Not Only Rewards but Also Constraints: Applications on Legged Robot Locomotion

Paper Review

24.09.01

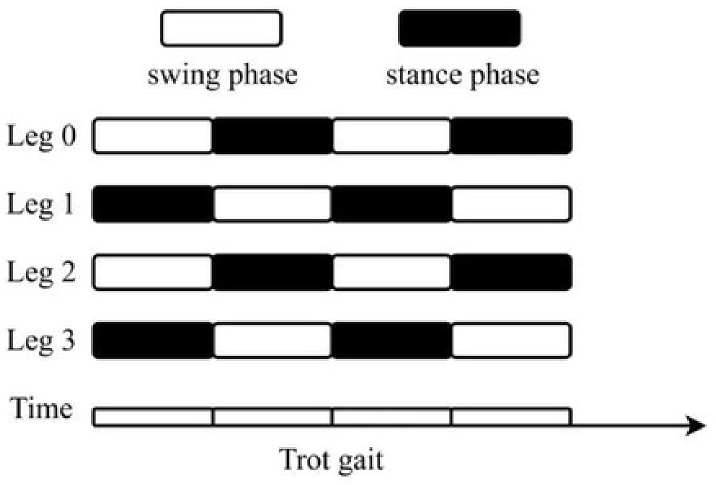
김지홍

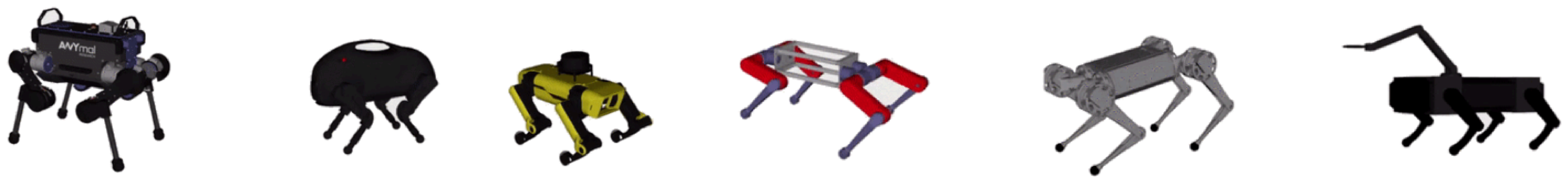
Introduction

- Step to build an RL controller
  1. Design a neural network architecture with observation and action spaces.
  2. Generate abundant environment interaction scenarios.
  3. Design Reward terms and tune their reward coefficients.

Why constraints have not been used explicitly to train policies for complex robotic systems?

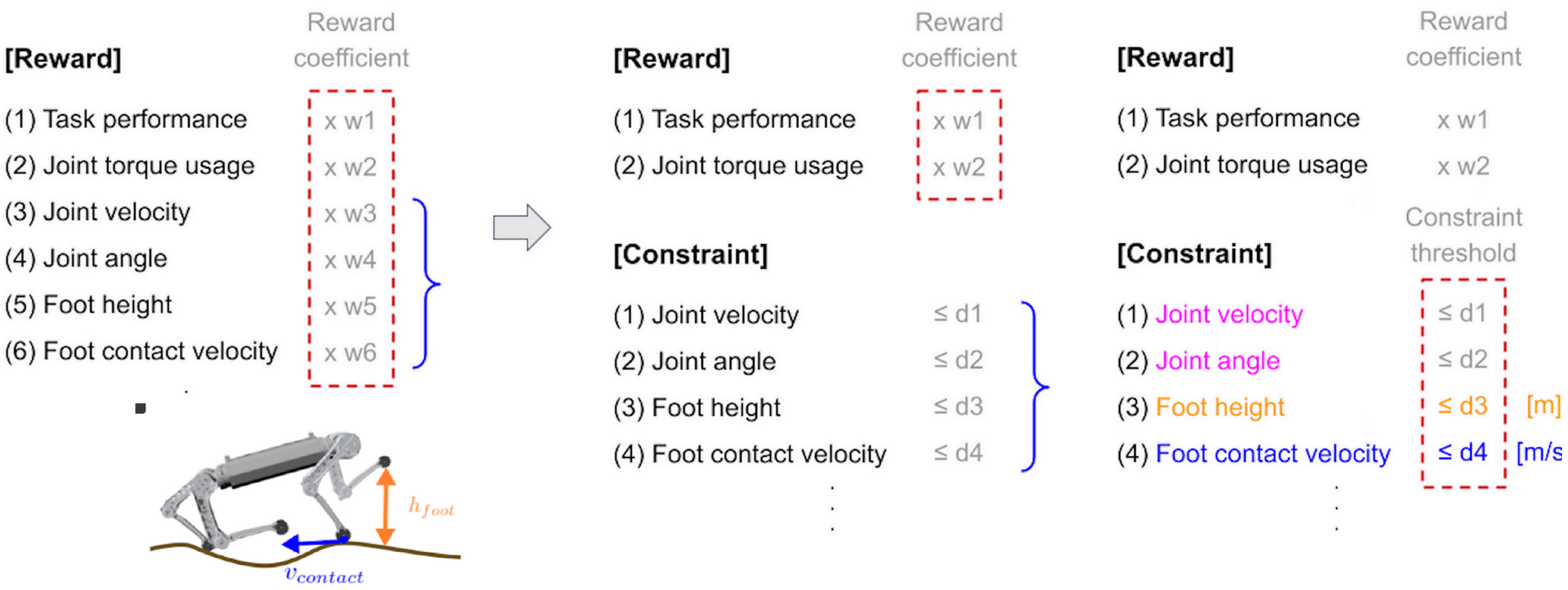
- Advantages of using Constraints.
  1. Training pipeline will be more generalizable across similar robot platforms
  2. Engineering process will be more straightforward and less time-consuming





Contribution

- Introduce an RL framework consisting of both rewards and constraints.
- Demonstrate the capability of leveraging constraints in the learning pipeline in real-world.



Constrained Markov Decision Process

$$\pi^* = \arg\max_{\pi \in \Pi_\theta} J(\pi)$$

$$\pi^* = \arg\max_{\pi \in \Pi_\theta} J(\pi)$$
$$\text{s.t. } J_{C_k}(\pi) \le d_k \quad \forall k \in \{1, \dots, K\}$$

$$J(\pi) := \mathop{\mathbb{E}}_{\rho,\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right]$$
$$J_{C_k}(\pi) := \mathop{\mathbb{E}}_{\rho,\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t C_k(s_t, a_t, s_{t+1})\right].$$

(1) Probabilistic constraint

$$\text{Prob}((s,a,s') \notin \mathbf{S}) = \mathop{\mathbb{E}}_{\rho,\pi,P}[C_k(s,a,s')] \le D_k$$

$$C_k(s,a,s') = \begin{cases} 0, & \text{if } (s,a,s') \in \mathbf{S} \\ 1, & \text{otherwise,} \end{cases}$$

(2) Average constraint

$$\mathop{\mathbb{E}}_{\rho,\pi,P}[f(s,a,s')] = \mathop{\mathbb{E}}_{\rho,\pi,P}[C_k(s,a,s')] \le D_k$$

$$C_k(s,a,s') = f(s,a,s')$$

$$\pi_{i+1} = \arg\max_{\pi \in \Pi_\theta} \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi_i} \\ a \sim \pi}} [A^{\pi_i}(s,a)]$$
$$\text{s.t. } J_{C_k}(\pi_i) + \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi_i} \\ a \sim \pi}} \left[A^{\pi_i}_{C_k}(s,a)\right] \le d_k \quad \forall k$$
$$\bar{D}_{KL}(\pi || \pi_i) \le \delta$$

[42] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained policy optimization,” in Proc. Int. Conf. Mach. Learn., 2017

[48] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in Proc. Int. Conf. Mach. Learn., 2015

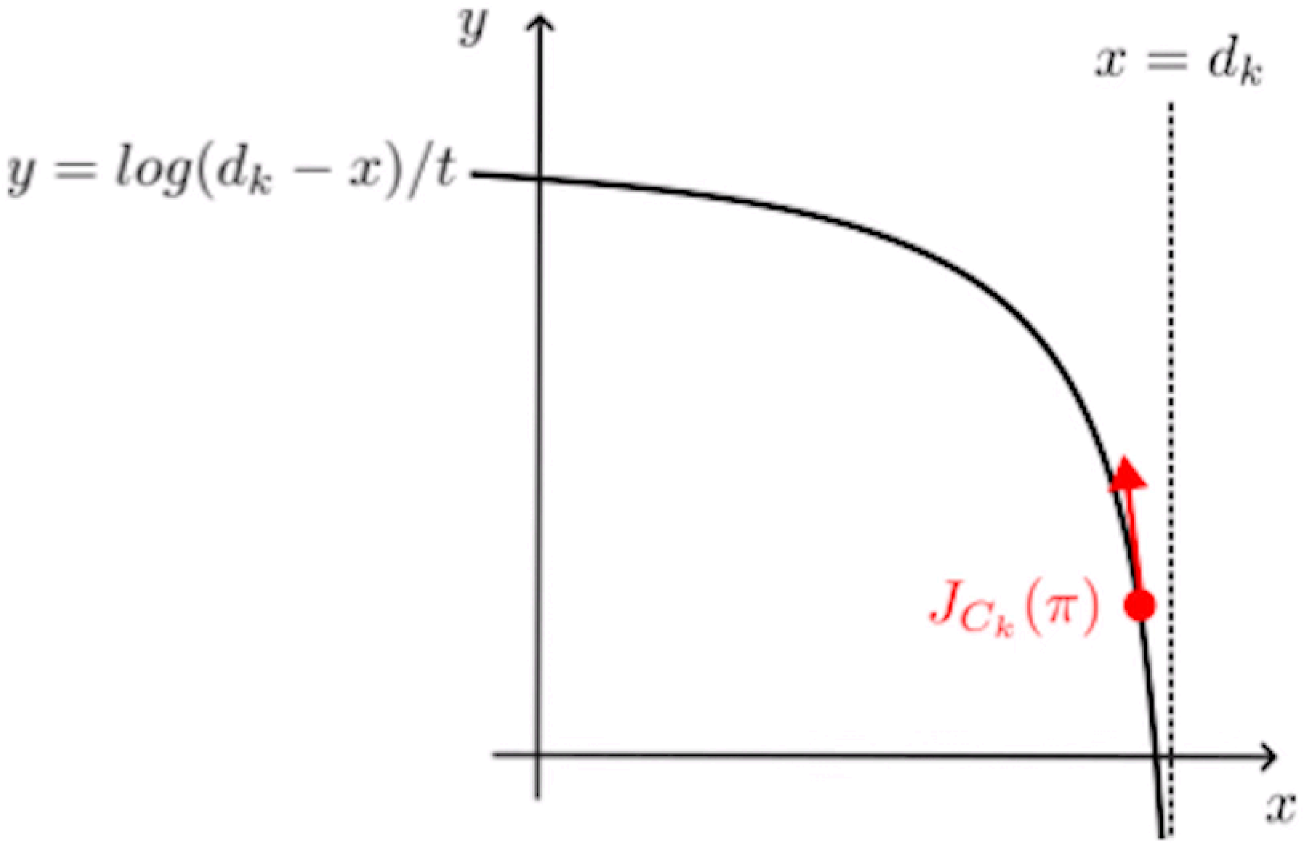
[43] Y. Liu, J. Ding, and X. Liu, “IPO: Interior-point policy optimization under constraints,” in Proc. AAAI Conf. Artif. Intell., vol. 34, no. 04, 2020

Policy Optimization

$$J_{C_k}(\pi) := \mathop{\mathbb{E}}_{\rho,\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t C_k(s_t, a_t, s_{t+1})\right] \leq d_k$$

$$\underset{\pi \in \Pi_\theta}{\text{maximize}}\ J(\pi) + \sum_{k=1}^{K} \log\left(d_k - J_{C_k}(\pi)\right)/t$$

penalize the policy as it gets closer to violating the constraint.



- If $J_{Ck}(\pi)$ is low
  - policy is well within the constraint
  - logarithmic term has a high value
  - contributes positively to the overall objective function

- If $J_{Ck}(\pi)$ approached d_k
  - effectively reducing the overall objective function
  - apply steep penalty as it gets closer to the constraint limit

- If d_k is too low
  - ensure safety, but limit the policy's ability to explore and achieve high rewards
  - lead to sub-optimal policies where the agent is overly conservative

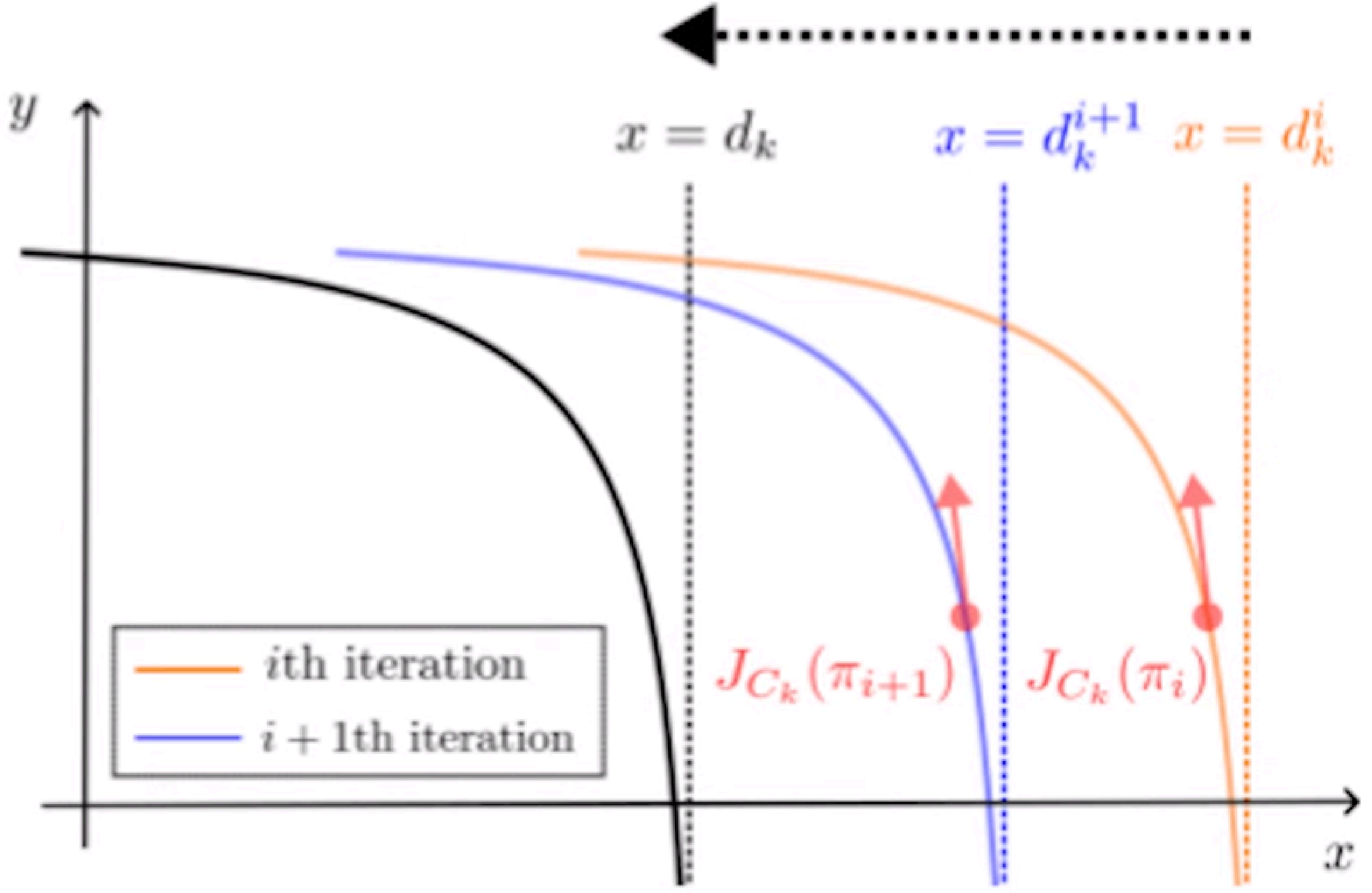
Modified IPO

- Adaptive Constraint Thresholding

$$J_{C_k}(\pi) := \mathbb{E}_{\rho,\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t C_k(s_t, a_t, s_{t+1})\right] \le d_k$$

$$\underset{\pi\in\Pi_\theta}{\text{maximize}}\ J(\pi) + \sum_{k=1}^{K}\log\left(d_k - J_{C_k}(\pi)\right)/t$$

$$d_k^i = \max(d_k, J_{C_k}(\pi_i) + \alpha \cdot d_k)$$

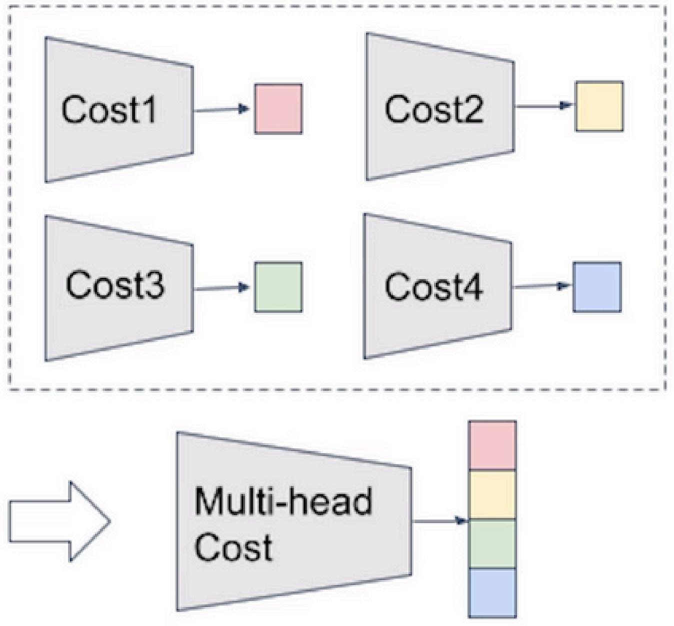


- Multihead Cost Value Function

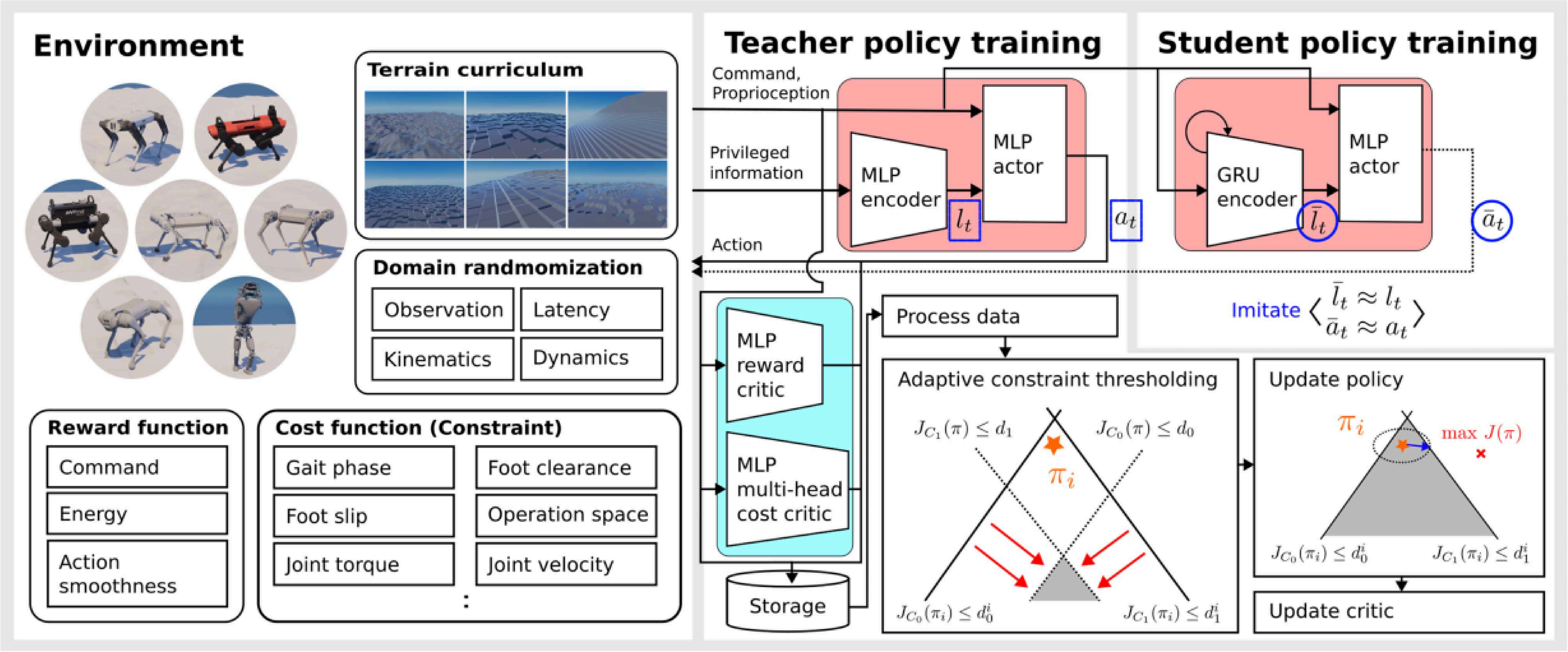
$$\text{Total Reward} = c_1 \times R_1 + c_2 \times R_2 + \cdots + c_n \times R_n$$

$$J_{Ck}(\pi) \le d_k \quad \text{for all } k$$

- There are multiple constraints, cannot combined into a single scalar value like the reward.
- Instead of training completely separate neural networks for each constraint, the multihead architecture allows the network to share a common backbone.
- A single neural network estimates all constraint values.



Framework



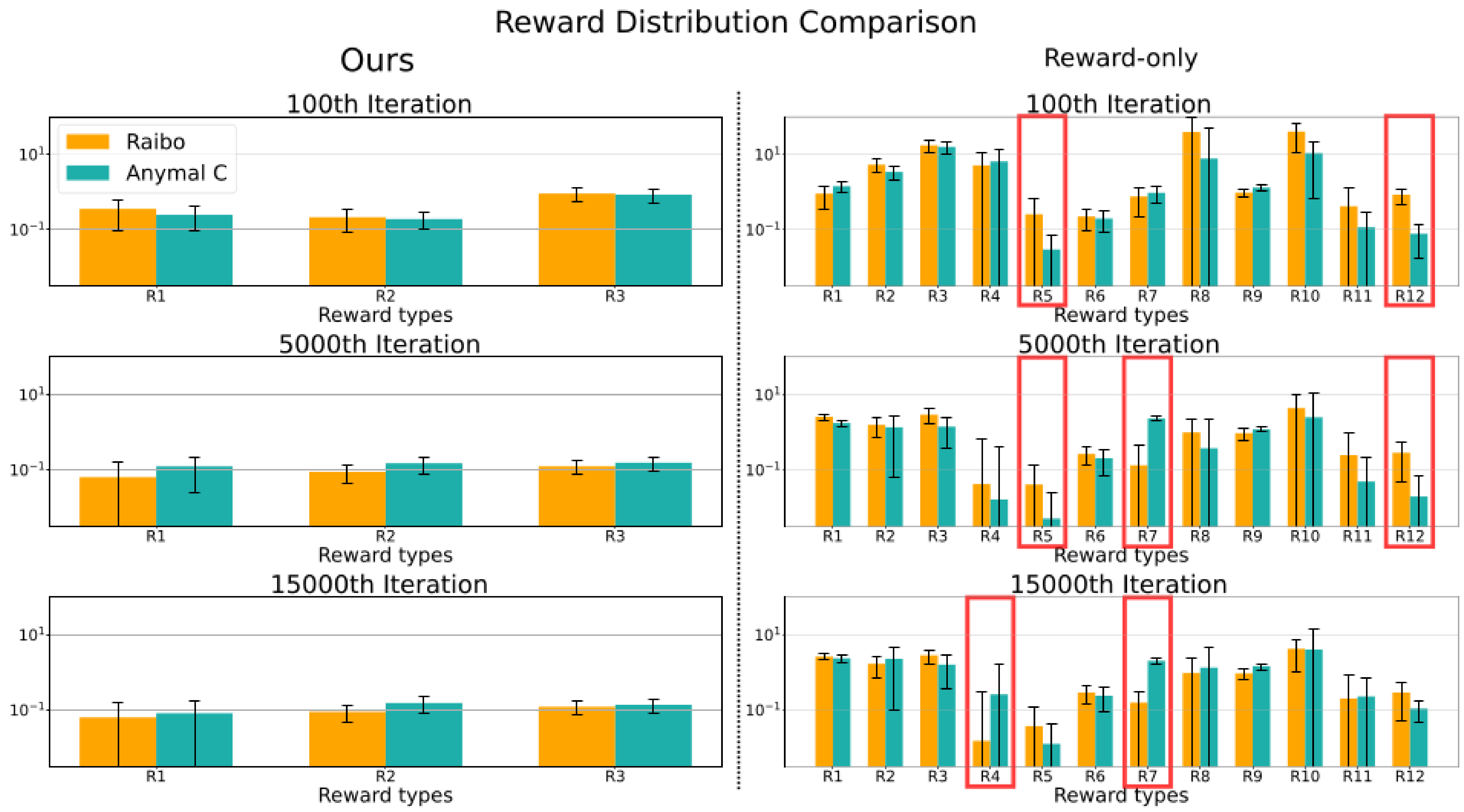
Environment
Terrain curriculum
Domain randmomization
Observation
Latency
Kinematics
Dynamics
Reward function
Command
Energy
Action smoothness
Cost function (Constraint)
Gait phase
Foot clearance
Foot slip
Operation space
Joint torque
Joint velocity
Teacher policy training
Command, Proprioception
Privileged information
Action
MLP encoder
MLP actor
l_t
a_t
Student policy training
GRU encoder
MLP actor
$\bar{l}_t$
$\bar{a}_t$
Imitate $\langle \bar{l}_t \approx l_t, \bar{a}_t \approx a_t \rangle$
MLP reward critic
MLP multi-head cost critic
Storage
Process data
Adaptive constraint thresholding
$J_{C_1}(\pi) \leq d_1$
$J_{C_0}(\pi) \leq d_0$
π_i
$J_{C_0}(\pi_i) \leq d_0^i$
$J_{C_1}(\pi_i) \leq d_1^i$
Update policy
max $J(\pi)$
Update critic

Comparison with Reward-only framework

- Generalizability
  - Tested by transferring it from Raibo to ANYmal, which has different physical properties.
    - Reward-Only Framework
      - Poor locomotion performance when transferred to the ANYmal.
      - Did not generalize well.
    - Proposed Framework
      - Explicit constraints helped ensure that the motion style was more consistent across different robots.

- Performance & Engineering Effort
    - Reward-Only Framework
      - When transferring the policy to a different robot, different physical properties led to significant changes in the reward signal distributions.
      - Time-consuming and needs to be repeated for each new robot or task.
    - Proposed Framework
      - Thus constraints correspond directly to physical limits, significantly reduced the need for extensive reward engineering.

Comparison with Reward-only framework



Comparison with Reward-only framework