Legged Locomotion:
로봇은 어떻게 걷는 법을 배우는가

RMA (Kumar et al., RSS 2021) · Perceptive Locomotion (Miki et al., Science Robotics 2022) 리뷰를 위한 배경

발표자 : 정윤

Why is Legged Locomotion Hard?

01

불연속적인 접촉

발이 땅에 닿고 떨어질 때마다 동역학이 바뀐다. 접촉 순서(gait)와 타이밍을 함께 결정해야 하는 하이브리드 시스템

02

떠 있는 몸체

몸통은 어디에도 고정되지 않은 floating base — 직접 구동할 수 없고 발의 지면 반력으로만 간접 제어 (underactuated)

03

예측 어려운 환경

마찰, 지면 변형, 하중, 모터 상태가 계속 변한다. 정확한 모델을 세우기도, 실시간으로 알아내기도 어렵다

그럼에도 다리가 필요한 이유: 계단, 돌밭, 풀숲처럼 바퀴가 갈 수 없는 지형을 이동할 수 있다.

Model-based vs Learning-based Control

모델 기반 (Model-based)

ZMP, MPC, 궤적 최적화 · MIT Cheetah, Boston Dynamics

방식 단순화된 동역학 모델 + 수작업 설계 컨트롤러

장점 해석 가능, 안정성 분석이 쉬움

한계 모델 가정이 깨지는 지형(모래, 미끄럼)에 취약, 튜닝에 많은 전문 인력

학습 기반 (Deep RL)

시뮬레이션에서 시행착오로 정책 신경망 학습

방식 보상 함수만 설계, 행동은 PPO 등으로 학습

장점 복잡한 접촉·지형에도 강인한 행동이 자연스럽게 등장

한계 실제 로봇에서 수백만 번 넘어질 수 없음 → 시뮬레이터 의존

→ **최근 흐름:** 시뮬레이션에서 RL로 학습하고 실제 로봇에 그대로 옮기는 **Sim-to-Real**

RL Locomotion Policy

관측 o□

관절 위치·속도, IMU(roll·pitch), 이전 행동

정책 π

MLP 신경망
~50–100 Hz

행동 a□

목표 관절 위치

PD 제어기

$\tau = Kp(q^* - q) - Kd \cdot \dot{q}$

로봇 / 시뮬레이터

토크 → 다음 상태

보상 r□ = 잘 걷기 − 비용

- 목표 속도 추종 (앞으로 나아가기)
- 에너지·토크·급격한 동작 페널티 (자연스러운 걸음)
- 넘어짐·미끄러짐·몸체 흔들림 억제

학습은 시뮬레이터에서

- PPO 등 on-policy RL, 수억~수십억 스텝
- RaiSim, Isaac Gym, MuJoCo 등 물리 엔진
- 카메라 없이 고유수용 감각(proprioception)만으로도 험지 보행 가능

The Sim-to-Real Gap

도메인 랜덤화

마찰·질량·모터 세기를 무작위로 바꿔가며 학습

한계 모든 환경에 대비하느라 보수적인 정책

Tan et al. 2018

시스템 식별 (SysID)

실제 로봇의 물리 파라미터를 측정·추정해 시뮬레이터 보정

한계 파라미터를 정확히 알기 어렵고 환경이 바뀌면 다시 해야 함

Hwangbo et al. 2019

실세계 적응

실제 로봇에서 데이터를 모아 정책을 미세조정

한계 수 분 이상 데이터 필요, 로봇 손상 위험

Peng et al. 2020

RMA의 질문: 실제 데이터 없이, 배포 중에 1초 안에 스스로 환경에 적응할 수는 없을까?

RMA: Rapid Motor Adaptation for Legged Robots

Ashish Kumar, Zipeng Fu, Deepak Pathak, Jitendra Malik

발표자 : 정윤

Introduction

Problem: The Real World ≠ Simulator

지형의 다양성

모래, 진흙, 풀밭, 매트리스, 기름 묻은 바닥 — 시뮬레이터로 정확히 모델링하기 어려운 변형·접촉 역학

로봇 자체의 변화

하중(payload) 변화, 모터 마모, 배터리 상태 등으로 로봇 동역학이 계속 바뀜

적응 속도

사람처럼 수 초가 아니라 **1초 미만**에 적응해야 넘어지지 않음.
실제 데이터 수집은 로봇을 손상시킬 위험

기존 접근의 한계: **도메인 랜덤화**는 보수적인 정책을 만들고, **real world finetuning** 은 수 분 이상의 데이터가 필요하다.

RMA in the Wild



Key Idea: Base Policy + Adaptation Module

Base policy π

환경 정보를 압축한 외재 변수 z□를 입력받아 관절 목표를 출력하는 보행 정책

Adaptation module ϕ

최근 0.5초의 상태·행동 이력만으로 z□를 실시간 추정 — 암묵적 시스템 식별기

Algorithm 1: Rapid Motor AdaptationTraining

Phase 1 Randomly initialize the base policy π;
Randomly initialize the environmental factor encoder μ; Empty replay buffer D_1;
for $0 \leq \text{itr} \leq N^1_{\text{itr}}$ **do**
  for $0 \leq i \leq N_{\text{env}}$ **do**
    $x_0, e_0 \leftarrow \text{envs}[i].\text{reset}()$;
    for $0 \leq t \leq T$ **do**
      $z_t \leftarrow \mu(e_t)$;
      $a_t \leftarrow \pi(x_t, a_{t-1}, z_t)$;
      $x_{t+1}, e_{t+1}, r_t \leftarrow \text{envs}[i].\text{step}(a_t)$;
      Store $((x_t, e_t), a_t, r_t, (x_{t+1}, e_{t+1}))$ in D_1;
    end
  end
  Update π and μ using PPO [48];
  Empty D_1;
end

Phase 2 Randomly initialize the adaptation module ϕ parameterized by θ_ϕ; Empty mini-batch D_2;
for $0 \leq \text{itr} \leq N^2_{\text{itr}}$ **do**
  for $0 \leq i \leq N_{\text{env}}$ **do**
    $x_0, e_0 \leftarrow \text{envs}[i].\text{reset}()$;
    for $0 \leq t \leq T$ **do**
      $\hat{\mathbf{z}}_\mathbf{t} \leftarrow \phi(x_{t-k:k}, a_{t-k-1:k-1})$;
      $z_t \leftarrow \mu(e_t)$;
      $a_t \leftarrow \pi(x_t, a_{t-1}, \hat{\mathbf{z}}_\mathbf{t})$;
      $x_{t+1}, e_{t+1}, _ \leftarrow \text{envs}[i].\text{step}(a_t)$;
      Store $(\hat{z}_t, z_t)$ in D_2;
    end
  end
  $\theta_\phi \leftarrow \theta_\phi - \lambda_{\theta_\phi} \nabla_{\theta_\phi} \frac{1}{TN_{\text{env}}} \sum \|\hat{z}_t - z_t\|^2$;
  Empty D_2;
end

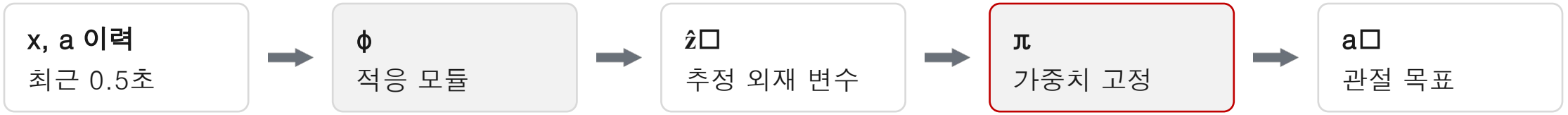
Method: Rapid Motor Adaptation

Two Phases, One Policy

Phase 1 · 시뮬레이션 RL (환경 정보 e_t 사용)

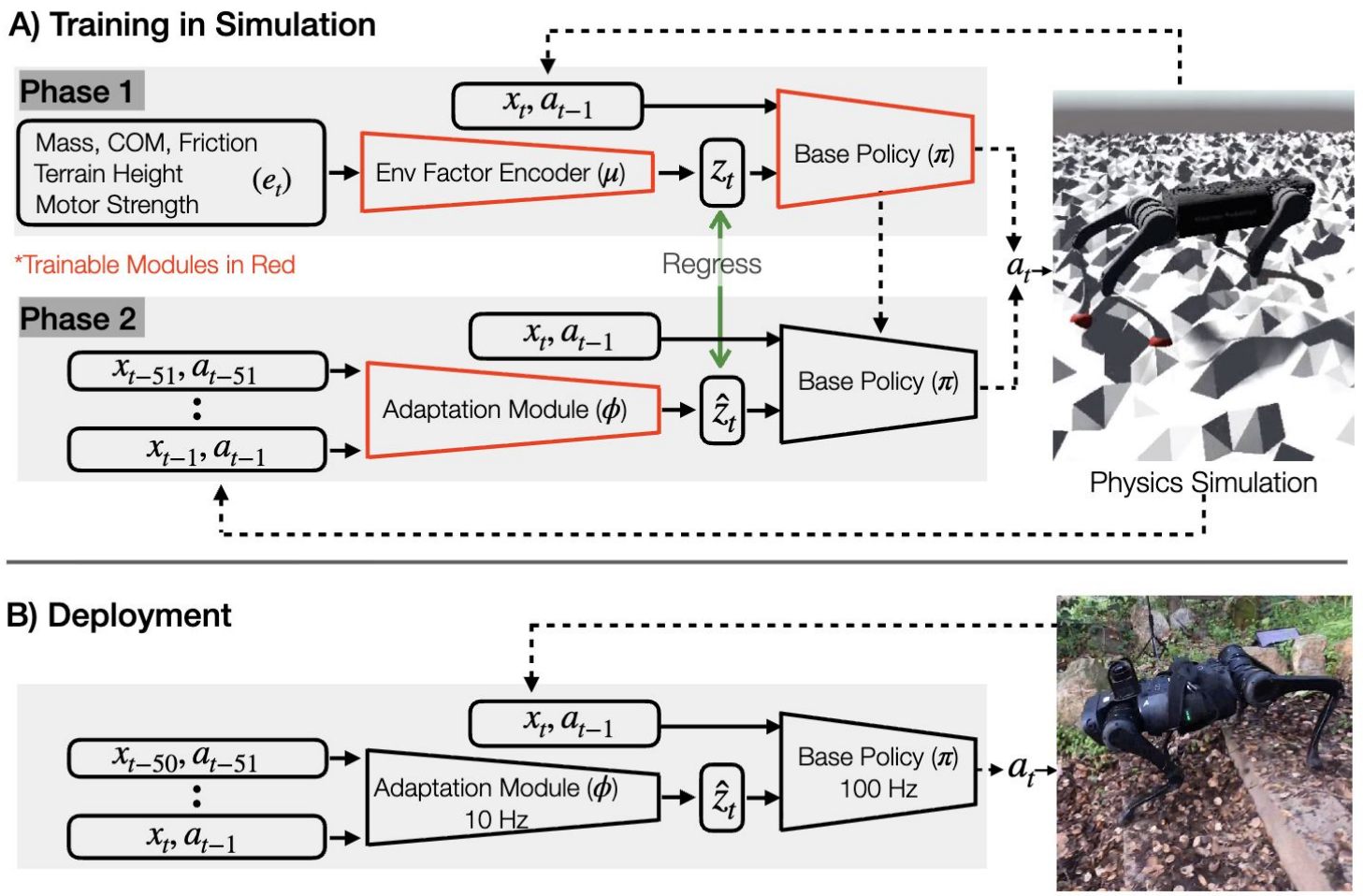


Phase 2 · 지도학습 & 배포 (로봇 고유 감각만 사용)



ϕ는 $\|\hat{z}_t - z_t\|^2$ 를 최소화하도록 학습 → 실제 로봇에서는 e_t 없이 $\hat{z}_t$로 대체

Overview: Two-Phase Training



Inputs & Outputs

30

상태 x□

관절 위치 12, 관절 속도 12, roll·pitch, 발 접촉 4

12

행동 a□

목표 관절 위치 → PD 제어기로 토크 변환

17

환경 e□

질량·무게중심, 모터 세기 12, 마찰, 지형 높이 (시뮬레이션에서만 알 수 있음)

8

외재 변수 z□

e□를 압축한 저차원 잠재 벡터 — 정책이 실제로 필요로 하는 정보만

로봇: **Unitree A1** (약 12kg, 다리당 모터 3개, 총 12 관절) · 카메라 없이 **고유수용 감각(proprioception)** 만 사용

Formulation

Phase 1 · 특권 정보 e□로 강화학습

$$z_t = \mu(e_t), \quad e_t \in \mathbb{R}^{17}, \ z_t \in \mathbb{R}^{8}$$

$$a_t = \pi(x_t, a_{t-1}, z_t), \quad x_t \in \mathbb{R}^{30}, \ a_t \in \mathbb{R}^{12}$$

$$J(\pi, \mu) = \mathbb{E}_{\tau\sim\pi}\left[\Sigma_t \gamma^t r_t \right] \rightarrow \max$$

π와 μ를 PPO로 end-to-end 공동 학습 — z_t는 "정책에 필요한 만큼만" 압축된 환경 정보

Phase 2 · 이력으로 z□를 회귀

$$\hat{z}_t = \phi(x_{t-k:t-1}, a_{t-k:t-1}), \quad k = 50$$

$$\min_\phi \ \mathbb{E}\left[\|\hat{z}_t - z_t\|^2 \right], \quad z_t = \mu(e_t)$$

$$a_t = \pi(x_t, a_{t-1}, \hat{z}_t)$$

π, μ는 고정 — RL 문제를 지도학습(회귀) 문제로 바꿈. k = 50 스텝 = 0.5초 (100 Hz)

행동 → 토크 (PD 제어)

$$\tau = K_p\,(\hat{q} - q) + K_d\,(\hat{\dot{q}} - \dot{q}) \quad (\hat{q} = a_t, \ \hat{\dot{q}} = 0)$$

네트워크 구조

π: 3층 MLP (hidden 128) · μ: 3층 MLP (256, 128)
ϕ: MLP 임베딩 (32) + 3층 1D-CNN + Linear

Phase 1 — Reward Design

항	수식	가중치
전진 (Forward)	$\min(v_x^t, 0.35)$	20
횡이동·회전 (Lateral Movement and Rotation)	$-\|v_y^t\|^2 - \|\omega_{yaw}^t\|^2$	21
일 (Work)	$-\|\tau^\top (q^t - q^{t-1})\|$	0.002
지면 충격 (Ground Impact)	$-\|f^t - f^{t-1}\|^2$	0.02
부드러움 (Smoothness)	$-\|\tau^t - \tau^{t-1}\|^2$	0.001
행동 크기 (Action Magnitude)	$-\|a^t\|^2$	0.07
관절 속도 (Joint Speed)	$-\|\dot{q}^t\|^2$	0.002
자세(Orientation)	$-\|\Theta_{roll,pitch}^t\|^2$	1.5
수직 가속 (Z Acceleration)	$-\|v_z^t\|^2$	2.0
발 미끄럼 (Foot Slip)	$-\|\mathrm{diag}(g^t) \cdot v_f^t\|^2$	0.8

기호

v 몸체 선속도, ω 각속도, Θ 자세, q 관절각, τ 토크, f 발 지면반력, v_f 발 속도, $g \in \{0,1\}^4$ 발 접촉

커리큘럼: 페널티를 점점 키운다

$$k_{j+1} = k_j^{0.997}, \quad k_0 = 0.03$$

페널티 항에 k_j를 곱함 (j: PPO iteration). $\ln k_j = 0.997^j \cdot \ln k_0 \to 0$ 이므로 $k_j \to 1$

0.03	0.19	0.46	0.84	0.99
j=0	j=250	j=500	j=1000	j=2000

처음부터 페널티가 크면 "가만히 서 있기"라는 국소해에 빠짐

Phase 1 — Optimization with PPO

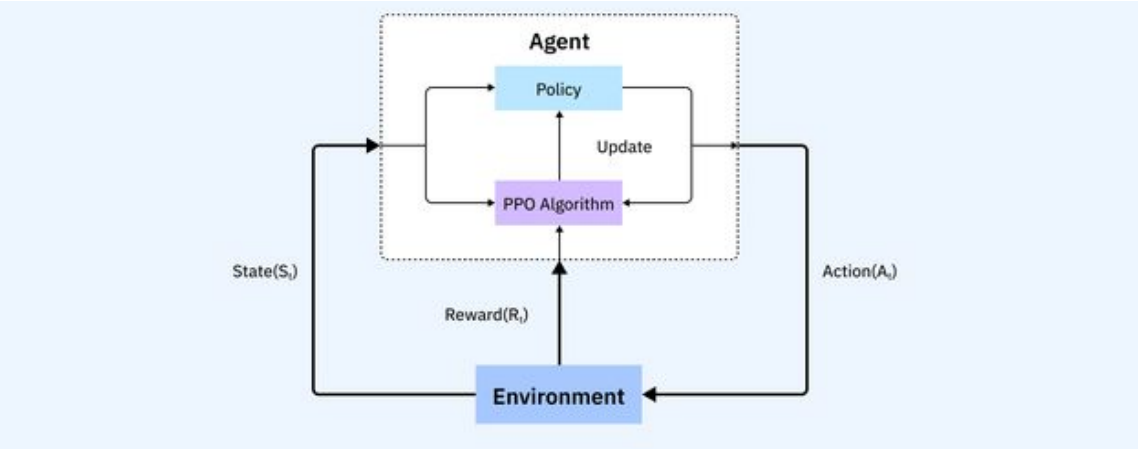
목적함수 (θ = {π, μ} 의 파라미터)

$$\rho_t(\theta) = \pi_\theta(a_t \mid s_t) \,/\, \pi_{old}(a_t \mid s_t)$$

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\, \min\left(\rho_t \hat{A}_t,\ \mathrm{clip}(\rho_t, 1-\varepsilon, 1+\varepsilon)\, \hat{A}_t \right) \right]$$

$$\hat{A}_t = \Sigma_l\, (\gamma\lambda)^l\, \delta_{t+l}, \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

- 관측 $s_t = (x_t, a_{t-1}, e_t)$: e_t가 μ를 거쳐 π로 들어가므로, 정책 gradient가 z_t를 통해 μ까지 역전파
- 정책 분포 $a_t \sim \mathcal{N}(\pi(\cdot), \sigma^2)$, σ ≥ 0.2로 제한 · 엔트로피 보너스 미사용
- 에피소드 종료: 몸체 높이 < 0.28 m, |roll| > 0.4 rad, |pitch| > 0.2 rad



하이퍼파라미터

반복 횟수	15,000 iter
배치	80,000 (미니배치 4 × 20,000)
최적화	Adam, lr 5×10^{-4}
할인율 γ	0.998
GAE λ	0.95
clip 범위	$\rho \in [0.8, 1.2]$ ($\varepsilon = 0.2$)
총 샘플	약 12억 스텝
비용	GPU 1개, 약 24시간

γ = 0.998 → 유효 시야 $1/(1-\gamma) \approx 500$ 스텝 (5초)

Phase 2 — Adaptation Module

$$\hat{z}_t = \phi(h_t), \quad h_t = (x_{t-50:t-1}, a_{t-50:t-1}) \qquad \mathcal{L}(\phi) = \mathbb{E}_{\mathcal{D}} \left[\| \phi(h_t) - \mu(e_t) \|^2 \right]$$

입력 이력	MLP 임베딩	Conv1D	Conv1D	Conv1D	Flatten	Linear
50 × 42	50 × 32	11 × 32	7 × 32	3 × 32	96	$\hat{z}_{\square} \in \mathbb{R}^8$
(x 30 + a 12)	스텝마다 2층	k 8, s 4	k 5, s 1	k 5, s 1		

시퀀스 길이: $L_{out} = \lfloor (L_{in} - k)/s \rfloor + 1 \rightarrow 50 \rightarrow 11 \rightarrow 7 \rightarrow 3$ (padding 없음 가정)

학습 루프 (on-policy)

1. ϕ 무작위 초기화, $\pi \cdot \mu$ 고정
2. $a_t = \pi(x_t, a_{t-1}, \hat{z}_t)$로 롤아웃 — ϕ의 추정치로 로봇을 움직임
3. 정답 $z_t = \mu(e_t)$와 짝지어 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(h_t, z_t)\}$
4. $\phi \leftarrow \phi - \eta \nabla_{\phi} \mathcal{L}(\phi)$, 2로 돌아가 반복

왜 on-policy인가

z_t(정답)로 모은 궤적만 쓰면 학습 분포 ≠ 배포 분포$\hat{z}_t$의 오차로 벗어난 상태 h_t에서도 회귀하도록 데이터셋 $\mathcal{D}$를 ϕ 자신이 만든다.

1,000 iter · 배치 80,000 · lr 5×10^{-4} · 약 8천만 스텝 · GPU 1개 3시간

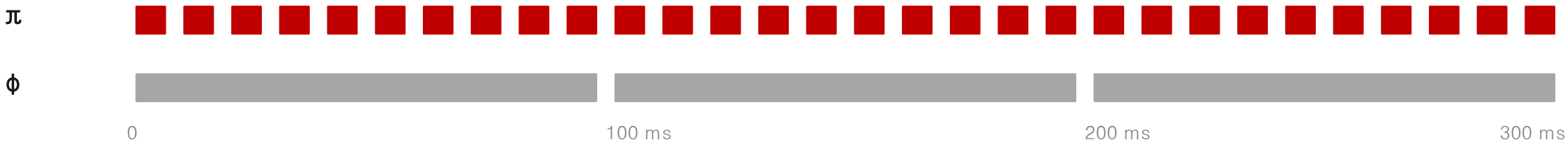
Deployment — Asynchronous Execution

100 Hz — 기본 정책 π

$a_t = \pi(x_t, a_{t-1}, \hat{z}_{t'})$, $t' = \phi$의 최근 출력 시점

10 Hz — 적응 모듈 ϕ

100 ms마다 $\hat{z} \leftarrow \phi(h_t)$ 갱신 — 동기화 없이 독립 실행



가정: 환경 e_t는 로봇 동작보다 천천히 변한다 → $\hat{z}$를 10스텝 재사용해도 충분. 출력 a_t는 PD 제어 $\tau = K_p(a_t - q) - K_d \dot{q}$로 토크 변환 → 저가 로봇 온보드 CPU만으로 실시간 적응

Results

Experimental Setup

Parameters	**Training Range**	**Testing Range**
Friction	[0.05, 4.5]	[0.04, 6.0]
K_p	[50, 60]	[45, 65]
K_d	[0.4, 0.8]	[0.3, 0.9]
Payload (Kg)	[0, 6]	[0, 7]
Center of Mass (cm)	[-0.15, 0.15]	[-0.18, 0.18]
Motor Strength	[0.90, 1.10]	[0.88, 1.22]
Re-sample Probability	0.004	0.01

TABLE I: Ranges of the environmental parameters.

Quantitative: Simulation Results

방법	성공률 (%)	TTF	보상	적응 샘플
Expert (e□ 사용, 상한)	76.2	0.86	5.23	0
RMA (제안)	73.5	0.85	5.22	0
Robust (도메인 랜덤화)	62.4	0.80	4.62	0
SysID (e□ 직접 추정)	56.5	0.74	4.82	0
RMA w/o 적응 모듈	52.1	0.75	4.72	0
AWR (실세계 적응)	41.7	0.65	4.17	40k

잠재 벡터 추정(RMA)이 물리 파라미터 직접 추정(SysID)보다 우수 — 정책에 필요한 정보만 추정하면 된다

	Success (%)	TTF	Reward	Distance (m)	Samples	Torque	Smoothness	Ground Impact
Robust [52, 40]	62.4	0.80	4.62	1.13	0	527.59	122.50	4.20
SysID [57]	56.5	0.74	4.82	1.17	0	565.85	149.75	4.03
AWR [41]	41.7	0.65	4.17	0.95	40k	599.71	162.60	4.02
RMA w/o Adapt	52.1	0.75	4.72	1.15	0	524.18	106.25	4.55
RMA	73.5	0.85	5.22	1.34	0	500.00	92.85	4.27
Expert	76.2	0.86	5.23	1.35	0	485.07	85.56	3.90

Quantitative: Real-world Results

실내 과제 (성공률 %)	RMA	w/o 적응	A1 기본
울퉁불퉁한 폼	100	20	0
메모리폼 매트리스	100	0	60
경사로 6°	100	40	80
계단 오르기 8cm	100	0	80
계단 내려가기 15cm	80	0	20

12kg

자기 체중만큼의 하중을 지고도 보행 (기존 컨트롤러는 8kg부터 저하)

야외: 모래, 진흙, 풀숲, 흙더미, 등산로, 기름 바른 바닥 — 모두 **같은 정책**, 실세계 미세조정 없음

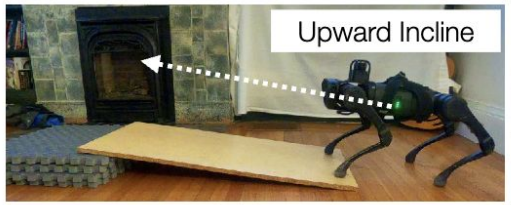
Real-world: Indoor Tasks & Payload



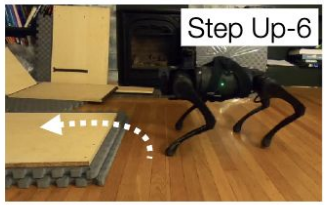
	Success	TTF	Distance
RMA	80	0.9	0.9
RMA w/o Adapt	0	0.1	0.1
A1	20	0.36	0.36



	Success
RMA	100
RMA w/o Adapt	0
A1	60



	Success	TTF	Distance
RMA	100	1	1
RMA w/o Adapt	20	1	0.6
A1	100	1	1



	Success
RMA	100
RMA w/o Adapt	40
A1	80

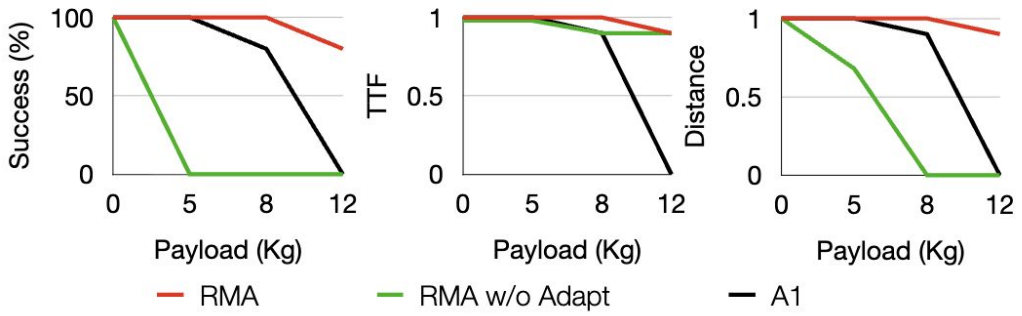


	Success	TTF	Distance
RMA	100	1	1
RMA w/o Adapt	0	1	0.25
A1	100	1	1



	Success
RMA	60
RMA w/o Adapt	0
A1	20





Qualitative: A1 Controller vs RMA

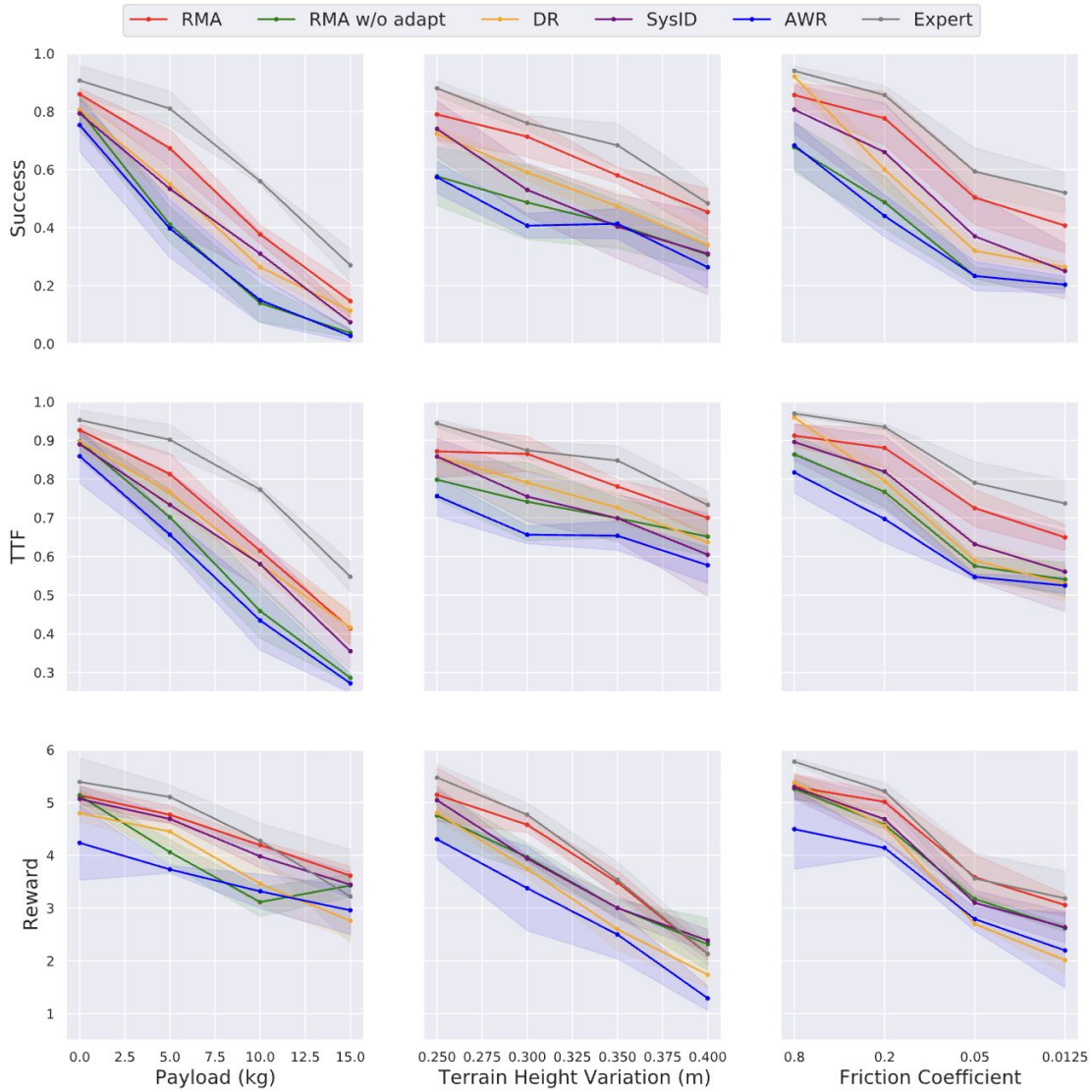
A1 controller



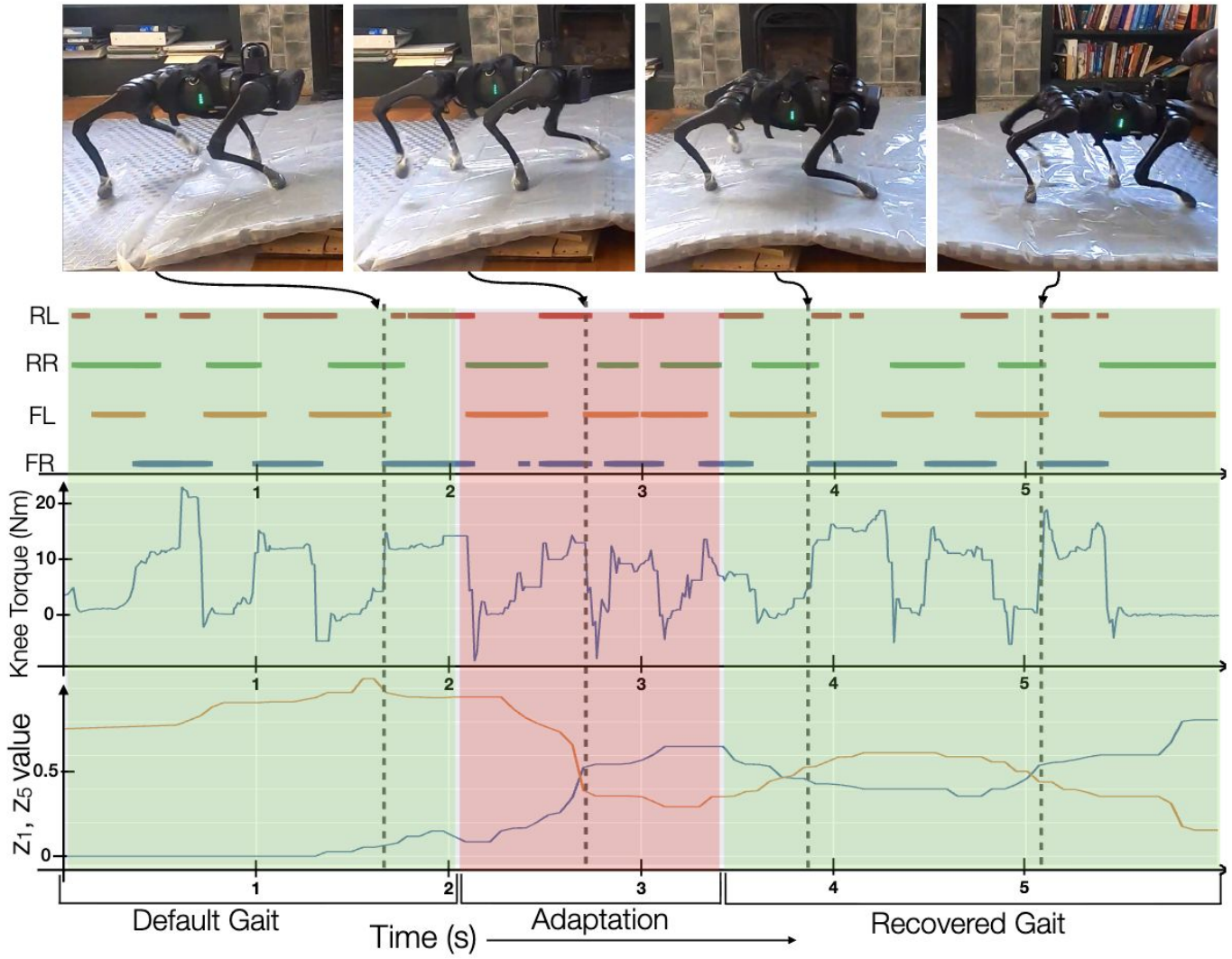
RMA



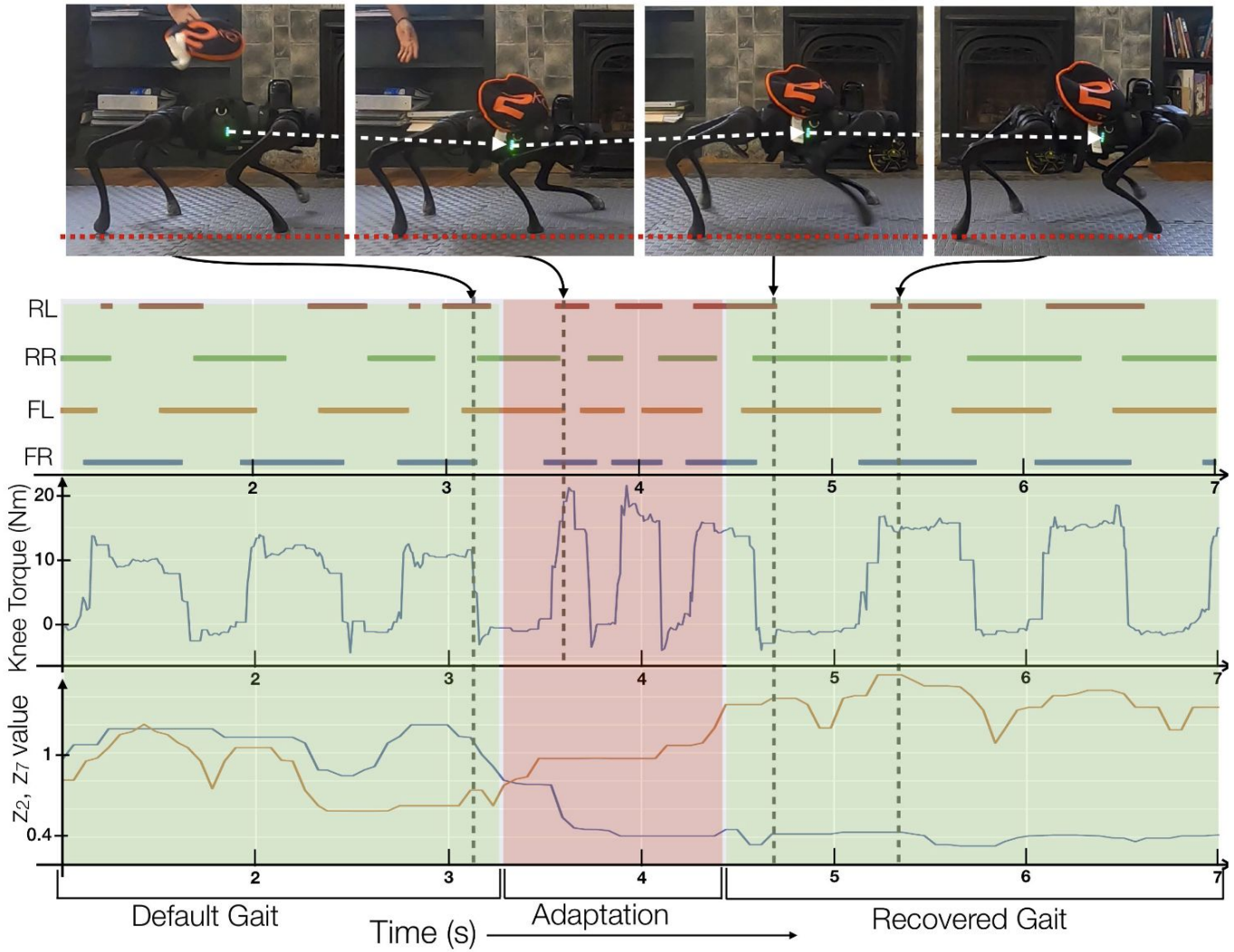
Performance vs Environment Difficulty



Analysis — Slippery Surface



Analysis — Sudden Payload



Analysis: $\hat{z}$ Encodes the Environment

미끄러운 바닥 진입

기름 묻은 구간에 들어서는 순간 $\hat{z}_{\square}$의 특정 성분(z_1 , z_5)이 급변하고, 구간을 지나는 동안 그 값을 유지

보행 주기는 비슷하게 회복되지만 무릎 토크가 커지며 대응

갑작스러운 5kg 하중

하중이 올라가면 몸체가 내려앉았다가, 다른 성분(z_2 , z_7)이 반응하며 수 초 안에 보행 회복

하중이 있는 동안 $\hat{z}_{\square}$가 계속 그 상태를 추적

→ ϕ는 힘·마찰 센서 없이 **고유 감각만으로 동작하는 암묵적 시스템 식별기**

Conclusion

Conclusion

기여

- 적응을 "특권 정보 추정"이라는 지도학습 문제로 전환
- 실세계 데이터·보정·미세조정 없이 sim-to-real 달성
- 저가 로봇에서 비동기 실행으로 실시간 동작
- 도메인 랜덤화, SysID, 제조사 컨트롤러보다 우수

한계와 향후 과제

- 시각 정보가 없어 장애물을 미리 피할 수 없음
- 계단에서의 갑작스러운 낙하 등 큰 교란에는 실패
- → 비전 결합, 장기·고속 보행

감사합니다

- https://youtu.be/nBy1piJrq1A?si=DzJ76ZXW1QPMY0T5