Not only Reward but Also Constraint 리뷰

2024.10.04

이제희

INTRODUCTION

Key Points

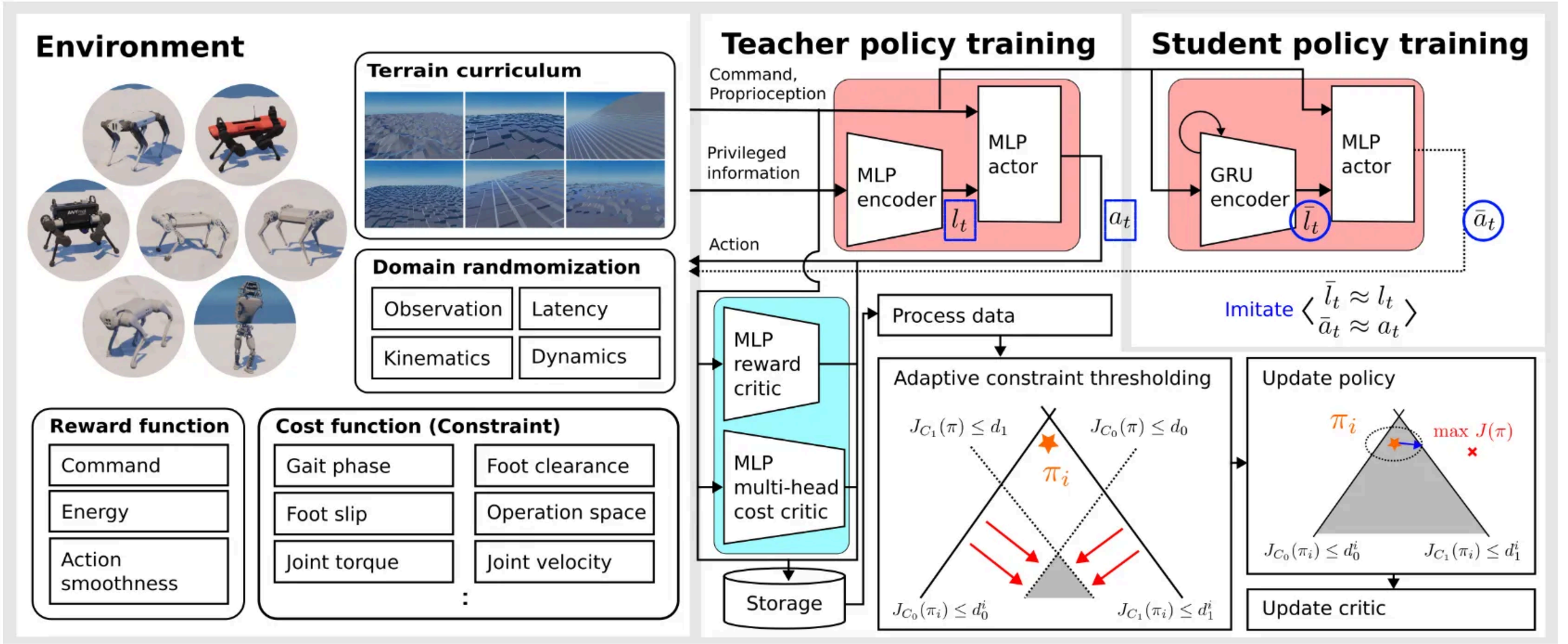
- **보상**과 **제약 조건**을 모두 포함한 **강화학습 프레임 워크 제시**
  - **물리적 속성 기반 제약 설정** → **파라미터 튜닝 난이도 완화**
    - **3개의 reward**, **11개의 제약 조건** 구성
    - 다양한 로봇 프레임에 적용 가능
  - **여러 제약조건 최적화를 위한 아래 트릭들을 적용**
    - CPO: Trust Region Approximation
    - IPO: logarithmic barrier functions
    - etc: GAE, Adaptive constraint thresholding, Multi-head cost value function

r_c	명령 추적 보상(command tracking)
r_τ	관절 토크 보상(joint torque)
r_s	행동 부드러움 보상(action smoothness)
c_{jp}	관절 각도(Joint position)
c_{jv}	관절 속도(Joint velocity)
c_{jt}	관절 토크(Joint torque)
c_{bc}	몸 접촉(Body contact)
c_{com}	프레임 무게 중심(Center of Mass frame)
c_{gp}	보행패턴(Gait pattern)
c_{ov}	직교 속도(Orthogonal velocity)
c_{cv}	접촉 속도(Contact velocity)
c_{fc}	발을 들어올리는 정도(Foot clearance)
c_{fh}	발 높이 제한(Foot height limit)
c_{sym}	대칭 제약(symmetric constraint)

〈적용 로봇〉

METHOD

Overview



METHOD

METHOD

Environment Setting - 평균 제약(Average Constraints)

비용 함수 형태

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

평균값이 임계값 D_k 이하가 되도록 표현

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

적용된 5개 제약

c_{ov}	직교 속도(Orthogonal velocity)
c_{cv}	접촉 속도(Contact velocity)
c_{fc}	발을 들어올리는 정도(Foot clearance)
c_{fh}	발 높이 제한(Foot height limit)
c_{sym}	대칭 제약(symmetric constraint)

METHOD

Environment Setting - 평균 제약(Average Constraints) - 대칭 제약(symmetric constraint)

대칭 제약(symmetric constraint) [52]

$$L_{sym} := \mathbb{E}_{s\sim d^{\pi}}\left[\left\|\mu_{\theta}(s) - \Psi_a(\mu_{\theta}(\Psi_s(s))\right\|_1\right] \leq d_{sym}$$

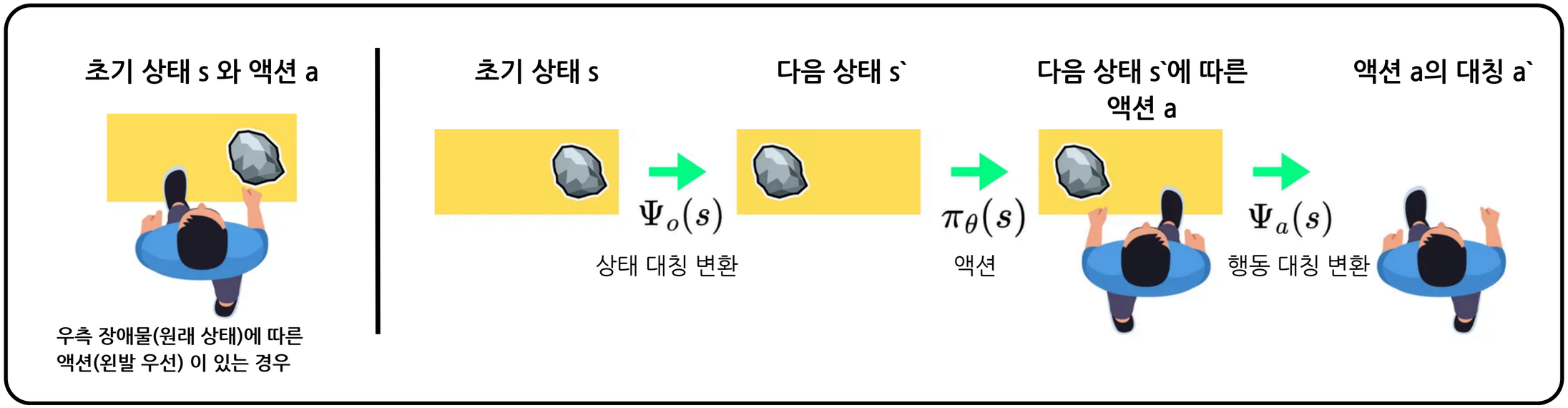
Mirror Symmetry Loss

$$L_{sym}(\theta) = \sum_{i=0}^{B} \|\pi_{\theta}(s_i) - \Psi_a(\pi_{\theta}(\Psi_o(s_i)))\|^2$$

완전 대칭 시

$$\pi_{\theta}(s) = \Psi_a(\pi_{\theta}(\Psi_o(s)))$$

- $\Psi_a(s)$ 행동 대칭 변환 함수
- $\Psi_o(s)$ 상태 대칭 변환 함수



[52] W. Yu, G. Turk, and C. K. Liu, “Learning symmetric and low-energylocomotion,” ACM Transactions on Graphics (TOG), vol. 37, no. 4, pp.1–12, 2018

METHOD

Environment Setting - 확률적 제약(Probabilistic Constraints)

비용 함수 형태

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

바람직하지 않는 영역에 들어갈 확률을 D_k 이하로 설정

- $D_k \in [0, 1]$ k 번째의 제약 조건

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

적용된 6개 제약

c_{jp}	관절 각도(Joint position)
c_{jv}	관절 속도(Joint velocity)
c_{jt}	관절 토크(Joint torque)
c_{bc}	몸 접촉(Body contact)
c_{com}	프레임 무게 중심(Center of Mass frame)
c_{gp}	보행패턴(Gait pattern)

평균 제약, 확률 제약을 선정 기준

(i.e) 미끄러운 빙판위 로봇

평균 제약: 평균적으로 0.01 m/s 을 유지하나 1m/s 미끄러짐이 발생

- 최대한 0.01m/s 주변을 유지하게 학습이 이루어짐
- 장기적으로 평균 0.01 m/s 를 유지하게 됨

확률 제약: 10%의 확률로 미끄러짐을 허용

- 0.01 m/s를 유지하게 학습이 이루어지나 10%의 확률로 미끄러짐이 발생 할 수 있음
- 이는 **치명적인 결과**를 초래 할 수 있음

METHOD

Environment Setting - 확률적 제약(Probabilistic Constraints) - 보행패턴(Gait pattern)

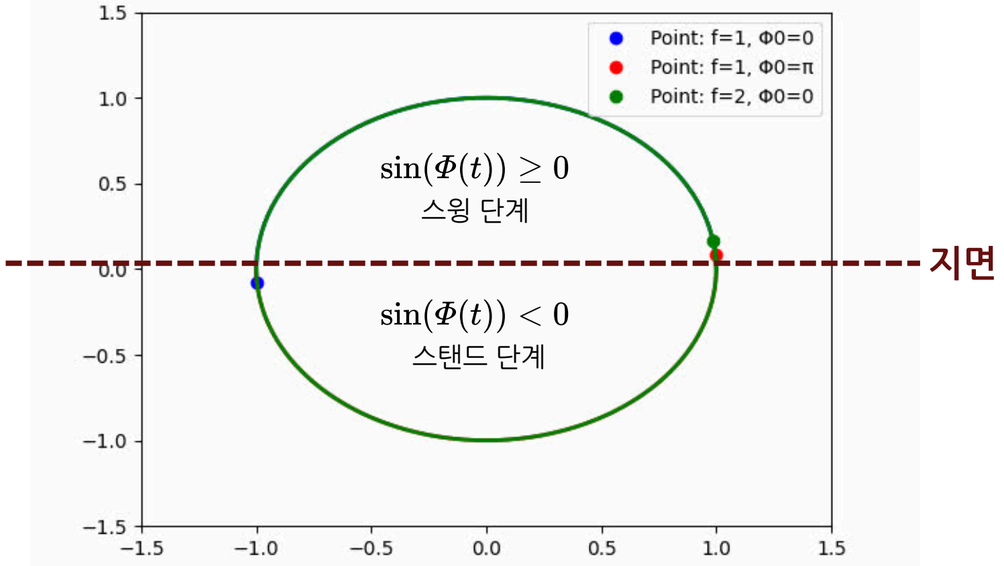
c_{gp} **보행패턴(Gait pattern)**

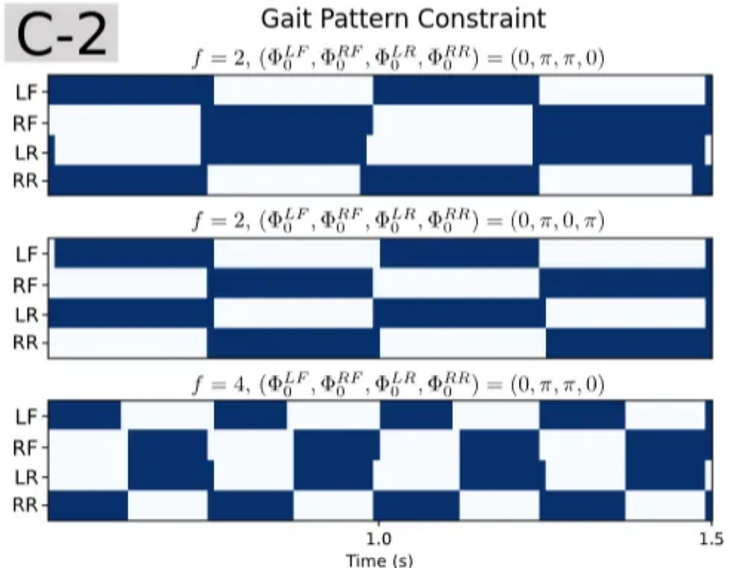
- internal 타이머를 통해 **스윙 / 스탠드 상태를 예측** 하고, 예측 상태와 일치 시 0, 불일치 시 **패널티** $1/n_{\text{legs}}$
- defined as **cosine and sine pairs** $\cos(\Phi(t)), \sin(\Phi(t))$

$$\Phi(t) = 2\pi f t + \Phi_0$$

- **initial phase offset** parameter
  - i.e trot

$$\Phi_0 = ([0, \pi, \pi, 0])$$





initial phase, frequency에 따른 보행 단계

METHOD

Environment Setting - 보상(Reward) - 관절 토크 보상

$$r = r_c + r_\tau + r_s$$

명령 추적 보상 (command tracking) | 관절 토크 보상 (joint torque) | 행동 부드러움 보상 (action smoothness)

$$r_\tau = -k_\tau \|\tau\|^2$$

scaling factor - **모션을 보고 튜닝**
※ 라이보의 경우 1

$$k_\tau = \hat{k}_\tau \cdot \frac{\bar{m}}{m} = \left(s_\tau \cdot \bar{k}_\tau\right) \cdot \frac{\bar{m}}{m}$$

$\bar{m}$ ← Raibo 무게: 27kg

m ← 적용 로봇 무게

$\bar{k}_\tau$ ↑ Raibo 토크 보상 상수

METHOD

Environment Setting - 보상(Reward) - 행동 부드러움 보상

$$r = r_c + r_\tau + r_s$$

명령 추적 보상 (command tracking) | 관절 토크 보상 (joint torque) | 행동 부드러움 보상 (action smoothness)

속도와 가속도가 너무 크지 않게 해준다

- real-world 에서 **모터 진동 감소** 역할

$$r_s = -k_s \left(\left\| \underbrace{q_t^{des} - q_{t-1}^{des}}_{\text{속도}} \right\|^2 + \left\| \underbrace{q_t^{des} - 2q_{t-1}^{des} + q_{t-2}^{des}}_{\text{가속도}} \right\|^2 \right)$$

METHOD

알고리즘 개요

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

$$\text{s.t. } J_{C_k}(\pi) \leq d_k \quad \forall k \in \{1, \ldots, K\}$$

신뢰 영역에서의 근사화
(Trust Region Approximation)

$$\pi_{i+1} = \arg\max_{\pi \in \Pi_\theta} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}(s, a) \right]$$

$$\text{s.t. } J_{C_k}(\pi_i) + \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}_{C_k}(s, a) \right] \leq d_k \quad \forall k$$

$$\bar{D}_{KL}(\pi \parallel \pi_i) \leq \delta,$$

로그 배리어 함수
(logarithm barrier function)

TRPO로 최적화

$$\underset{\pi \in \Pi_\theta}{\text{maximize}} \quad \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}(s, a) \right]$$

$$+ \sum_{k=1}^{K} \log(d_k - \left(J_{C_k}(\pi_i) + \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}_{C_k}(s, a) \right] \right))/t$$

$$\text{s.t.} \quad \bar{D}_{KL}(\pi || \pi_i) \leq \delta.$$

METHOD

알고리즘 - Constrained Markov Decision Process(CMDP)

비용 함수를 **만족**하면서 **보상함수를 최대화** 하는 **정책**을 찾는것

보상 함수 (reward function)

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

비용 함수(cost function)

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

CMDP의 목적함수

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

$$\text{s.t. } J_{C_k}(\pi) \le d_k \quad \forall k \in \{1, \dots, K\}$$

METHOD

알고리즘 - CPO - 신뢰 영역에서의 근사화(Trust Region Approximation)

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

$$\text{s.t. } J_{C_k}(\pi) \leq d_k \quad \forall k \in \{1, \ldots, K\}$$

제약이 있는 정책 최적화 문제

$$\pi_{i+1} = \arg\max_{\pi \in \Pi_\theta} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}(s, a) \right]$$

$$\text{s.t. } J_{C_k}(\pi_i) + \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}_{C_k}(s, a) \right] \leq d_k \quad \forall k$$

$$\bar{D}_{KL}(\pi \,\|\, \pi_i) \leq \delta,$$

[48] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trustregion policy optimization,” in International conference on machinelearning. PMLR, 2015, pp. 1889–1897.

METHOD

알고리즘 - IPO - logarithm barrier function

$$\max_{\theta} \; L^{CLIP}(\theta),$$

$$\text{s.t. } J_{C_i}^{\pi^{\theta}} \leq \epsilon_i.$$

제약 조건의 표현

$$\hat{J}_{C_i}^{\pi_\theta} = J_{C_i}^{\pi_\theta} - \epsilon_i,$$

제약 조건 → 패널티 함수로 변환

$$\hat{J}_{C_i}^{\pi\theta} < 0$$

[43] Y. Liu, J. Ding, and X. Liu, “Ipo: Interior-point policy optimizationunder constraints,” in Proceedings of the AAAI conference on artificialintelligence, vol. 34, no. 04, 2020, pp. 4940–4947.

METHOD

알고리즘 - IPO - logarithm barrier function

지시 함수(Indicator Function) 도입

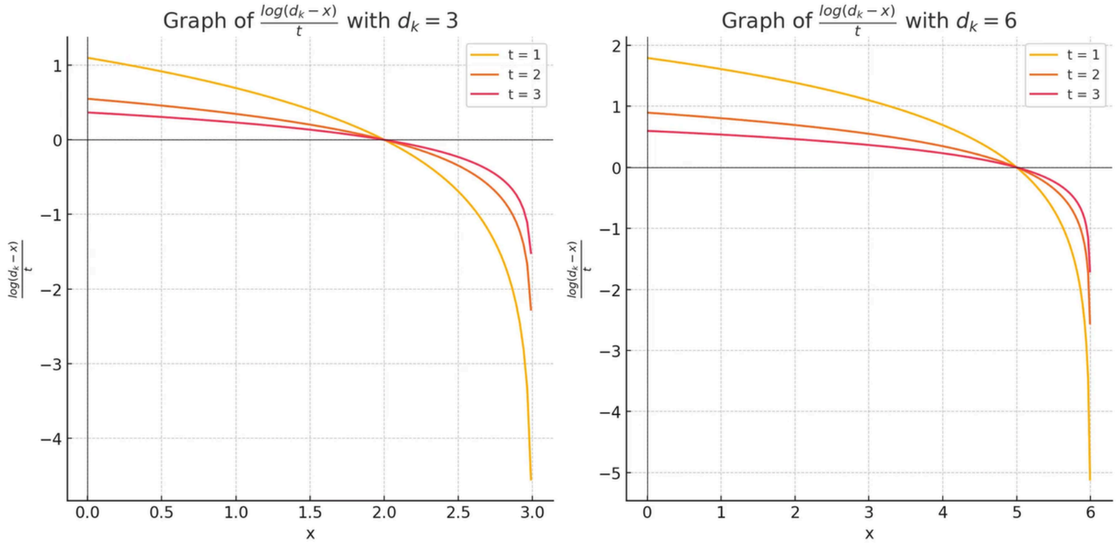
$$I(\hat{J}_{C_i}^{\pi_\theta}) = \begin{cases} 0 & \text{if } \hat{J}_{C_i}^{\pi_\theta} \leq 0, \\ -\infty & \text{if } \hat{J}_{C_i}^{\pi_\theta} > 0. \end{cases}$$

미분 하기 위해 로그 함수를 지시 함수(Indicator Function)로 사용

- 제약 조건을 잘 **만족할수록 0**
- 제약 조건 위반에 **가까울수록 - ∞**

$$\phi(\hat{J}_{C_i}^{\pi_\theta}) = \frac{\log(-\hat{J}_{C_i}^{\pi_\theta})}{t},$$

큰 t : 리워드와 *cost*가 증가한다, 단 수렴성이 떨어진다



[43] Y. Liu, J. Ding, and X. Liu, “Ipo: Interior-point policy optimizationunder constraints,” in Proceedings of the AAAI conference on artificialintelligence, vol. 34, no. 04, 2020, pp. 4940–4947.

METHOD

알고리즘 - IPO - logarithm barrier function

$$\max_{\theta} \; L^{CLIP}(\theta),$$

$$\text{s.t. } J_{C_i}^{\pi^{\theta}} \leq \epsilon_i.$$

⬇

$$\max_{\theta} \; L^{IPO}(\theta),$$

$$L^{IPO}(\theta) = L^{CLIP}(\theta) - \sum_{i=1}^{m} \phi\left(\hat{J}_{C_i}^{\pi^{\theta}}\right).$$

미분을 하기위해 지시 함수(Indicator Function) 를 로그 함수로 근사화

- 제약 조건을 잘 **만족할수록 0**
- 제약 조건 위반에 **가까울수록 - ∞**

$$\phi(\hat{J}_{C_i}^{\pi_\theta}) = \frac{\log(-\hat{J}_{C_i}^{\pi_\theta})}{t},$$

제약 조건이 없는 최적화 문제로 변환 되었다

[43] Y. Liu, J. Ding, and X. Liu, “Ipo: Interior-point policy optimizationunder constraints,” in Proceedings of the AAAI conference on artificialintelligence, vol. 34, no. 04, 2020, pp. 4940–4947.

METHOD

알고리즘 - IPO - logarithm barrier function

IPO

$$\max_{\theta} \; L^{CLIP}(\theta),$$

$$\text{s.t. } J_{C_i}^{\pi_\theta} \leq \epsilon_i.$$

➡

$$\max_{\theta} \; L^{IPO}(\theta),$$

$$L^{IPO}(\theta) = L^{CLIP}(\theta) + \sum_{i=1}^{m} \frac{\log(-\hat{J}_{C_i}^{\pi_\theta})}{t},$$

Not only Reward but Also Constraint

$$\pi_{i+1} = \arg\max_{\pi \in \Pi_\theta} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}(s,a) \right]$$

$$\text{s.t. } J_{C_k}(\pi_i) + \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A_{C_k}^{\pi_i}(s,a) \right] \leq d_k \quad \forall k$$

$$\bar{D}_{KL}(\pi \parallel \pi_i) \leq \delta,$$

신뢰 영역에서의 근사화 식
(Trust Region Approximation)

➡

$$\underset{\pi \in \Pi_\theta}{\text{maximize}} \quad \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}(s,a) \right]$$

$$+ \sum_{k=1}^{K} \log\left(d_k - \left(J_{C_k}(\pi_i) + \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A_{C_k}^{\pi_i}(s,a) \right] \right)\right)/t$$

$$\text{s.t.} \quad \bar{D}_{KL}(\pi || \pi_i) \leq \delta.$$

로그 배리어 함수 사용 후
(logarithm barrier function)

METHOD

알고리즘 - TRPO

최적화를 TRPO로 진행
<논문 발췌>
IPO의 원 논문과 달리 **최적화를 TRPO**로 하였다
PPO를 테스트했을때 적용은 쉬웠으나, **안정성이 떨어지는 트레이드 오프**가 있었다

in the original paper [43], the IPO objective (10) is optimized with PPO [49]. However, **we utilized Trust Region Policy Optimization (TRPO)**

We also experimented with the PPO [49] optimizer and found several algorithmic advantages, such as **ease of implementation** and the availability to utilize recurrent neural networks, albeit with the **trade-off of training stability**

어떻게 TRPO 로 최적화를 시켰는지는 추후 업데이트 예정

$$\underset{\pi \in \Pi_\theta}{\text{maximize}} \quad \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}(s,a) \right]$$

$$+ \sum_{k=1}^{K} \log(d_k - \left(J_{C_k}(\pi_i) + \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi} \left[A^{\pi_i}_{C_k}(s,a) \right] \right))/t$$

$$\text{s.t.} \quad \bar{D}_{KL}(\pi || \pi_i) \leq \delta.$$

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Prox-imal policy optimization algorithms,” arXiv preprint arXiv:1707.06347,2017

METHOD

최적화 트릭 - GAE(Generalized Advantage Estimation)

어드밴티지 (1-스탭 관계식)

- 편향 ↑
- 분산 ↓

$$A^{\pi_\theta}(s_t, a) = r + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$$

어드밴티지 (몬테카를로)

- 편향 ↓
- 분산 ↑

$$A^{\pi_\theta}(s_t, a) = \sum_{k=t}^{\infty} \gamma^{k-t} r^k - V^{\pi_\theta}(s_t)$$

어드밴티지 (N -스탭 관계식)

- 편향: N 과 반비례
- 분산: N 과 비례

$$A^{\pi_\theta}(s_n, a) = \sum_{k=1}^{t+n-1} \gamma^{k-t} r^k + \gamma^n V^{\pi_\theta}(s_{t+n}) - V^{\pi_\theta}(s_t)$$

어드밴티지 (GAE)

- λ 에 따라 분산과 편향을 조절 할 수 있다

$$A_\pi^{GAE}(s_t, a) = \sum_{k=t}^{\infty} (\gamma\lambda)^{k-t} \delta_k$$

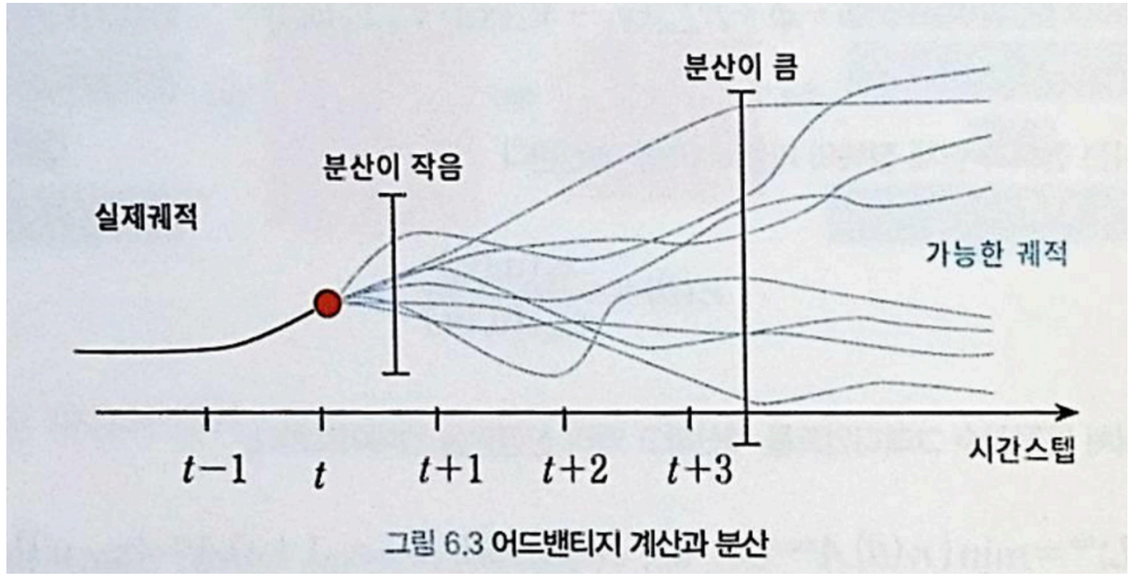


그림 6.3 어드밴티지 계산과 분산

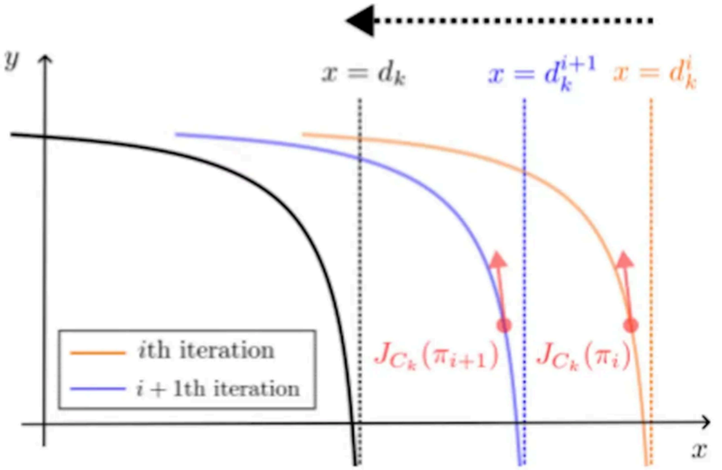
[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Prox-imal policy optimization algorithms,” arXiv preprint arXiv:1707.06347,2017

METHOD

최적화 트릭 - 적응적 제약 임계값(Adaptive constraint thresholding)

초기값은 큰 랜덤성(exploration)을 갖기 때문에 제약을 엄격하게 적용하면 학습을 하지 못할 가능성이 있다.
이를 줄이기 위해, 단계적으로 제약을 강화시켜 목표에 달성시키게 한다

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



위반 상황 예

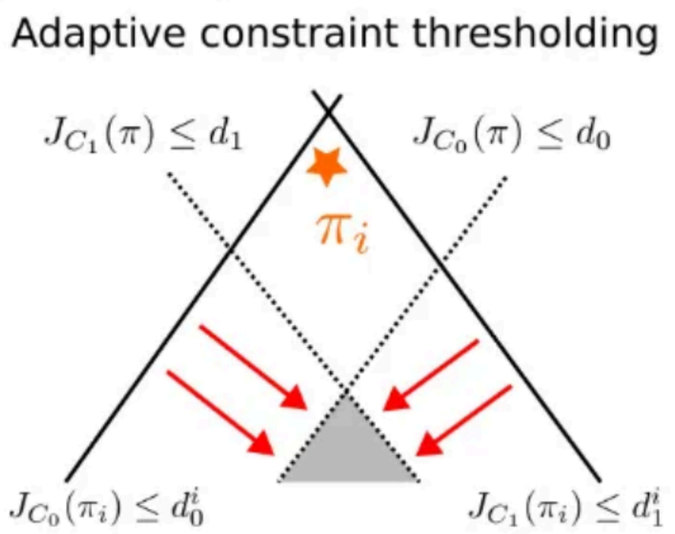
- 기본 임계값: 0.05
- 확대 비율: 0.1
- 현재 위반 상황: 0.08

임계값 계산 = max(0.05, 0.08 + 0.1 * 0.05) = 0.085
임계값이 0.05 에서 **0.085로 완화되었다**

준수 상황 예

- 기본 임계값: 0.05
- 확대 비율: 0.1
- 현재 위반 상황: 0.04

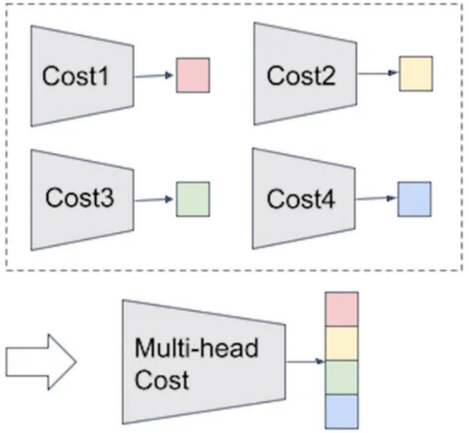
임계값 계산 = max(0.05, 0.04 + 0.1 * 0.05) = 0.05
0.05로 복귀 되었다

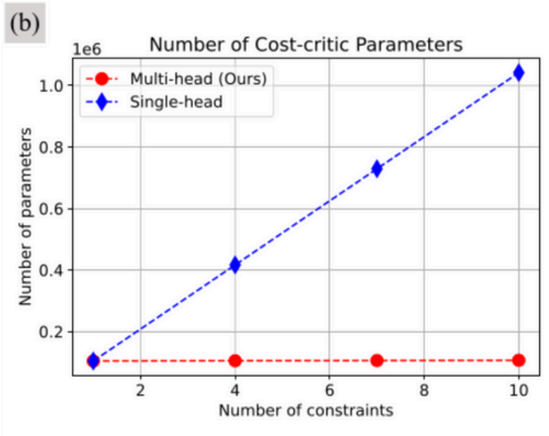


METHOD

최적화 트릭 - Multi-head cost value function

- 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.





1. **multihead(0.1 M) - 1개**
   - 속도: 가장 빠름
   - loss: 중간
2. **single-head(1 M) - 10개**
   - 속도: 가장 느림
     - loss: 가장 적음
       - 10개의 신경망(1 M)들이 각 예측을 하므로 정확하다
3. **single-head(0.1 M) - 10개:**
   - 속도: 중간
   - loss: 가장 큼

TABLE VI

MULTI-HEAD COST CRITIC EVALUATION (# CONSTRAINTS = 10)

	# parameters	Computation time [s]	Prediction loss
Multi-head (Ours)	**0.1 M**	**0.23** (10.22 %)	0.468 (± 0.120)
Single-head	1 M	0.56 (24.89 %)	**0.435** (± 0.139)
Single-head (Reduced)	**0.1 M**	0.33 (14.67 %)	0.534 (± 0.205)

* Percentage in the "Computation time" indicates the ratio of time spent on cost-critic versus others (e.g., simulation), where others are identical for all three methods.

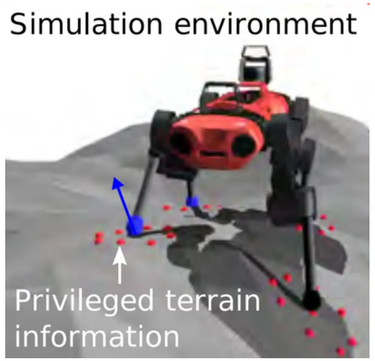
METHOD

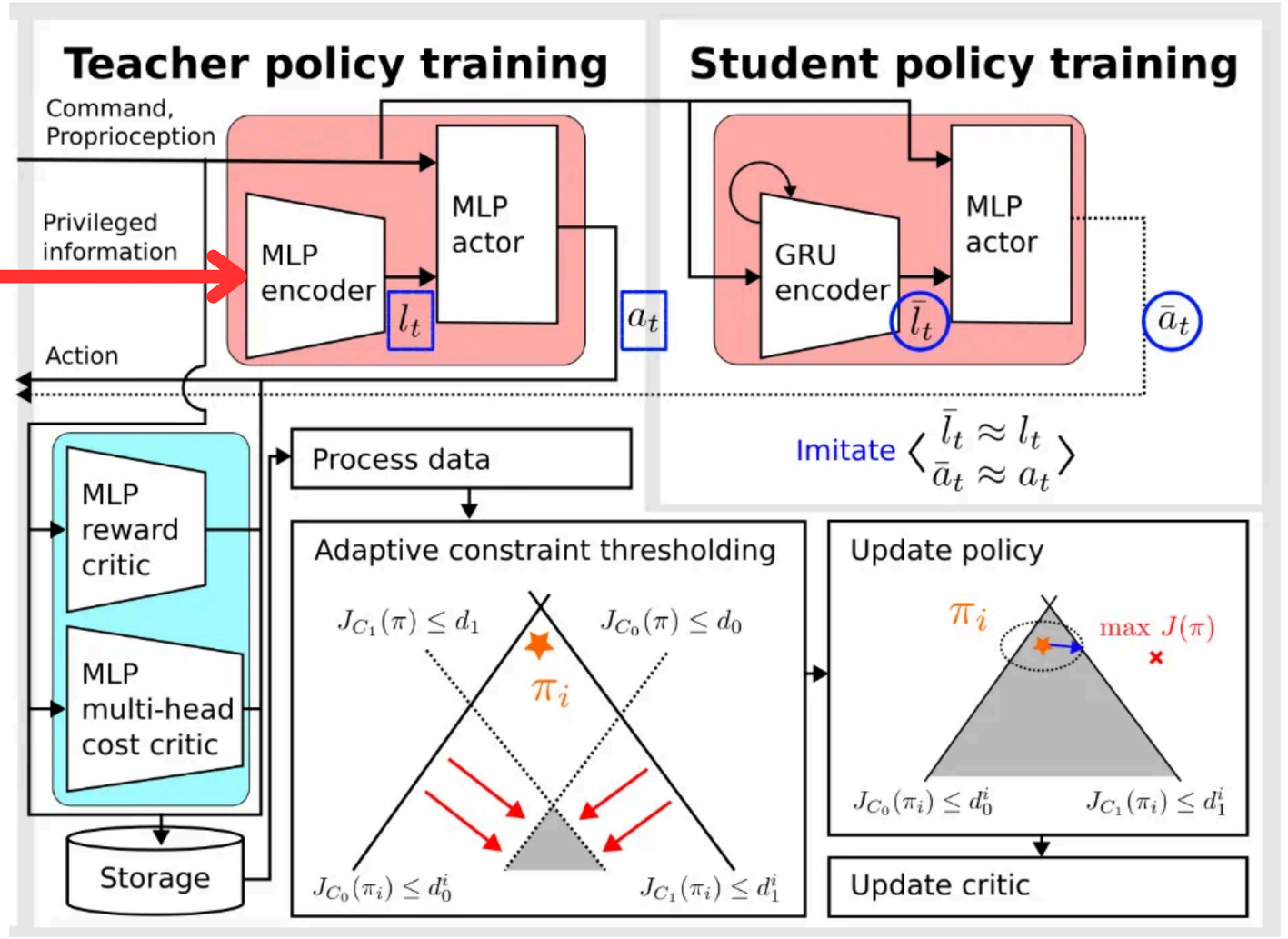
Training Framework

특권 정보(privileged information)

- terrain friction coefficient
- body height with respect to the ground
- body linear vel
- body contact state,
- foot contact impulse,
- foot airtime
- foot clearance
- terrain profiles: circular height scan near each foot

p_t



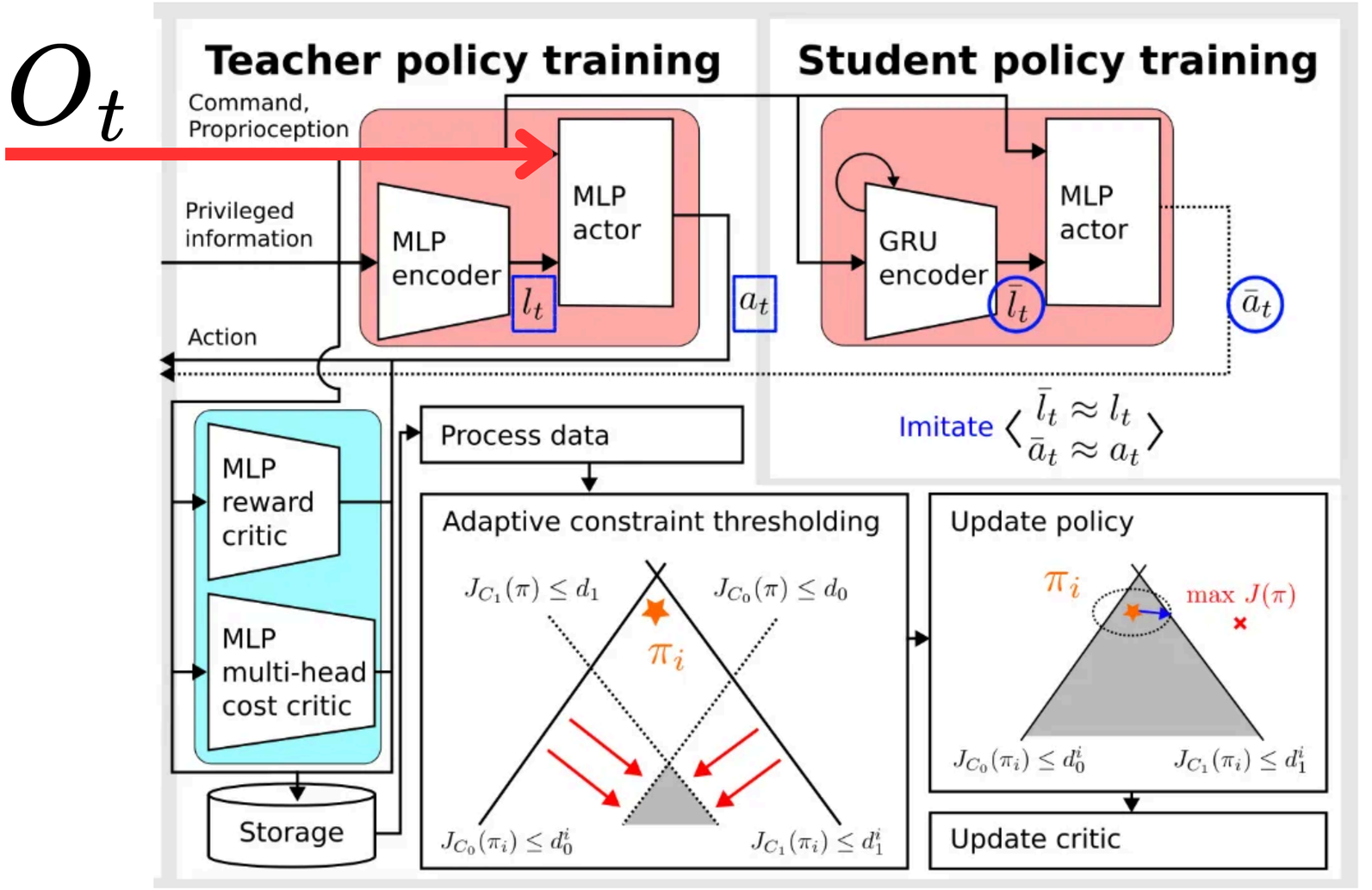


METHOD

Training Framework

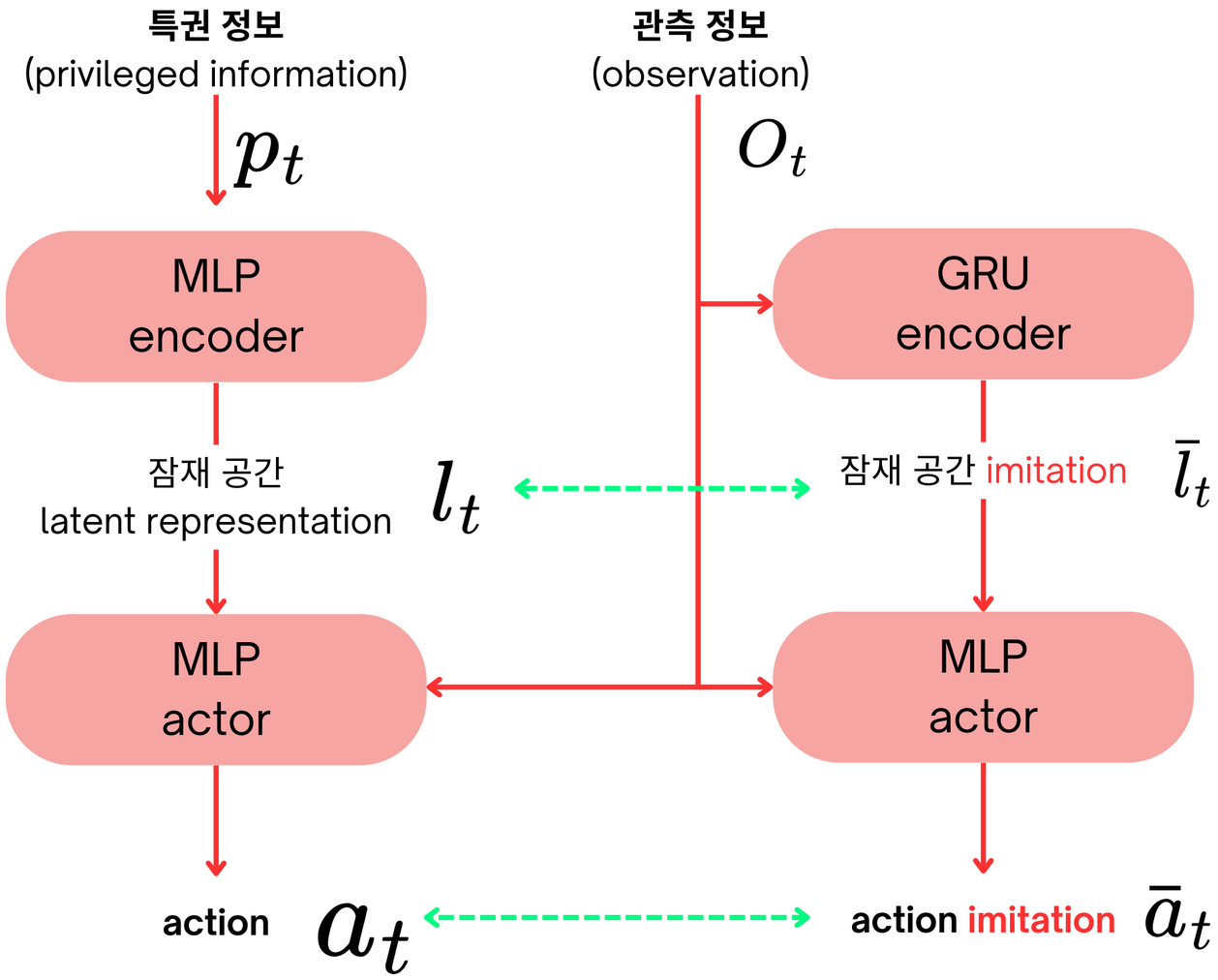
관측 정보(observation)

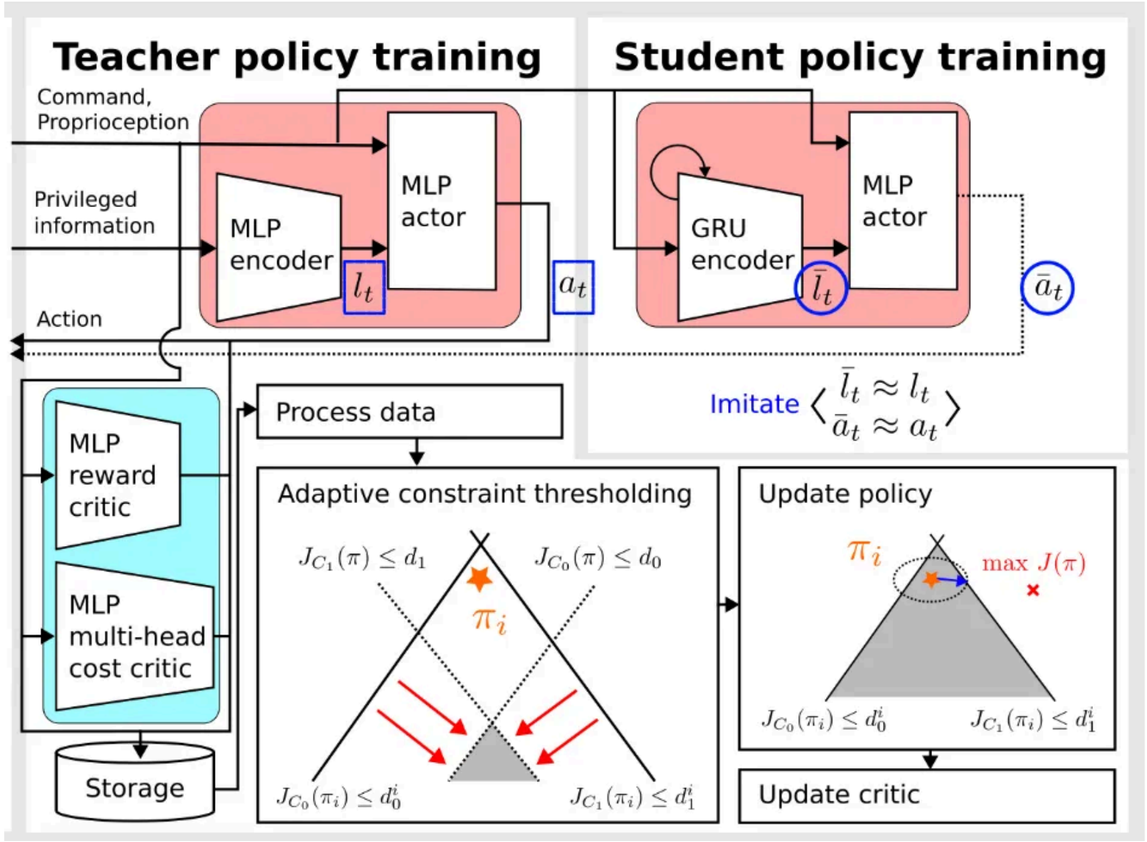
- given velocity command,
- body orientation, body angular velocity
- current joint positions and velocities,
- joint position errors
- 속도 히스토리 (-0.02s, -0.04s, -0.06s)
- 액션 histories (-0.01s, -0.02s)
- central phases for each leg



METHOD

Training Framework - Teacher & Student

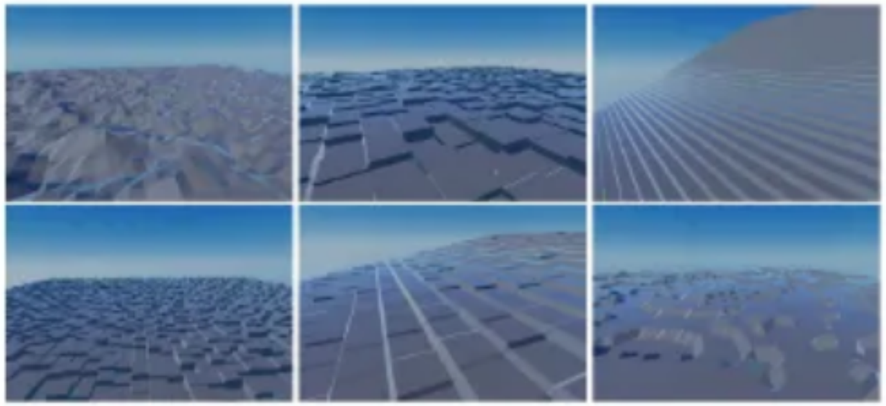




METHOD

Training Frame work

적응형 지형 커리큘럼 (Adaptive Terrain Curriculum)



Domain randomization

- 관찰 잡음 (observation noise)
- 모터 마찰 (motor frictions)
- PD 제어기 이득 (PD controller gains)
- 발 위치 (foot positions)
- 충돌 기하학 (collision geometry)
- 지면 마찰 (ground friction)
- 제어 지연 (control latency)

RESULT

RESULT

Framework Evaluation

Anymal 과 같이 다리가 반대여도, Atlas처럼 2족보행 이여도 **constraint 수정이 크게 필요하지 않았다**

s_τ 수정이 필요했는데, 이는 에너지 사용(**토크**)은 **단순 무게 비례가 아니기 때문**이다

TABLE III

REWARD COEFFICIENTS FOR SEVERAL LEGGED ROBOTS

	k_c	$\hat{k}_\tau$	k_s
Raibo	10.	0.005	0.06
Hound	10.	0.0025	0.06
Go1	10.	0.025	0.06
Mini-cheetah	10.	0.03	0.06
Anymal B	10.	0.005	0.06
Anymal C	10.	0.003	0.06
Atlas	10.	0.001	0.06

RECALL

scaling factor - 모션을 보고 상수값을 변경한다

※ 라이보의 경우 1

$$k_\tau = \hat{k}_\tau \cdot \frac{\bar{m}}{m} = \left(s_\tau \cdot \bar{k}_\tau\right) \cdot \frac{\bar{m}}{m}$$

Raibo 무게: 27kg

Raibo 토크 보상 상수

적용 로봇 무게

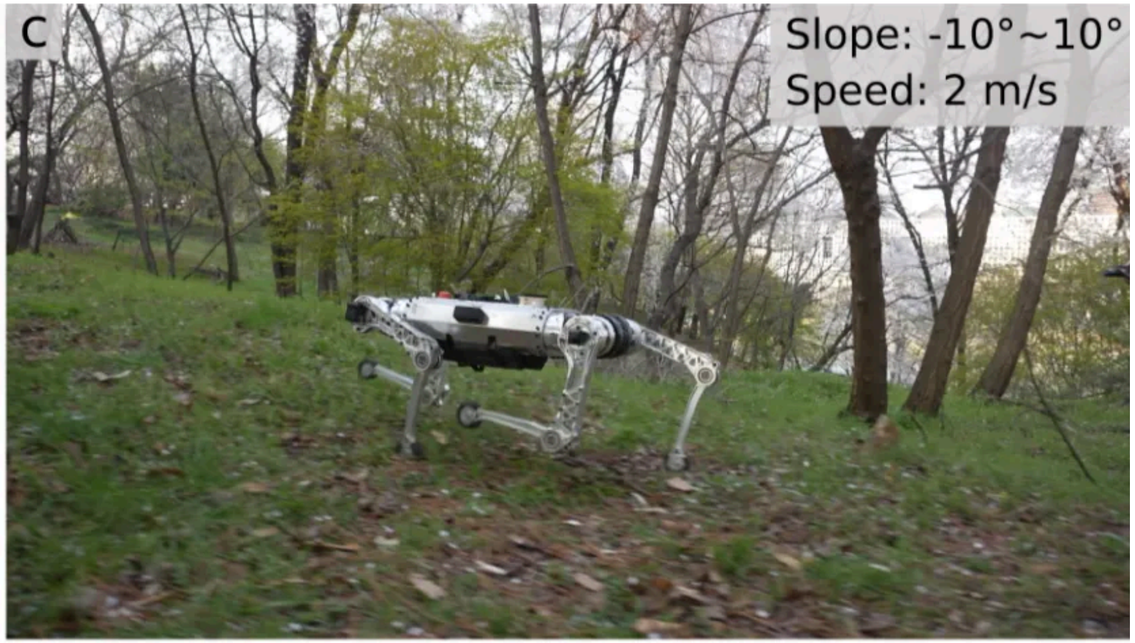
RESULT

Implementation detail

구현: 파이토치와 **Safe-RL-Algos**활용
시뮬레이션 환경: RaiSim 사용
학습 시간

- 24시간(교사) + 8 시간(학생)
- AMD Ryzen9 5959X + RTX 3070

locomotion control policy: 100 Hz
deployment: Eigen library + C++



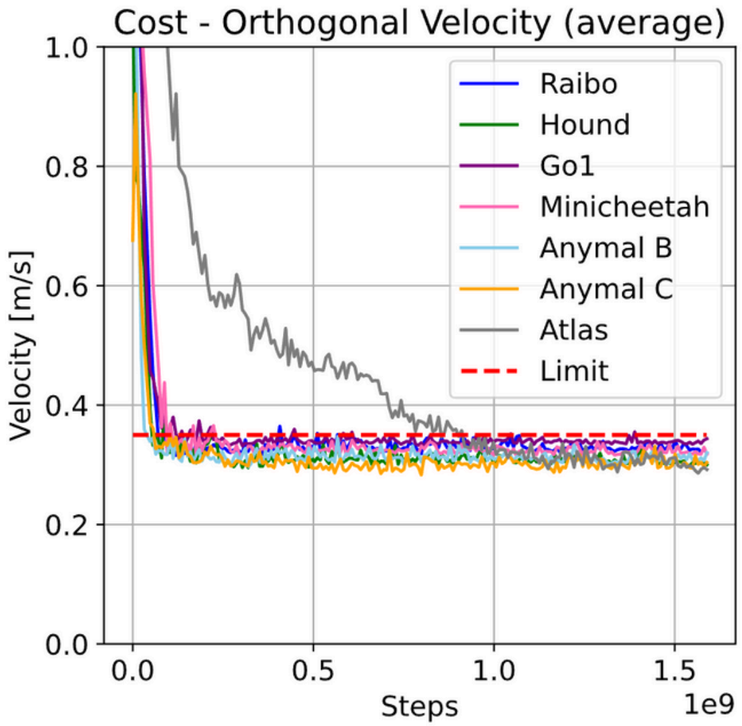
수풀 지형 2m/s

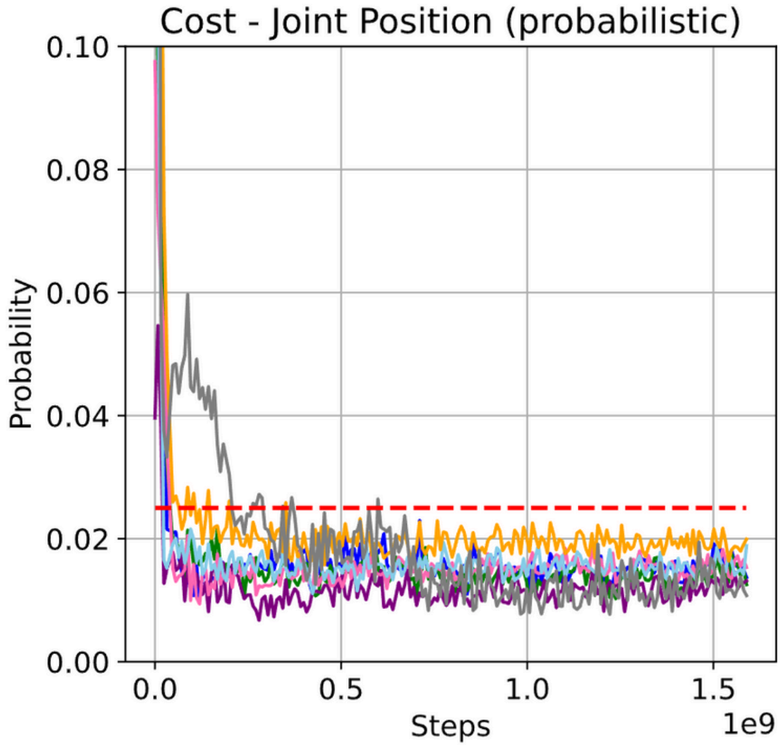
극복 각도 32 deg 1m/s, 이전 논문 대비 2배 상승

RESULT

Framework Analyzation - 1

직교속도/관절 위치가 **제약을 만족함**을 확인 할 수 있다



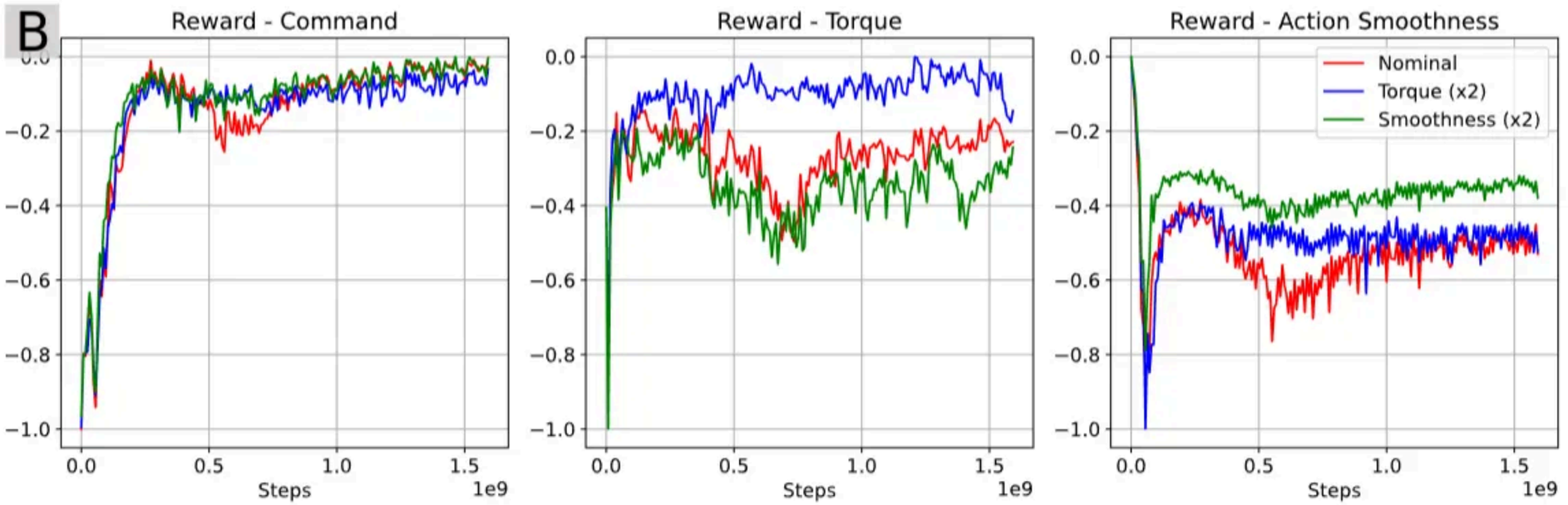


RESULT

Framework Analyzation - 1

목표 민감성(Objective sensitivity) - 보상 계수 변경 시, 학습에 실제 반영이 되는지

1. r_{τ} (관절 토크 보상)을 2배 늘린 경우 -> 에너지 2배 적게 사용
2. r_{s} (action smotthness)를 2배 늘린 경우 -> 그래프에 반영



RESULT

Framework Analyzation - 2

제약 조건 만족(Constraint satisfaction) - 보상/제약 계수 변경시에도 제약이 항상 만족 하는지

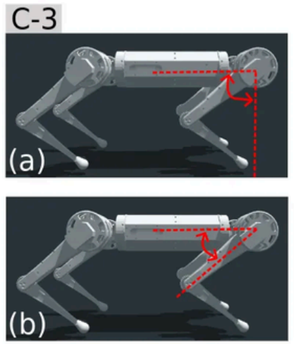
1. r_τ (관절 토크 보상)을 2배 늘린 경우
2. joint angle을 0 ~ -90 deg → -45deg → -90 deg

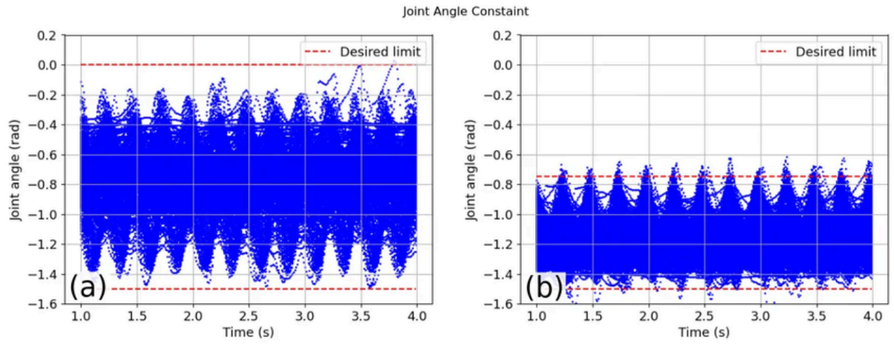
TABLE II
CONSTRAINT SATISFACTION RESULTS

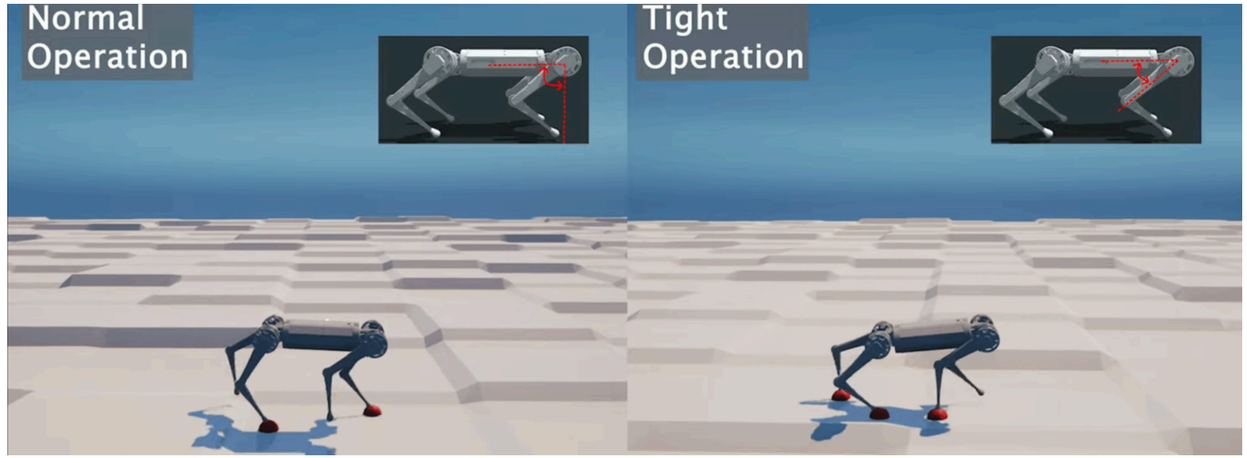
		c_{jp}	c_{jv}	c_{jt}	c_{bc}	c_{com}	c_{gp}	c_{ov} [m/s]	c_{cv} [m/s]	c_{fc} [m]	c_{fh} [m]	c_{sym}
	Limit	0.025	0.025	0.025	0.025	0.025	0.25	0.35	-	-	-	0.1
Sec. VI-B	Raibo	0.015	0.004	0.001	0.002	0.009	0.16	0.32	0.15 [0.2]	-0.11 [-0.07]	0.09 [0.11]	0.07
	Hound	0.014	0.020	0.000	0.003	0.002	0.17	0.30	0.13 [0.2]	-0.09 [-0.07]	0.07 [0.11]	0.07
	Go1	0.012	0.007	0.000	0.008	0.001	0.16	0.33	0.10 [0.2]	-0.08 [-0.05]	0.06 [0.09]	0.07
	Mini-cheetah	0.015	0.000	0.001	0.002	0.001	0.13	0.32	0.10 [0.2]	-0.08 [-0.05]	0.06 [0.09]	0.06
	Anymal B	0.016	0.019	0.000	0.002	0.005	0.16	0.31	0.13 [0.2]	-0.09 [-0.07]	0.07 [0.11]	0.07
	Anymal C	0.019	0.021	0.002	0.004	0.005	0.16	0.30	0.11 [0.2]	-0.08 [-0.07]	0.06 [0.11]	0.07
	Atlas	0.011	0.019	0.004	0.008	0.008	0.12	0.29	0.35 [0.5]	-0.17 [-0.15]	0.12 [0.2]	0.08
Sec. VI-C	Torque (x2)	0.017	0.005	0.001	0.001	0.008	0.15	0.33	0.16 [0.2]	-0.13 [-0.07]	0.10 [0.11]	0.08
	Smoothness (x2)	0.013	0.005	0.001	0.001	0.009	0.14	0.32	0.15 [0.2]	-0.13 [-0.07]	0.10 [0.11]	0.07
Sec. VI-E	P3O [10], [44]	**0.025**	0.005	0.000	0.002	0.016	0.24	**0.35**	**0.2** [0.2]	-0.08 [-0.07]	0.09 [0.11]	**0.1**
	$t = 100$, $\alpha = 0.002$	0.015	0.004	0.002	0.003	0.009	0.17	0.32	0.15 [0.2]	-0.11 [-0.07]	0.09 [0.11]	0.07
	$t = 100$, $\alpha = 0.2$	0.015	0.004	0.002	0.002	0.011	0.18	**0.44**	0.17 [0.2]	-0.11 [-0.07]	0.09 [0.11]	0.07
	$t = 10$, $\alpha = 0.02$	0.004	0.001	0.000	0.000	0.002	0.11	0.24	0.07 [0.2]	-0.14 [-0.07]	0.07 [0.11]	0.05
	$t = 1000$, $\alpha = 0.02$	0.023	0.010	0.002	0.002	0.015	0.22	**0.45**	**0.2** [0.2]	-0.09 [-0.07]	0.09 [0.11]	0.09

*Results are after training with 1.6 billion time steps. Constraint thresholds are reported in the top row or inside the bracket. Bolded texts indicate violated or barely satisfied constraints.

c_{jp}	c_{jv}	c_{jt}	c_{bc}	c_{com}	c_{gp}	c_{ov} [m/s]	c_{cv} [m/s]	c_{fc} [m]	c_{fh} [m]	c_{sym}
0.025	0.025	0.025	0.025	0.025	0.25	0.35	-	-	-	0.1
0.017	0.005	0.001	0.001	0.008	0.15	0.33	0.16 [0.2]	-0.13 [-0.07]	0.10 [0.11]	0.08







RESULT

Framework Analyzation - 3

높이 및 gait 조정 후 real world deploy

gait: Φ_0 (initial phase offset parameter) 값 만 조정하였다

높이 조정: crawling,COM height를 조정

d experiments of generating multiple motions by varying constraints. A robot walking with a trotting gait (top) was changed to locomote
gait (middle) or crawl beneath an overhanging obstacle (bottom) by modifying the gait pattern constraint or COM height with joint angle

RESULT

기존 보상 전용 프레임워크와의 비교(Comparison with the Reward-only Framework)

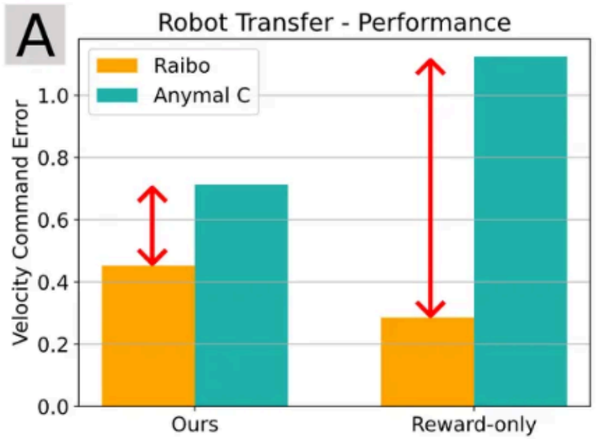
기존 대비 높은 범용성을 확인 할 수 있다

- 실험 방법: 12개의 리워드텀으로 TRPO를 라이보로 정책 학습 후, anymal-c에 적용

명령 추적

Anymal-c로 이전 시 명령 추적이 매우 낮았다

- 그래프를 보았을때, Anymal-C의 reward-only 속도 추적 에러 가 굉장히 크다



Raibo의 경우 Reward-only의 속도 에러가 더 낮은데 constraint **추가 알고리즘 성능이 더 안좋은것 아닌가?**

- 이는 **명령 추적 보상 함수**의 차이

해당 논문에서는 단순화를 위해 명령 추적 보상을 **이차함수**로 나타내었음.

$$r_c = -k_c \left(\|cmdv_{xy} - V_{xy}\|^2 + (cmdw_z - w_z)^2 \right)$$

보상 전용 프레임 워크에서는 **지수 함수**를 사용

$$r_c = k_c \left(\exp\left(-\|cmdv_{xy} - V_{xy}\|^2 \right) + \exp\left(-1.5(cmdw_z - w_z)^2 \right) \right)$$

제약 조건 추가된 알고리즘에서 지수 함수를 커널 함수로 사용 시 **에러가 큰 차이가 없음을** 확인함

TABLE VII
VELOCITY COMMAND TRACKING PERFORMANCE COMPARISON

	Velocity command error	Torque
Ours + L2 (Raibo)	0.498 (± 0.308)	**24.769** (± 5.811)
Ours + Exp (Raibo)	0.367 (± 0.290)	29.489 (± 8.311)
Ours + Exp (Anymal C)	0.547 (± 0.380)	60.310 (± 14.978)
Reward-only (Raibo)	**0.359** (± 0.354)	30.773 (± 10.181)

RESULT

기존 보상 전용 프레임워크와의 비교(Comparison with the Reward-only Framework)

기존 대비 높은 범용성을 확인 할 수 있다

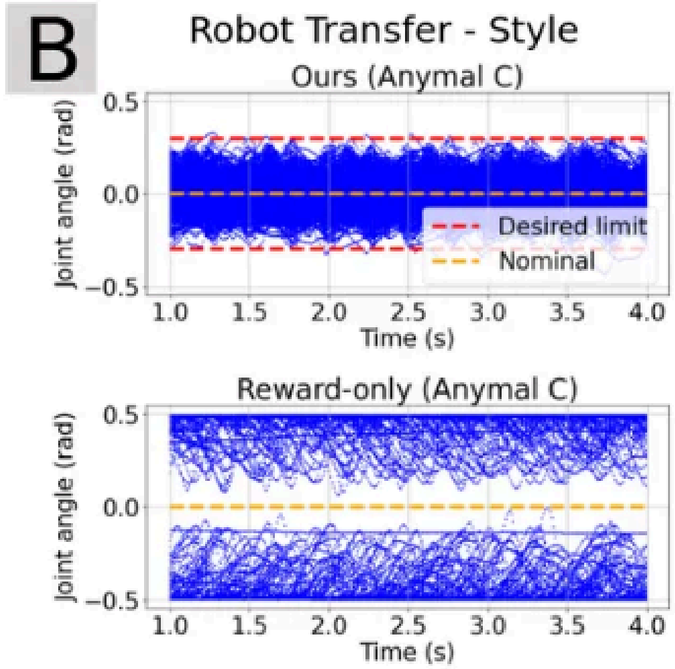
- 실험 방법: 12개의 리워드텀으로 TRPO를 라이보로 정책 학습 후, anymal-c에 적용

관절 움직임

기본 관절(nominal) 구성을 벗어나 움직임이 이상해졌다

- Anymal-c 이전 시 nominal과 멀어진것으로 보아 이전이 제대로 안됐음을 알 수 있다
  - Anymal-c의 reward의 보상계수 변경이 필요했다

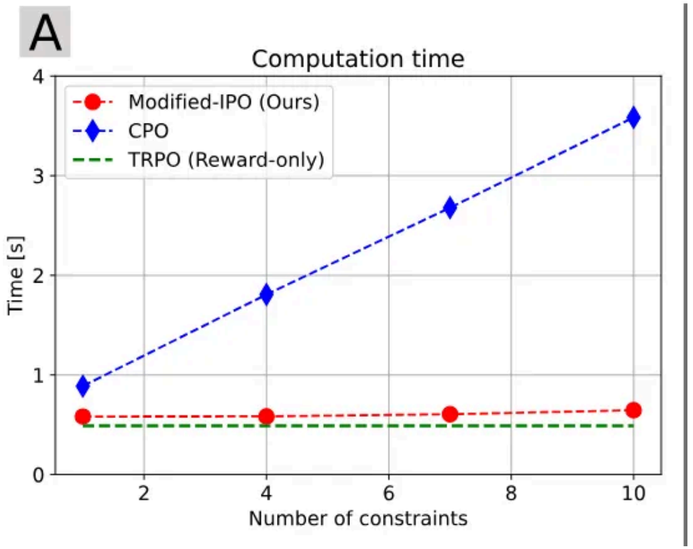
$$q_t^{des} = q^{nominal} + \sigma_a \cdot a_t,$$



RESULT

타 알고리즘과 비교

- **CPO:** 계산시간과 제약조건은 선형적이라 너무 오래 걸린다
- **Penalized Proximal Policy Optimization (P3O):**
  - 제약 조건 한계에 도달하며, 상대적으로 낮은 리워드를 얻는다
- **PPO:** 구현은 쉽다, 단 학습이 불안정했다



<계산 시간 비교표>

TABLE V
ALGORITHM PERFORMANCE COMPARISON

	Reward (r)	Torque	Smoothness	Traversability
Ours	**-0.242** ($\pm$ 0.007)	**24.769** ($\pm$ 5.811)	**0.093** ($\pm$ 0.002)	**0.811** ($\pm$ 0.055)
P3O [10], [44]	-0.271 ($\pm$ 0.012)	26.186 ($\pm$ 6.116)	0.101 ($\pm$ 0.002)	0.781 ($\pm$ 0.068)

P3O 대비 더 높은 리워드(우수한 성능)을 보인다

TABLE II
CONSTRAINT SATISFACTION RESULTS

		c_{jp}	c_{jv}	c_{jt}	c_{bc}	c_{com}	c_{gp}	c_{ov} [m/s]	c_{cv} [m/s]	c_{fc} [m]	c_{fh} [m]	c_{sym}
	Limit	0.025	0.025	0.025	0.025	0.025	0.25	0.35	-	-	-	0.1
Sec. VI-B	Raibo	0.015	0.004	0.001	0.002	0.009	0.16	0.32	0.15 [0.2]	-0.11 [-0.07]	0.09 [0.11]	0.07
	Hound	0.014	0.020	0.000	0.003	0.002	0.17	0.30	0.13 [0.2]	-0.09 [-0.07]	0.07 [0.11]	0.07
	Go1	0.012	0.007	0.000	0.008	0.001	0.16	0.33	0.10 [0.2]	-0.08 [-0.05]	0.06 [0.09]	0.07
	Mini-cheetah	0.015	0.000	0.001	0.002	0.001	0.13	0.32	0.10 [0.2]	-0.08 [-0.05]	0.06 [0.09]	0.06
	Anymal B	0.016	0.019	0.000	0.002	0.005	0.16	0.31	0.13 [0.2]	-0.09 [-0.07]	0.07 [0.11]	0.07
	Anymal C	0.019	0.021	0.002	0.004	0.005	0.16	0.30	0.11 [0.2]	-0.08 [-0.07]	0.06 [0.11]	0.07
	Atlas	0.011	0.019	0.004	0.008	0.008	0.12	0.29	0.35 [0.5]	-0.17 [-0.15]	0.12 [0.2]	0.08
Sec. VI-C	Torque (x2)	0.017	0.005	0.001	0.001	0.008	0.15	0.33	0.16 [0.2]	-0.13 [-0.07]	0.10 [0.11]	0.08
	Smoothness (x2)	0.013	0.005	0.001	0.001	0.009	0.14	0.32	0.15 [0.2]	-0.13 [-0.07]	0.10 [0.11]	0.07
Sec. VI-E	P3O [10], [44]	**0.025**	0.005	0.000	0.002	0.016	0.24	**0.35**	**0.2** [0.2]	-0.08 [-0.07]	0.09 [0.11]	**0.1**
	$t = 100$, $\alpha = 0.002$	0.015	0.004	0.002	0.003	0.009	0.17	0.32	0.15 [0.2]	-0.11 [-0.07]	0.09 [0.11]	0.07
	$t = 100$, $\alpha = 0.2$	0.015	0.004	0.002	0.002	0.011	0.18	**0.44**	0.17 [0.2]	-0.11 [-0.07]	0.09 [0.11]	0.07
	$t = 10$, $\alpha = 0.02$	0.004	0.001	0.000	0.000	0.002	0.11	0.24	0.07 [0.2]	-0.14 [-0.07]	0.07 [0.11]	0.05
	$t = 1000$, $\alpha = 0.02$	0.023	0.010	0.002	0.002	0.015	0.22	**0.45**	**0.2** [0.2]	-0.09 [-0.07]	0.09 [0.11]	0.09

*Results are after training with 1.6 billion time steps. Constraint thresholds are reported in the top row or inside the bracket. Bolded texts indicate violated

P3O의 경우 **제약 조건 한계에 도달**한다

RESULT

하이퍼 파라미터 비교

- t 값이 큰 경우: 직교 속도 불만족 → 너무 제약이 엄격해 제한적으로 업데이트가 이루어진다
- α 값이 클 경우: 직교 속도 불만족 → 탐색 공간이 너무 커져 제약 조건 불만족
- t 값이 작은 경우 : **제약을 큰 여유로 만족**한다, 따라서 낮은 리워드를 보인다
- α 값이 작은 경우 : TRPO 최적화 과정의 라인검색(line serach)동안 업데이트가 느려져서 **학습속도가 느리다**

$$\phi(\hat{J}^{\pi_\theta}_{C_i}) = \frac{\log(-\hat{J}^{\pi_\theta}_{C_i})}{t},$$

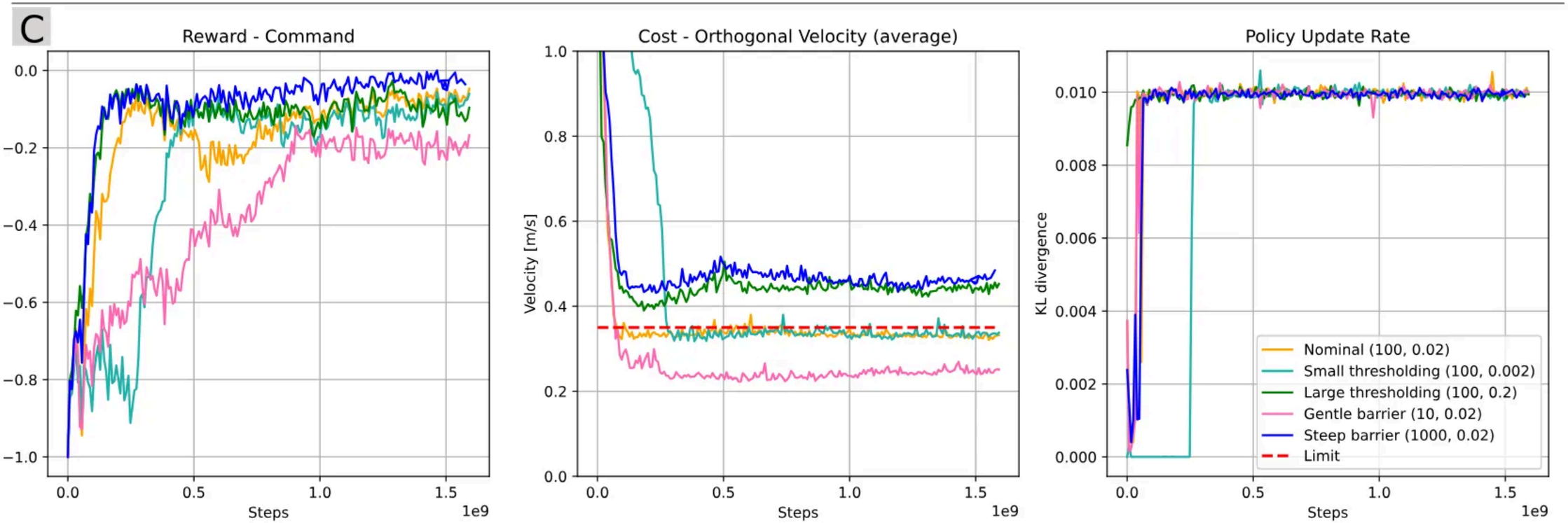
t : 제약 조건의 엄격함을 정하는 하이퍼파라미터

- Large *t*: 위반시 약한 패널티 부과
- Small *t*: 위반 시 큰 패널티 부과

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

- 적응적 제약 임계값*(Adaptive constraint thresholding)*

t = 100, α = 0.02 기준



END