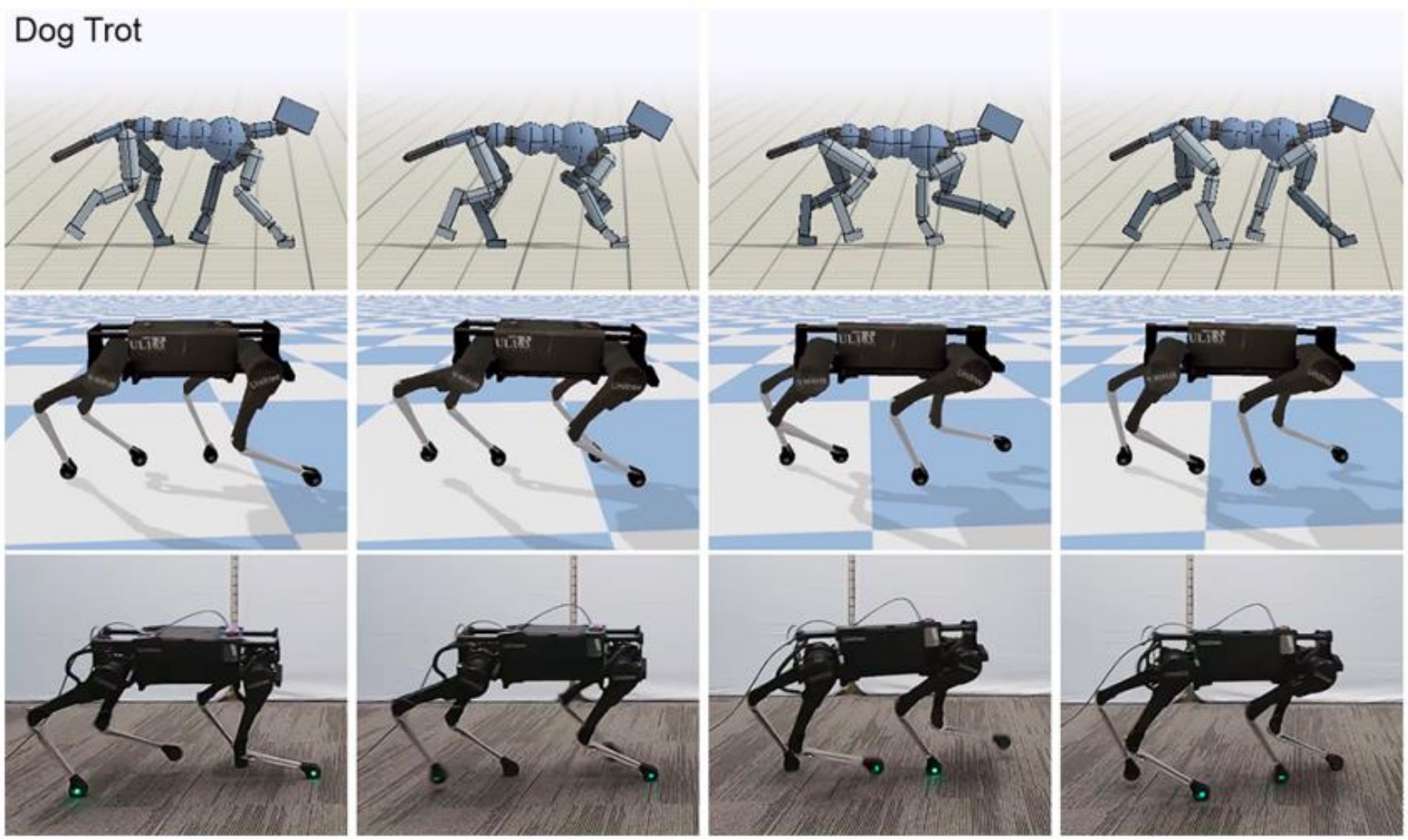
Learning Agile Robotic Locomotion Skills by Imitating Animals



MICRO-507 – Legged Robots

Leen Daher
Marc Mouawad
Matteo Mesa Gonzalez

École
polytechnique
fédérale
de Lausanne

Introduction

Presents an imitation learning framework that allows legged robots to learn agile and diverse locomotion skills by imitating real-world animal behaviours.

Motivation:

Manual control strategies:

- Require dynamic modelling and robotics expertise
- Require deep understanding of the movement
- Have limited flexibility and efficiency in replicating complex movement

Proposed Solution

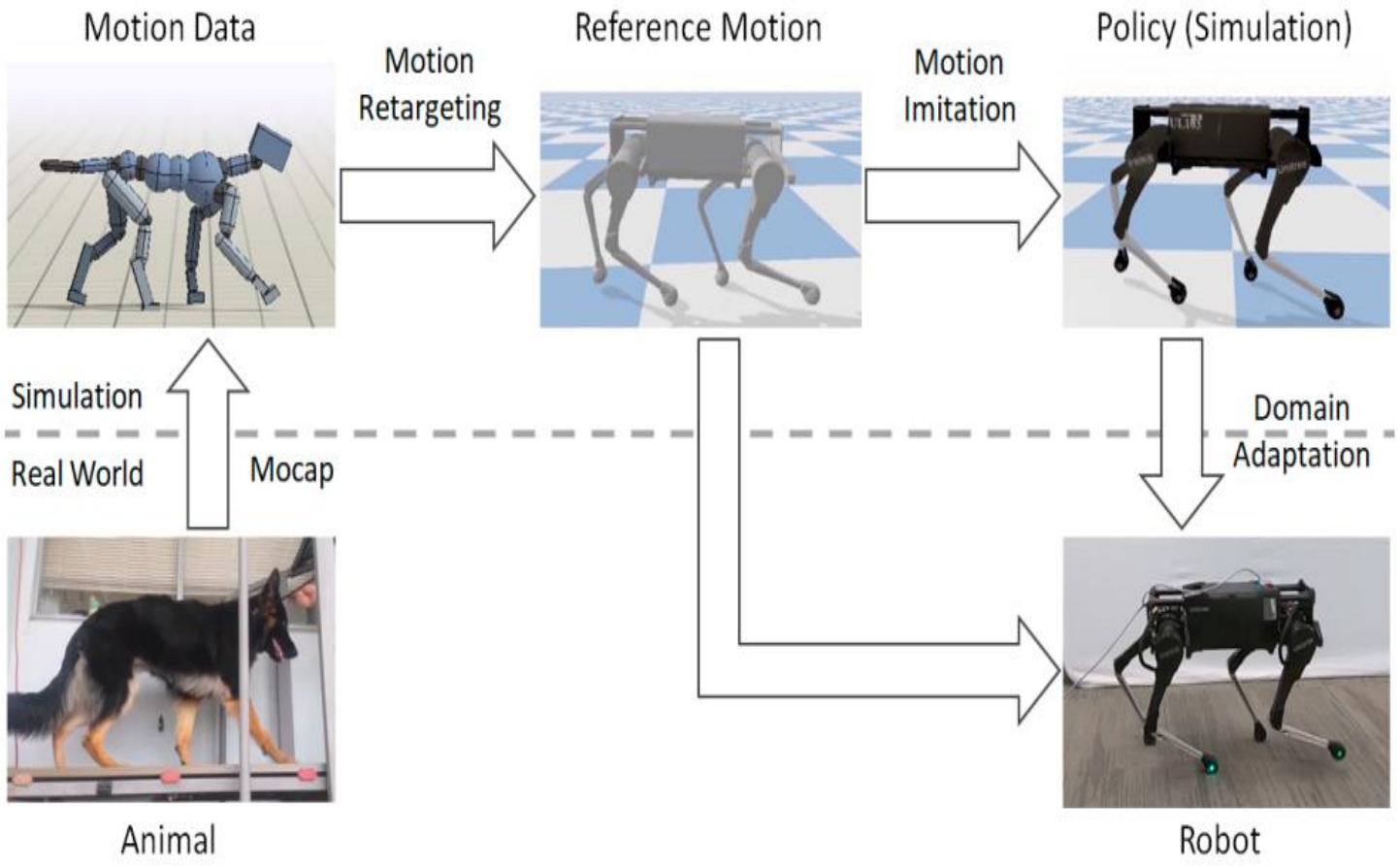
Imitation learning: Leverages animals' motion data to teach robots through simulation

Domain Adaptation: Bridges the gap between simulation and real-world deployment

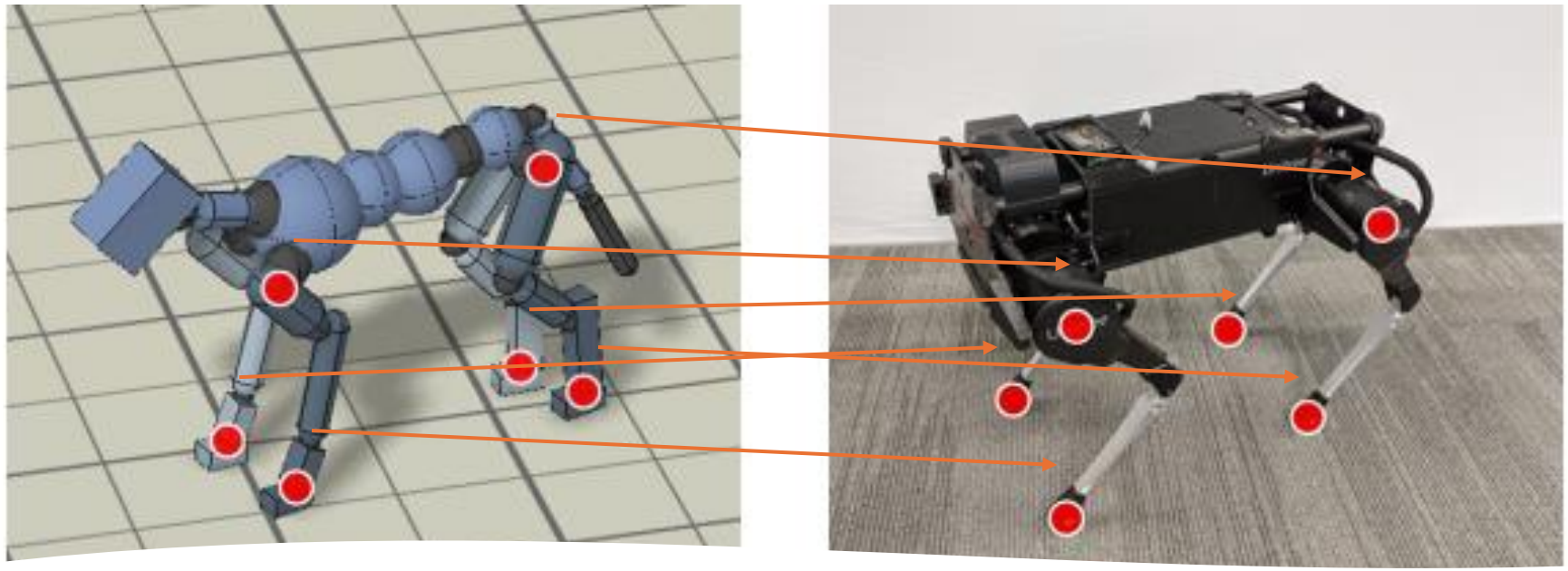
- **Robot used**: Quadruped robot (Laikago)
- **Control type**: Reinforcement learning

Design Method Overview

- Motion Retargeting
- Motion Imitation
- Domain Adaptation



Motion Retargeting



- Differences between the subject's and the robot's morphologies.
- Define a set of keypoints on the hips and the feet for both.
- Define a direct mapping between the keypoints.
- Then choose the joints of the robots that minimize the objective:
- $\arg\min \sum_i \sum_i \left|\left|\hat{x}_i(t) - x_i(q)\right|\right| + (\bar{q} - q)^T W (\bar{q} - q)$

Motion Imitation as an RL problem (1)

- Learn a control policy π that maximizes its expected return for some task:

$$J(\pi) = E_{\tau \sim p(\tau|\pi)}[\sum_{t=0}^{T-1} \gamma^t r_t]$$

- To do so: At every t:
  - Sample an action $a_t \sim \pi(a_t|s_t)$ which results in state s_{t+1}.
  - Compute reward $r_t = r(s_t, a_t, s_{t+1})$.
  - At every time-step this results in a trajectory: $\tau = \{(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots\}$
  - Note:

$$p(\tau|\pi) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\, \pi(a_t|s_t)$$

Motion Imitation as an RL problem (2)

- $s_t = (q_{t-2:t}, a_{t-3:t-1})$: Last three joint positions and actions.
- q: IMU measurements, (row, pitch, yaw).
- We introduce to the policy desired goal poses:
- $g_t = (\hat{q}_{t+1}, \hat{q}_{t+2}, \hat{q}_{t+10}, \hat{q}_{t+30})$: Four desired poses spanning 1s.
- With this we now sample actions: $a_t = \pi(a_t | s_t, g_t)$.
- Reward function is defined as:
- $r_t = 0.5r_t^p + 0.05r_t^v + 0.2r_t^e + 0.15r_t^{rp} + 0.1r_t^{rv}$
- The different reward terms are defined as:

$$r_t^k = \exp\left(-a \sum_i ||\hat{k}_t^i - k_t^i| \Big|^2\right), a > 0.$$

Domain Adaptation and Domain Randomization

- To transfer to real world, the main challenge is that the dynamics are different from the simulator's dynamics.
- μ: $vector\ of\ dynamics\ parameters$.
- Randomly generate multiple μ, and train policies over different dynamics → Domain Randomization.
- $z \sim E(z|\mu)$: Latent space features of the dynamics, trained using a stochastic encoder → Domain Adaptation.
- At every time step, a search is performed where an optimal z* is found.
- This representation is fed to the policy network.

Potential Issue and fix:

- The issue at hand is the model might overfit by assuming z is too accurate of a representation of the system dynamics.
- Due to unmodeled effects, no vector would be a perfect representation.
- Fix: Add a regularization term to the amount of mutual information the model can access to z and μ. We get the following optimization problem:

$$argmax\ E_{\mu\sim p(\mu)}E_{z\sim E(z|\mu)}E_{\tau\sim p(\tau|\pi,\mu,z)}\left[\sum_{t}^{T-1}\gamma^{t}r_{t}\right]-\beta E_{\mu\sim p(\mu)}[D_{KL}[E(z|\mu)||\rho(z)]]$$

Regularizing the mutual information

- The additional term acts as a regularization over the mutual information between the dynamic parameters and their latent representation.
- The D_KL divergence acts as a distance measure between E and a variational prior p(z).
- β: Acts as a tradeoff between robustness and adaptability. For higher values, the optimizer would not look into as much information between mu and z, therefore we get closer to domain randomization. For lower values, we are in a domain adaptation setup. Hence, for moderate values of beta, we get the best of both worlds.

Overview of Policy Types

- **Policy Types Compared**
  1. **No Rand** (Non-Adaptive, No Randomness)
     Trained without dynamic variation; performs consistently in a fixed environment.

  2. **Robust** (Robust with Randomness)
     Trained with randomized dynamics for better resistance to variations but lacks adaptability in new conditions.

  3. **Adaptive before** (Adaptive Policy)
     Adjusts actions to the real-world environment, enhancing stability and performance

  4. **Adaptive After** (Adaptive with information Bottleneck)
     Limits access to precise dynamic information with optimal β, improving robustness while allowing flexible adaptation.

Experimental Methodology

- **Two-step Procedure**

1. **Simulation**

   Trained in PyBullet with mocap data from dogs and artist-created animation.

2. **Transfer to Real World**

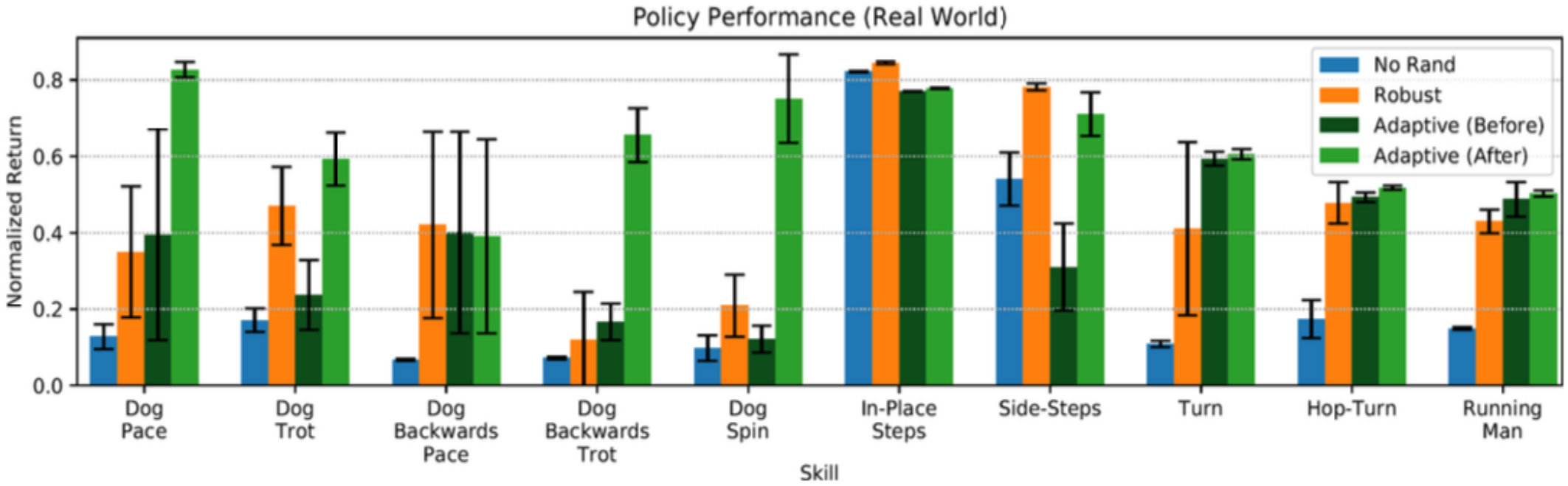
   Each Policies tested on the physical robot

   - **Uniform Evaluation**

     Normalized Return: Policies scored on a normalized scale from 0 to 1 for objective comparison.

Real-World Adaptation

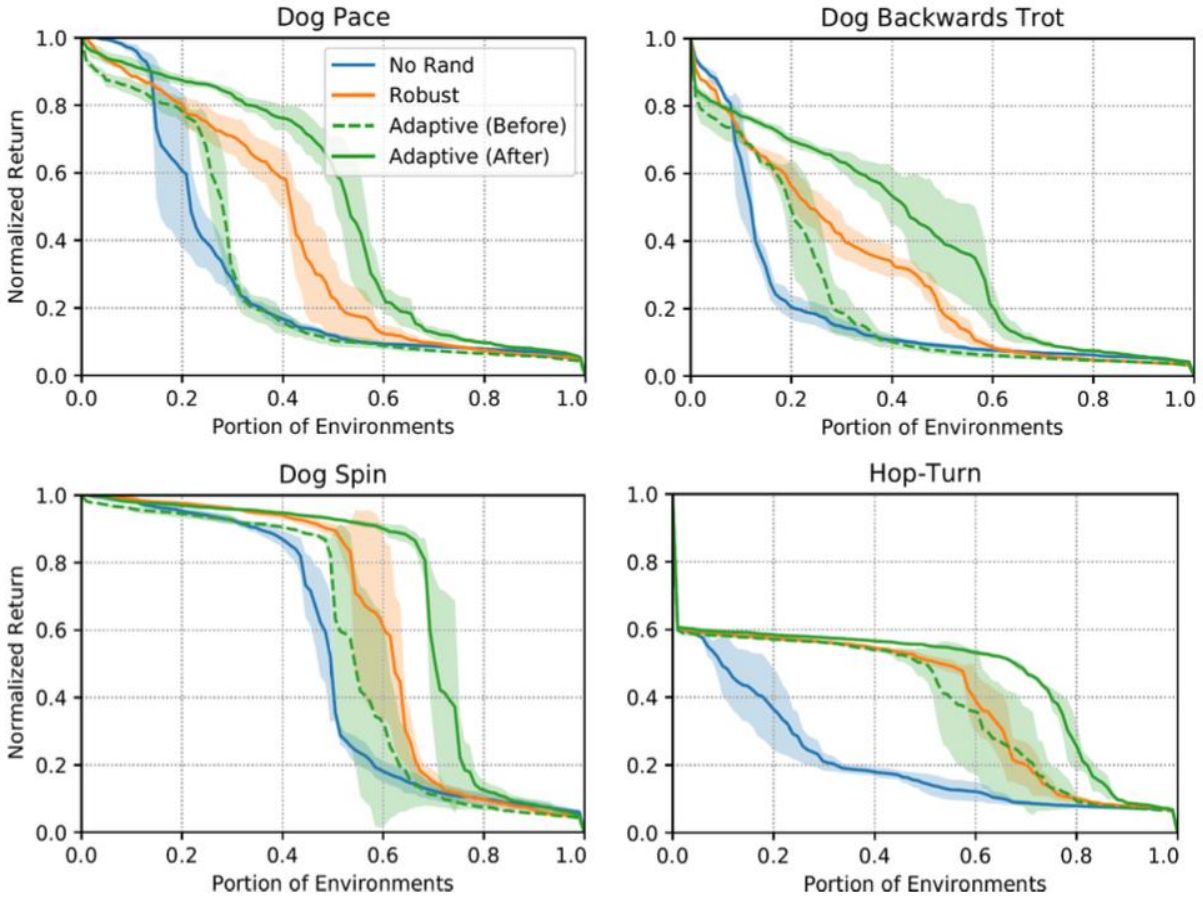
- **Experimental Results**
  - **Learned Locomotion Skills**



Caption: A showcase of various locomotion skills acquired by Laikago, including pacing, trotting, and backward movements. The figure compares performance across policies, demonstrating the advanced abilities of adaptive policies.

Adaptability and Performance in Diverse Environments

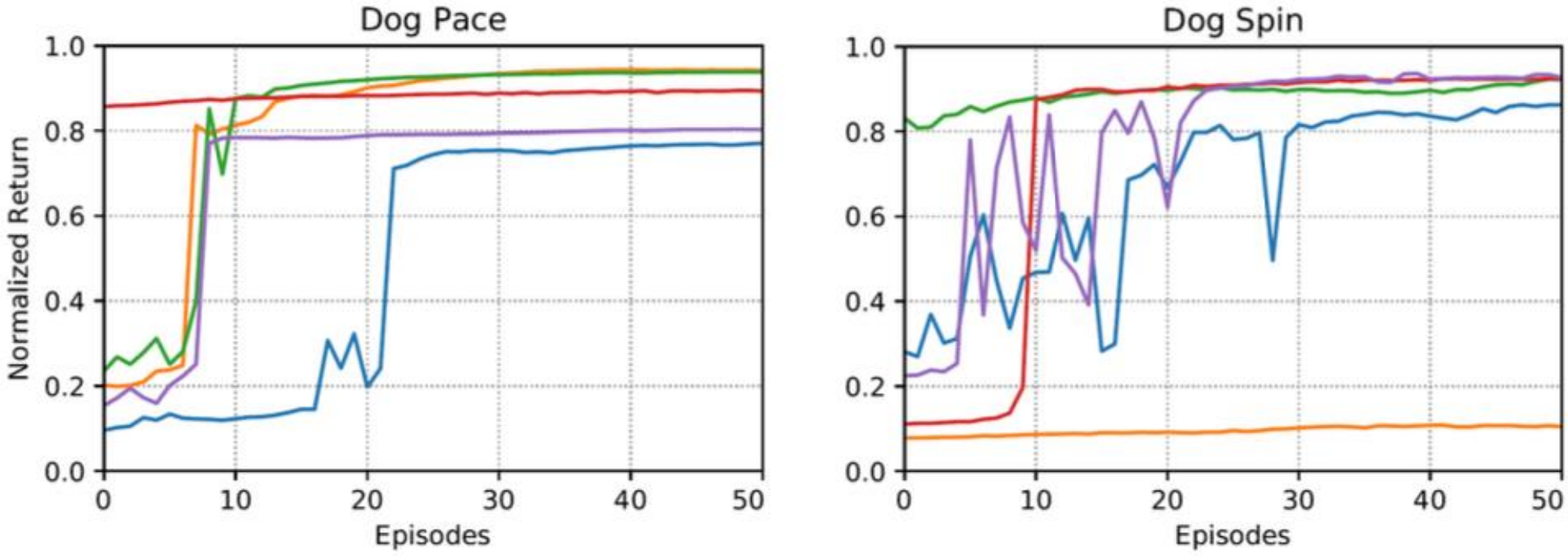
- **Experimental Results**
  - **Performance Across Simulated Environments**



Caption: Performance of each policy type across multiple simulated environments with varied dynamics. Adaptive policies show higher robustness by achieving better results in a larger proportion of these environments.

Adaptability and Performance in Diverse Environments

- **Experimental Results**
  - **Adaptation Across Varied Environments**



Caption: Learning curves of policies across five new environments, showing quick adaptation. Adaptive policies with an information bottleneck adjust efficiently to changing conditions.

Accelerated Video of Results

- Highlights

Pros and Cons

- **Pros**
  - Increased efficiency with dynamic skills
  - Improved adaptability to various environments
  - Easier real-world implementation
  - Better performance compared to manufacturing

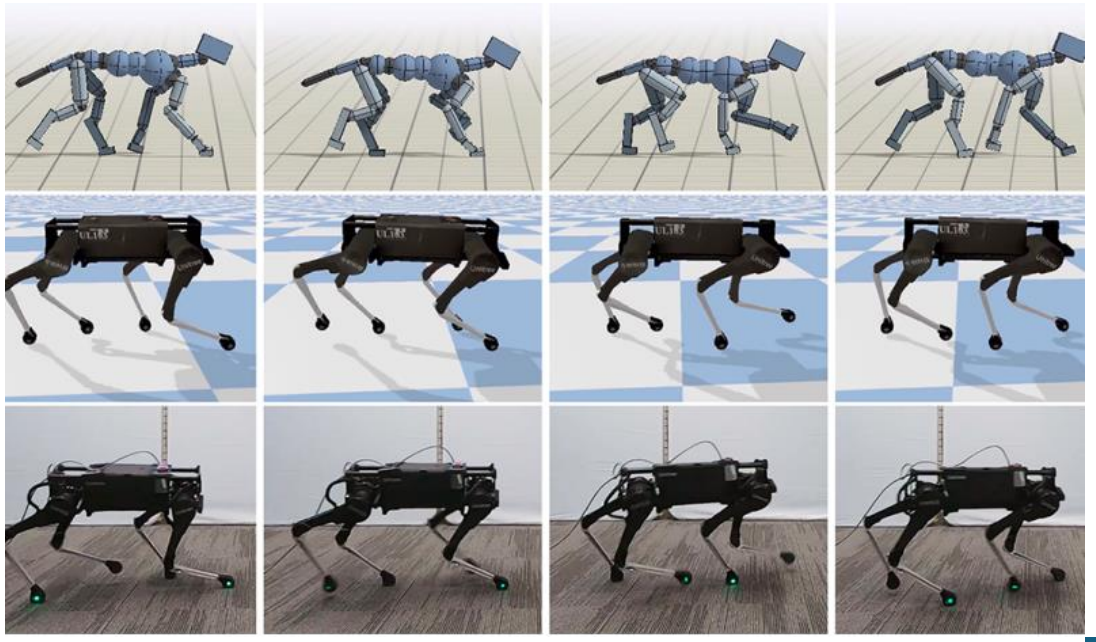
- **Cons**
  - Limited by hardware and algorithmic constraints
  - Less stability compared to top manual controllers
  - Still constrained by real-world limitations

Influence on the Robotics Community

- Kumar, Ashish & Fu, Zipeng & Pathak, Deepak & Malik, Jitendra. (2021). RMA: Rapid Motor Adaptation for Legged Robots. 10.48550/arXiv.2107.04034.

- Li, He & Yu, Wenhao & Zhang, Tingnan & Wensing, Patrick. (2022). Zero-Shot Retargeting of Learned Quadruped Locomotion Policies Using Hybrid Kinodynamic Model Predictive Control. 11971-11977. 10.1109/IROS47612.2022.9981967.

- Klipfel, Arnaud & Sontakke, Nitish & Liu, Ren & Ha, Sehoon. (2023). Learning a Single Policy for Diverse Behaviors on a Quadrupedal Robot Using Scalable Motion Imitation. 2768-2775. 10.1109/IROS55552.2023.10341709.



Q & A

Leen Daher

Marc Mouawad

Matteo Mesa Gonzalez

École polytechnique fédérale de Lausanne

Two possible exam questions

- What are the steps for motion retargeting ?

- Propose a domain adaptation to decrease the gap between simulation and Real World. What is a potential issue to this method, and how would you fix it?