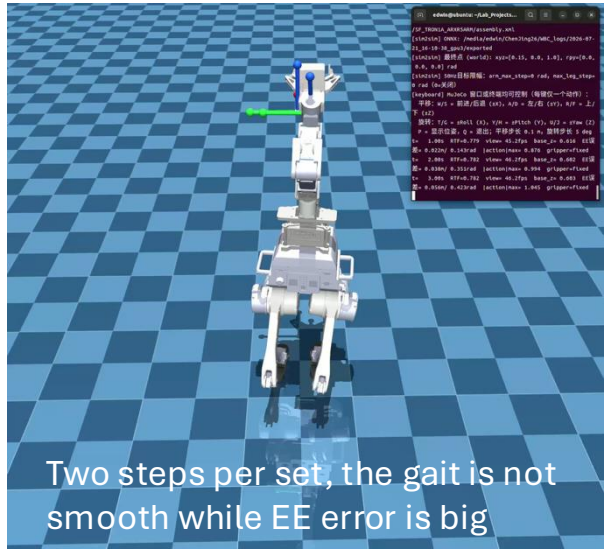


Table 3: Reward-weight configurations for four parallel training runs on July 21.

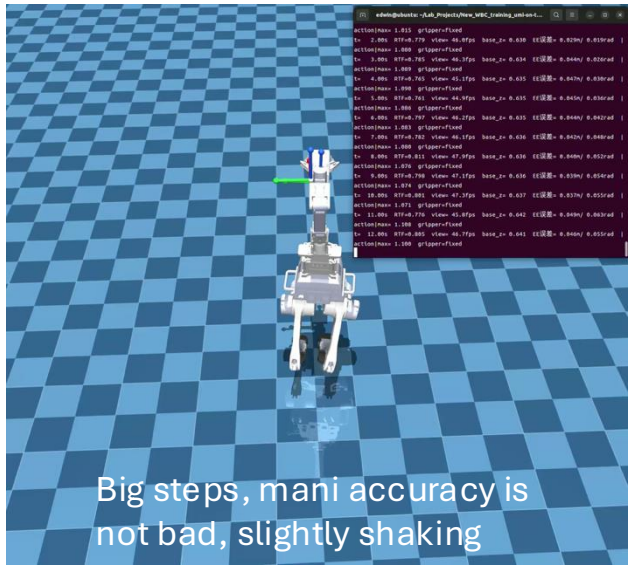
Reward term	GPU 0	GPU 1	GPU 2	GPU 3
safety_weight	3.0	3.0	2.0	3.0
track_EE_position_exp	3.0	4.0	3.0	3.0
track_EE_orientation_exp	3.0	5.0	3.0	3.0
track_EE_pb	15.0	15.0	15.0	20.0
track_EE_reference_exp	5.0	5.0	5.0	5.0
action_rate_l2	-1.0	-1.0	-1.0	-1.0
body_ee_alignment	-1.5	-1.5	-1.5	-1.5
feet_contacts_reg	0.5	0.5	0.5	0.5
foot_flat_l2	-2.0	-2.0	-2.0	-2.0
foot_slip_l2	-2.0	-2.0	-2.0	-2.0
legs_min_separation	-10.0	-10.0	-10.0	-10.0
base_height	-0.2	-0.2	-0.2	-0.2

Conclusion

- **Increasing the reward for manipulation** causes the controller to **lift off** in the final stage, resulting in very low orientation accuracy.
- The change in foot_slip **does not significantly affect the robot's foot slippage** in the sim2sim environment; it may be an issue with the sim2sim environment itself.

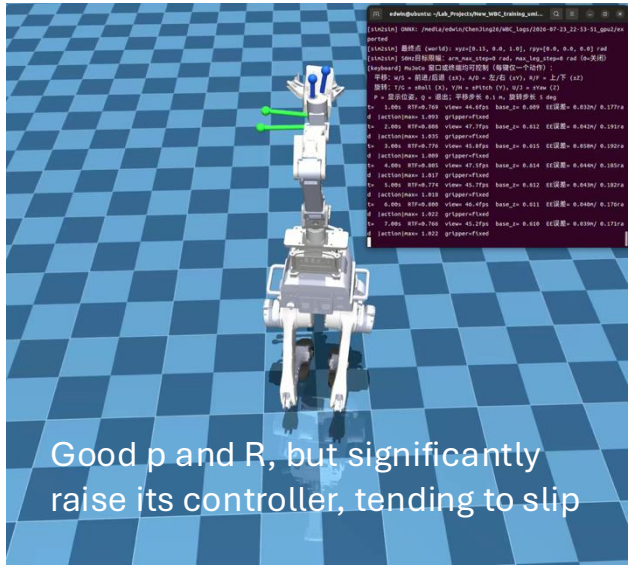
0723: Four GPUs training for params comparison

GPU0



GPU1

GPU2



GPU3

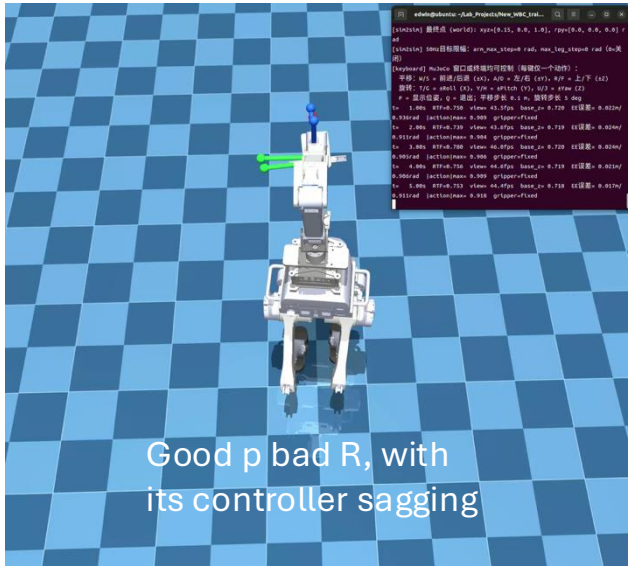


Table 4: Reward-weight configurations for four parallel training runs on July 23.

Reward term	GPU 0	GPU 1	GPU 2	GPU 3
safety_weight	3.0	3.0	3.0	3.0
track_EE_position_exp	3.0	4.0	5.0	5.0
track_EE_orientation_exp	3.0	5.0	5.0	0.0
track_EE_pb	15.0	15.0	15.0	15.0
track_EE_reference_exp	5.0	5.0	5.0	5.0
action_rate_l2	-0.5	-0.5	-0.5	-0.5
body_ee_alignment	-1.5	-1.5	-1.5	-1.5
feet_contacts_reg	0.5	0.5	0.5	0.5
foot_flat_l2	-2.0	-2.0	-2.0	-2.0
foot_slip_l2	-1.0	-1.0	-1.0	-1.0
legs_min_separation	-5.0	-5.0	-5.0	-5.0
base_height	-0.5	-0.5	-0.5	-0.5

Conclusion

- Reducing the weights of **legs_min** and **action_rate** can, to some extent, compensate for the overly conservative base strategy resulting from an excessively high **safety** weight, but make its gait not smooth and natural.
- More precisely, an excessively high **position** weight will cause the gripper to lift unnaturally.