Learning Agile Robotic Locomotion Skills by Imitating Animals

Presenter: Venkatesh Prasad Venkataramanan

28th of May, 2020

Outline

- Introduction
- Related work
- Method
- Experiments
- Conclusion

Introduction

- Reproducing the diverse and agile locomotion skills of animals has been a longstanding challenge in robotics.

- Manually designed controllers can emulate many behaviours, but require effort and expertise.

- In this work, the authors present an imitation learning based approach to automate the tedious task of design of controllers

Related Work

- Performing RL training in real-world is expensive.
- Motion Imitation is successful in sim, but fails in real-world
- Focus is on domain transfer approaches i.e., sim to real world
- Here, domain adaptation is done through a method broadly classified as latent space methods.

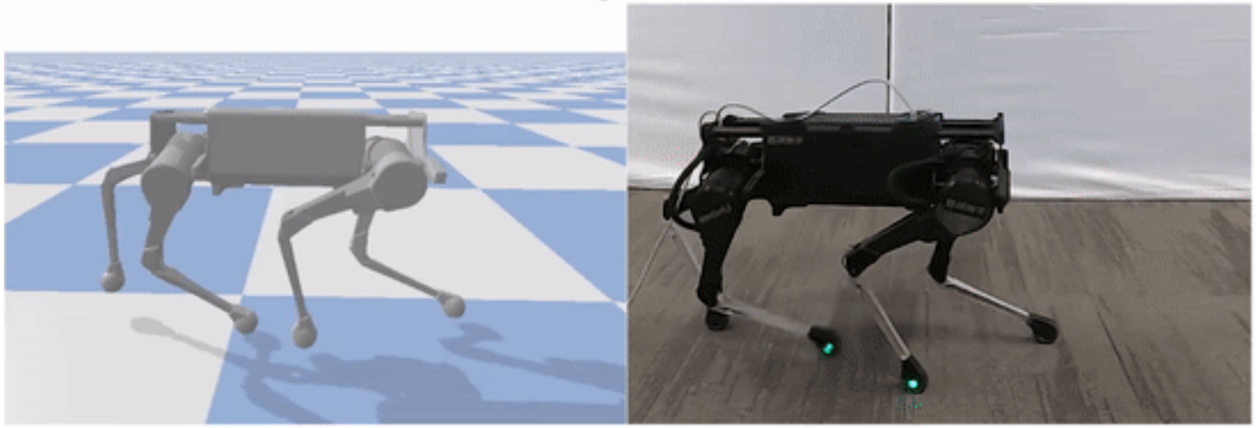
Related Work

- Previous latent space methods used a manually designed reward function, eg ANYmal Robot

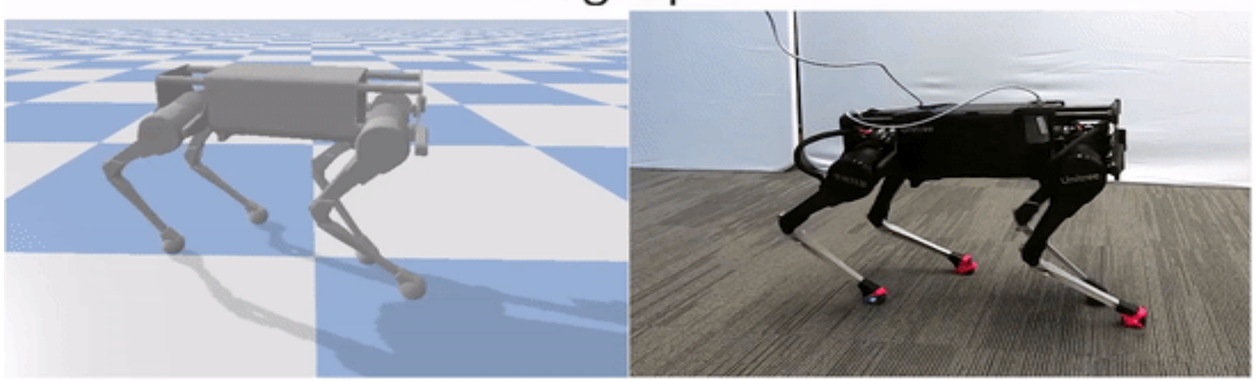
- Motion Imitation can help prevent need for manually designing reward functions.

- Motion Imitation + Latent Space Method = Success !!!

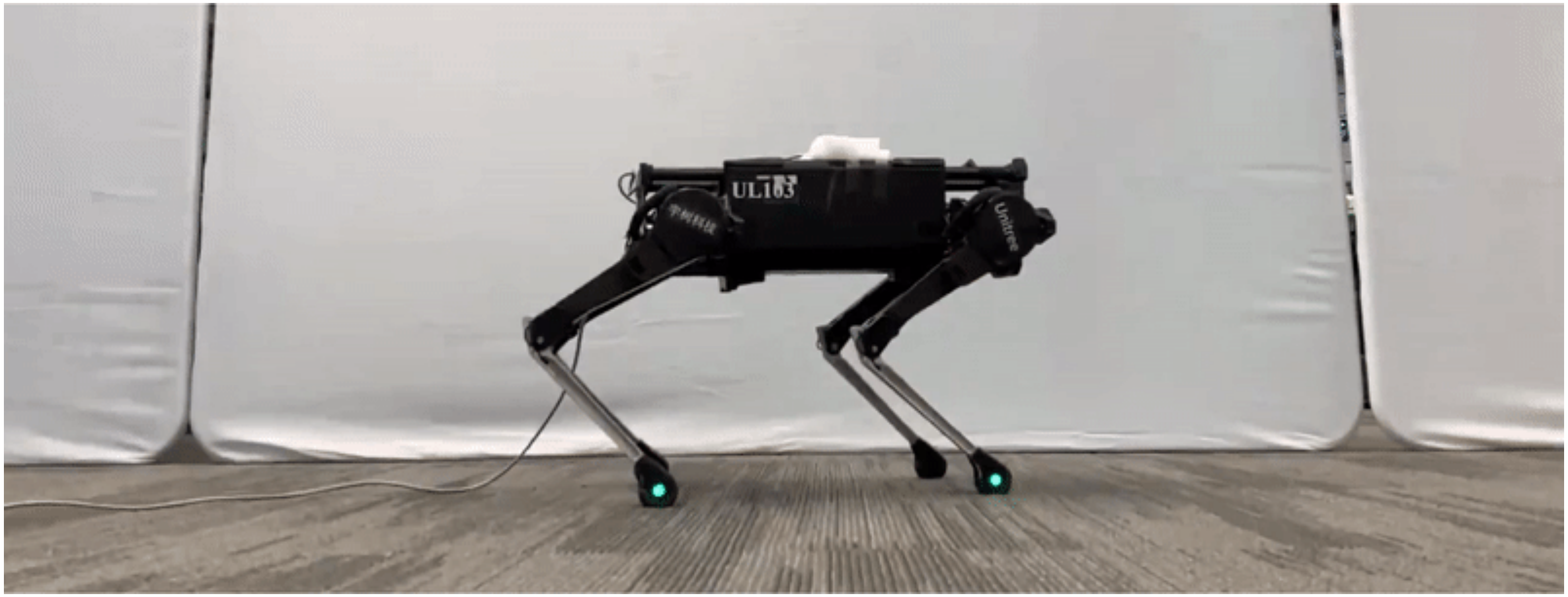
Dog Trot



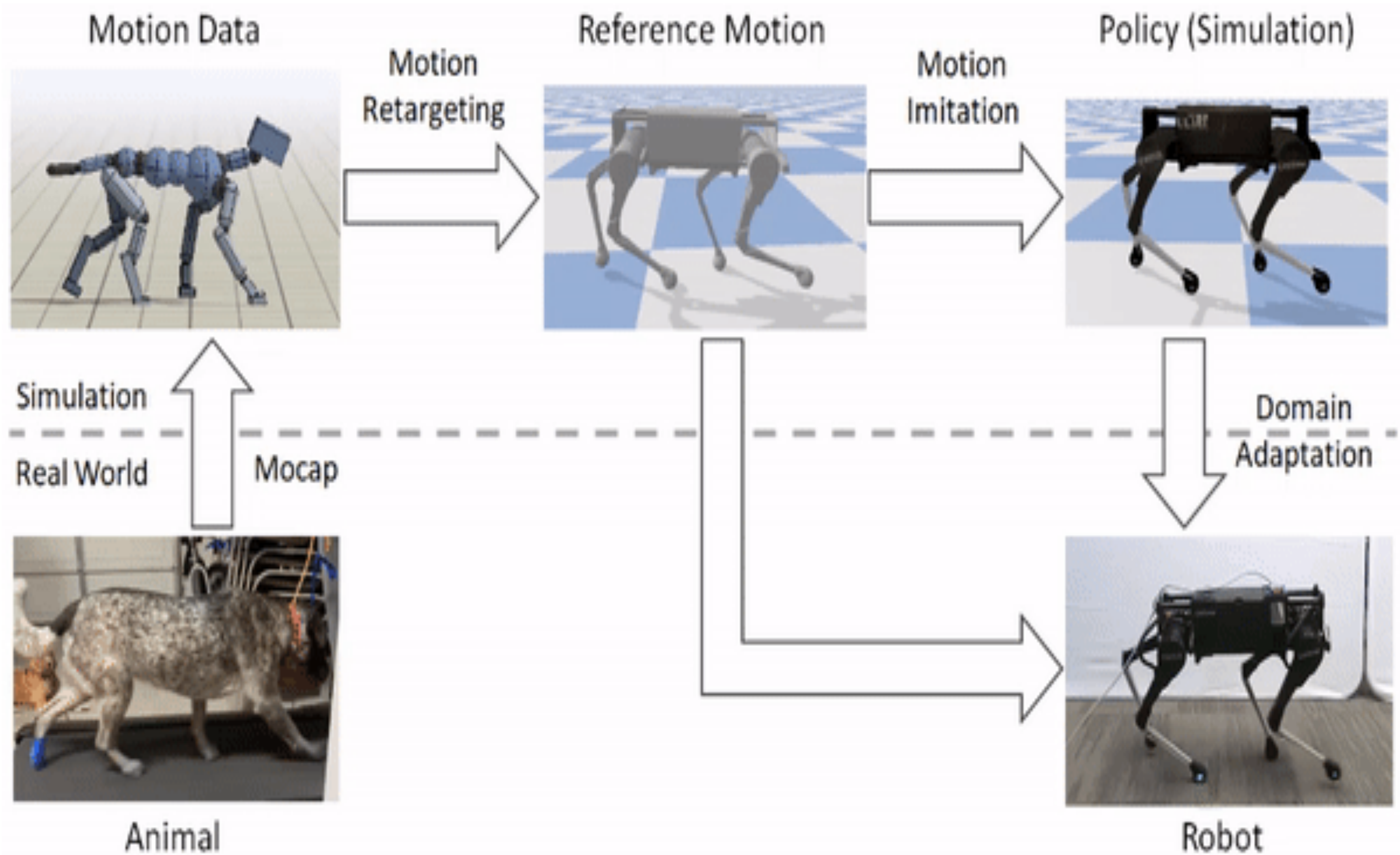
Dog Spin



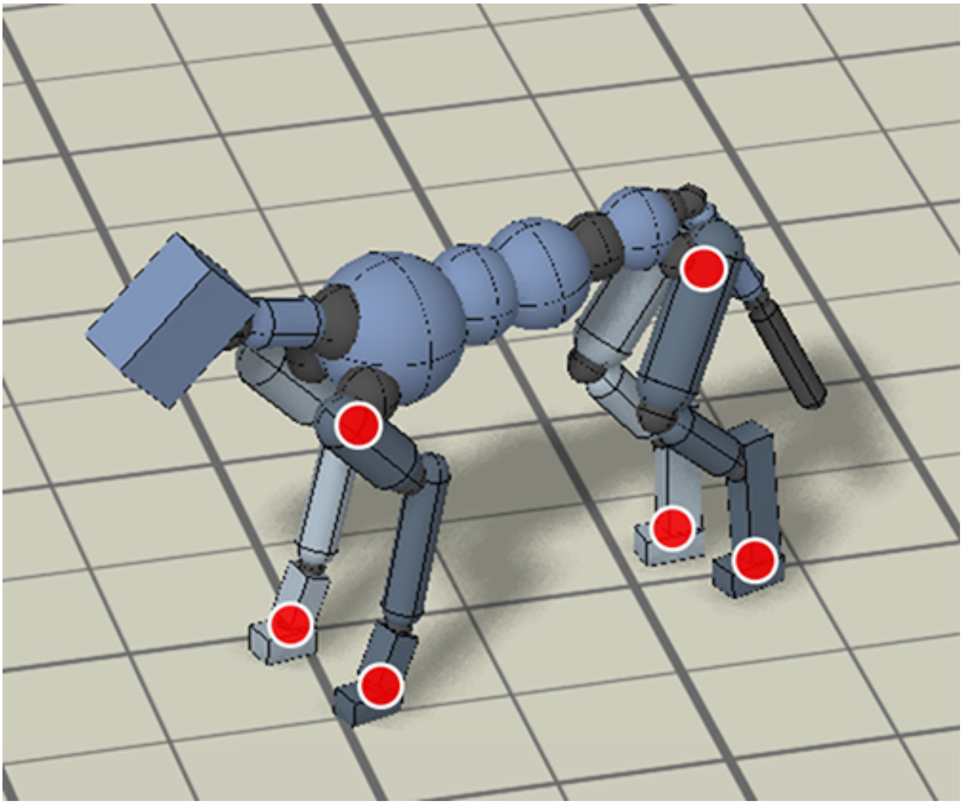
Side-Steps

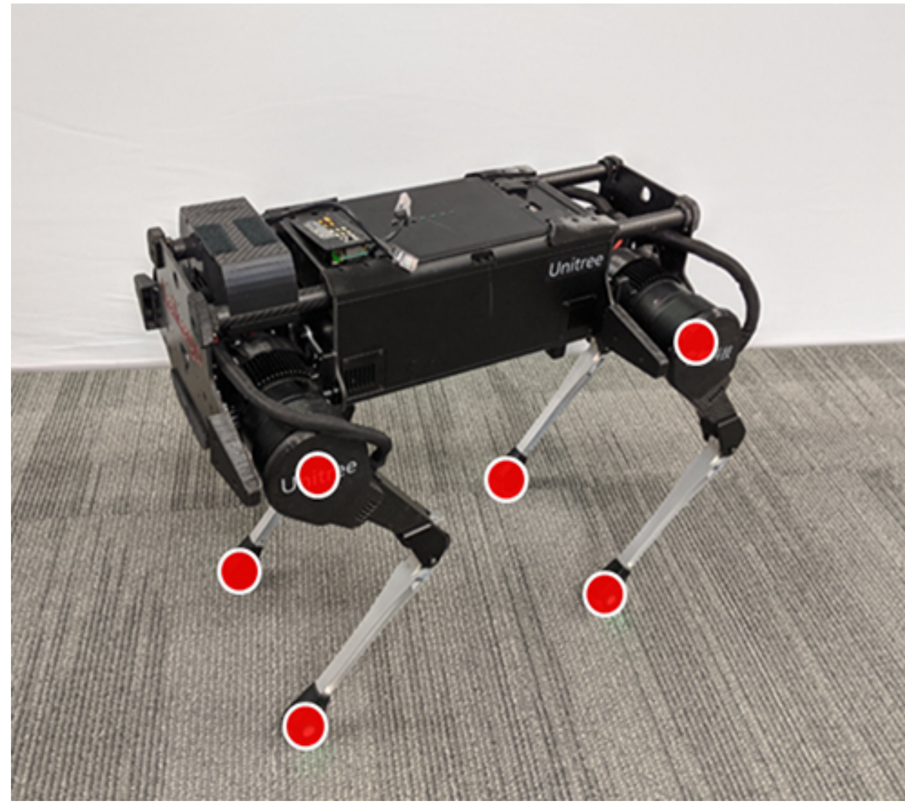


Overview of Method



Motion Retargeting





Motion Retargeting

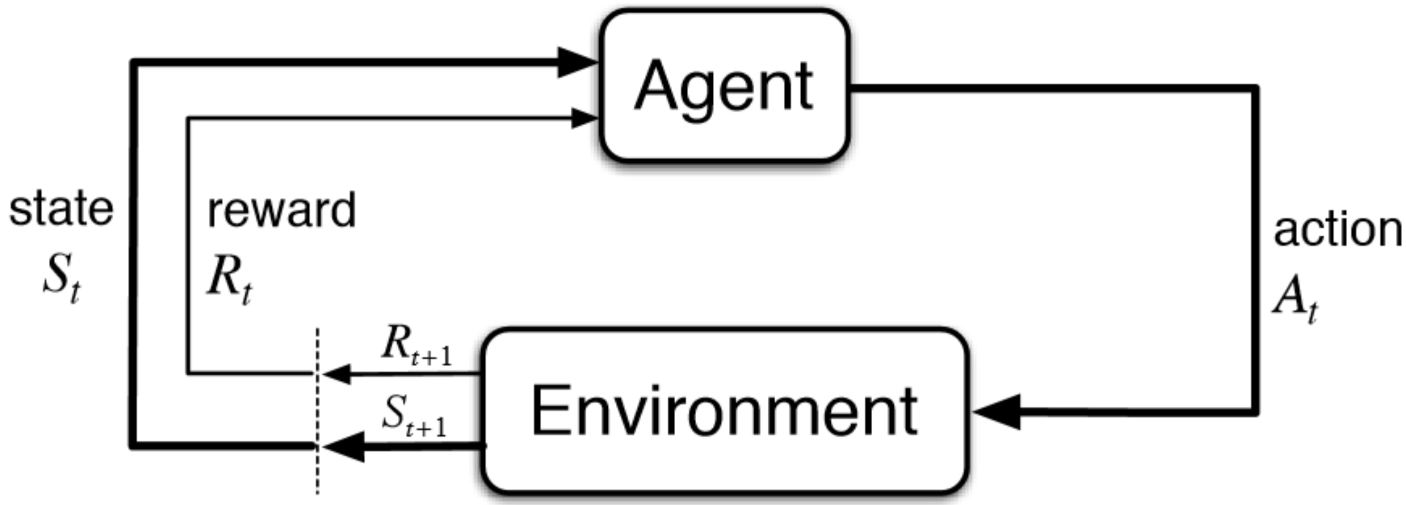
- At each timestep, the source motion specifies the 3D location $x_i(t)$ of each keypoint i.

- The corresponding target keypoint $x_i(q_t)$ is determined by the robot's pose q_t

- IK is then applied to construct a sequence of poses that track the keypoints represented by $q_{0:T}$

$$\arg\min_{\mathbf{q}_{0:T}} \sum_t \sum_i ||\hat{\mathbf{x}}_i(t) - \mathbf{x}_i(\mathbf{q}_t)||^2 + (\bar{\mathbf{q}} - \mathbf{q}_t)^T \mathbf{W} (\bar{\mathbf{q}} - \mathbf{q}_t).$$

Motion Imitation



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

$$p(\tau|\pi) = p(\mathbf{s}_0) \prod_{t=0}^{T-1} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t)$$

Motion Imitation

- Method used is similar to Peng et al.

- The inputs to the policy is augmented with an additional goal g_t, which specifies the motion that the robot should imitate $\pi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{g}_t)$

$$\mathbf{s}_t = (\mathbf{q}_{t-2:t}, \mathbf{a}_{t-3:t-1})$$

- Pose q_t taken from IMU (yaw pitch roll) and joint rotations.

Reward Function

- Again, borrowed from Peng et al.

Joint Pose

Joint Velocity

End Effector Position

Root Position

Root Velocity

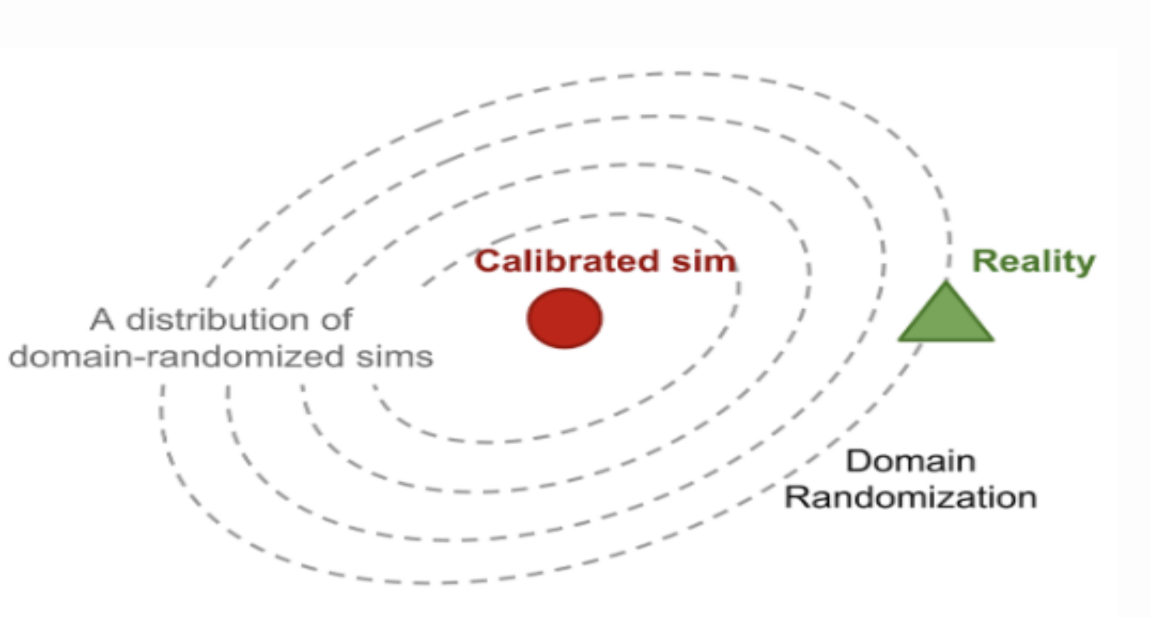
$$r_t = w^{\mathrm{p}} r_t^{\mathrm{p}} + w^{\mathrm{v}} r_t^{\mathrm{v}} + w^{\mathrm{e}} r_t^{\mathrm{e}} + w^{\mathrm{rp}} r_t^{\mathrm{rp}} + w^{\mathrm{rv}} r_t^{\mathrm{rv}}$$

$$w^{\mathrm{p}} = 0.5,\ w^{\mathrm{v}} = 0.05,\ w^{\mathrm{e}} = 0.2,\ w^{\mathrm{rp}} = 0.15,\ w^{\mathrm{rv}} = 0.1$$

$$r_t^{\mathrm{p}} = \exp\left[-5 \sum_j ||\hat{\mathbf{q}}_t^j - \mathbf{q}_t^j||^2\right]$$

Domain Randomization fails!!!

- Instead of training a policy in a single environment with fixed dynamics, domain randomization varies the dynamics during training
- Due to unmodeled effects in the real world, systems may nonetheless fail when deployed in a physical system.

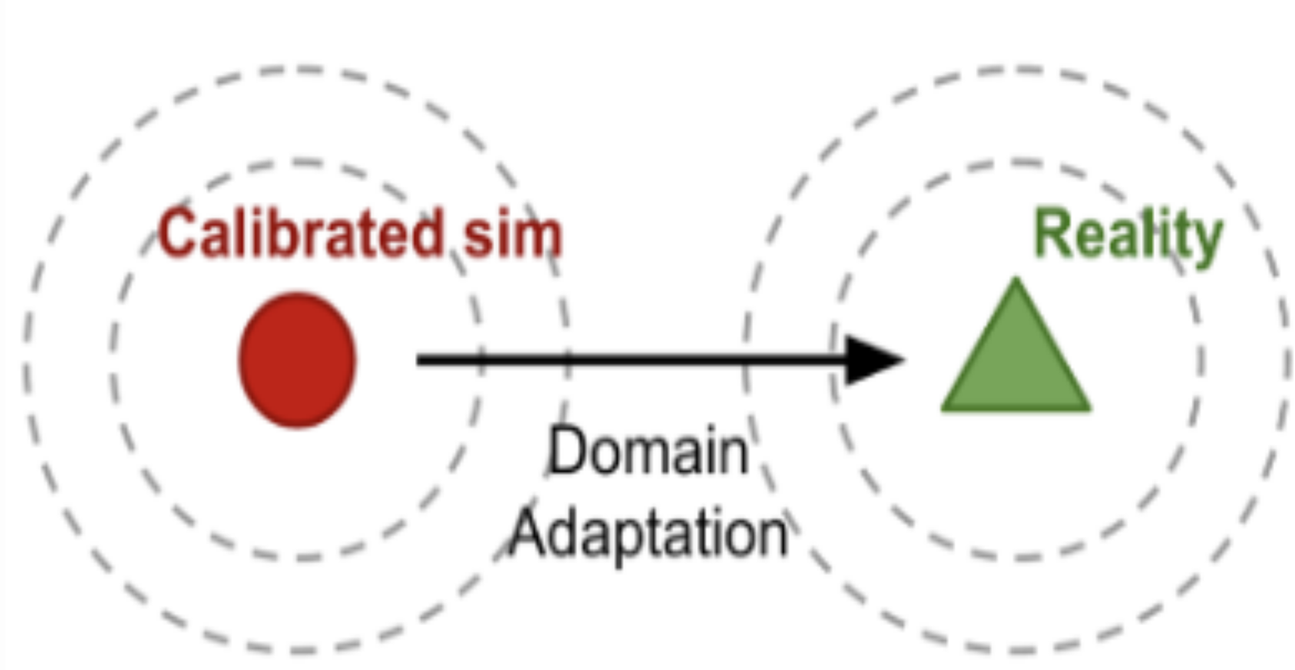


Domain Adaptation

- Search is performed to find a latent encoding $\mathbf{z} \sim E(\mathbf{z}|\boldsymbol{\mu})$

$\boldsymbol{\mu} \sim p(\boldsymbol{\mu})$ → Simulation parameters

Stochastic Encoder



Domain Adaptation

- They incorporate an information bottleneck into the encoder between the dynamics parameters M and the encoding Z

$$I(\mathbf{M}, \mathbf{Z}) \leq I_c.$$

$$\arg\max_{\pi, E} \quad \mathbb{E}_{\boldsymbol{\mu} \sim p(\boldsymbol{\mu})} \mathbb{E}_{\mathbf{z} \sim E(\mathbf{z}|\boldsymbol{\mu})} \mathbb{E}_{\tau \sim p(\tau|\pi, \boldsymbol{\mu}, \mathbf{z})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

Domain Adaptation

$$I(\mathrm{M}, \mathrm{Z}) \leq \mathbb{E}_{\boldsymbol{\mu} \sim p(\boldsymbol{\mu})} \left[\mathrm{D}_{\mathrm{KL}} \left[E(\cdot | \boldsymbol{\mu}) || \rho(\cdot) \right] \right]$$

Stochastic Encoder

Variational Prior

$$\underset{\pi, E}{\arg\max} \; \mathbb{E}_{\boldsymbol{\mu} \sim p(\boldsymbol{\mu})} \mathbb{E}_{\mathbf{z} \sim E(\mathbf{z}|\boldsymbol{\mu})} \mathbb{E}_{\tau \sim p(\tau | \pi, \boldsymbol{\mu}, \mathbf{z})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$
$$- \beta \; \mathbb{E}_{\boldsymbol{\mu} \sim p(\boldsymbol{\mu})} \left[\mathrm{D}_{\mathrm{KL}} \left[E(\cdot | \boldsymbol{\mu}) || \rho(\cdot) \right] \right],$$

Lagrange Multiplier

Algorithm

Algorithm 1 Adaptation with Advantage-Weighted Regression

$\pi \leftarrow$ trained policy
$\omega_0 \leftarrow \mathcal{N}(0, I)$
$\mathcal{D} \leftarrow \emptyset$
for iteration $k = 0, ..., k_{\max} - 1$ **do**
    $\mathbf{z}_k \leftarrow$ sampled encoding from $\omega_k(\mathbf{z})$
    Rollout an episode with π conditioned $\mathbf{z}_k$ and record the return $\mathcal{R}_k$
    Store $(\mathbf{z}_k, \mathcal{R}_k)$ in $\mathcal{D}$
    $\bar{v} \leftarrow \frac{1}{k} \sum_{i=1}^{k} \mathcal{R}_i$
    $\omega_{k+1} \leftarrow \arg\max_{\omega} \sum_{i=1}^{k} \left[\log \omega(\mathbf{z}_i) \exp\left(\frac{1}{\alpha} \left(\mathcal{R}_i - \bar{v}\right)\right)\right]$
end for

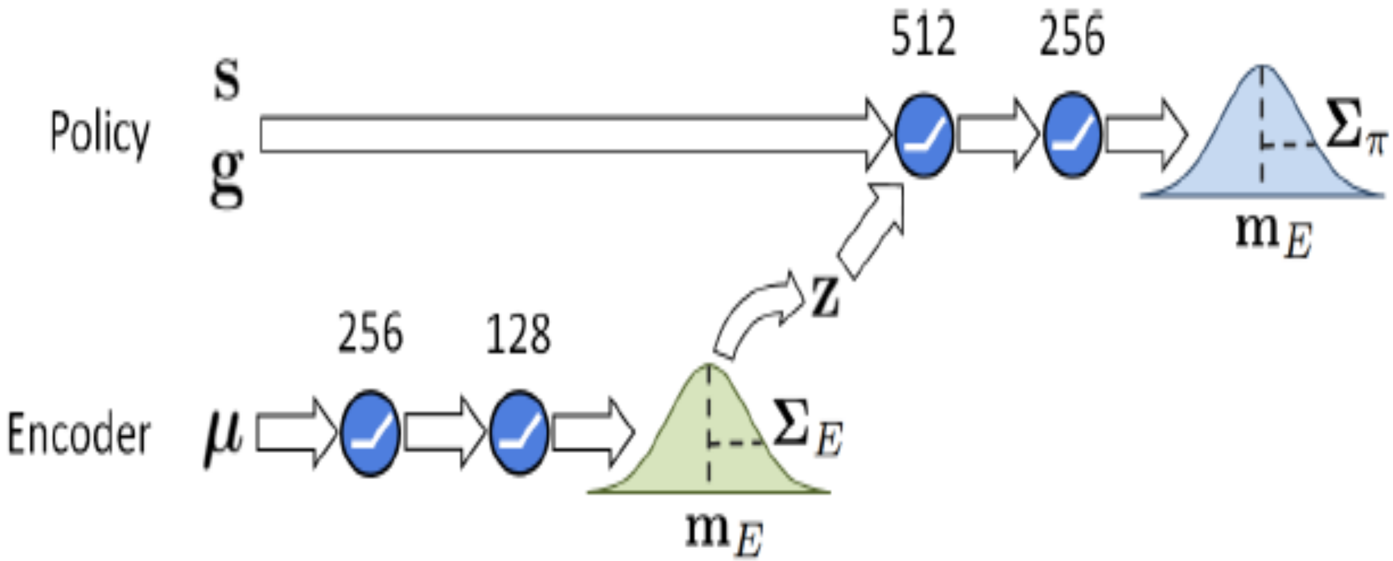
Experimental Setup

- Simulation performed on PyBullet

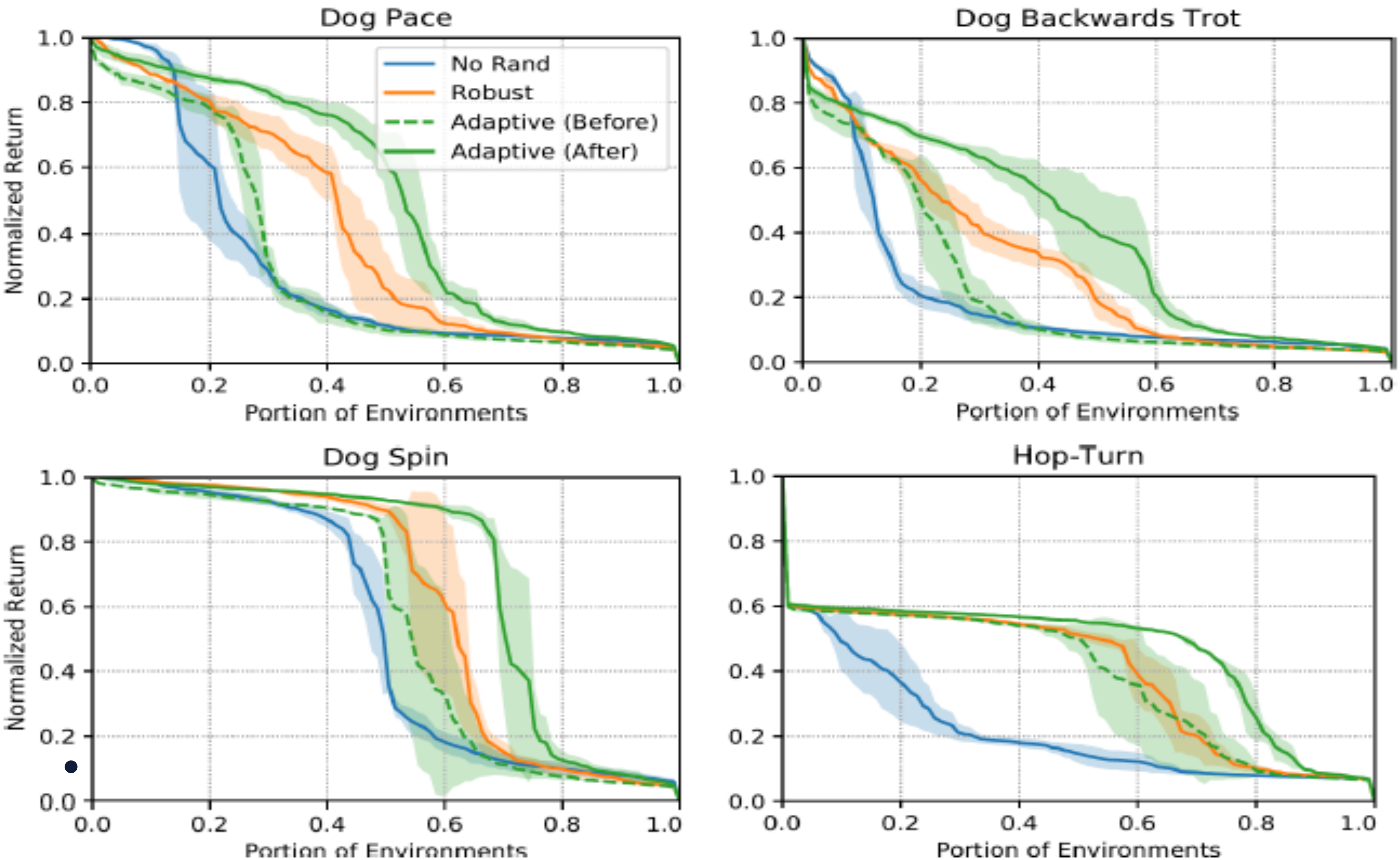
- MoCap clips from dog and artist generated renditions.

- Skills learned
  - Dog Pace
  - Dog Backwards Trot
  - Side-Steps
  - Turn
  - Hop-Turn

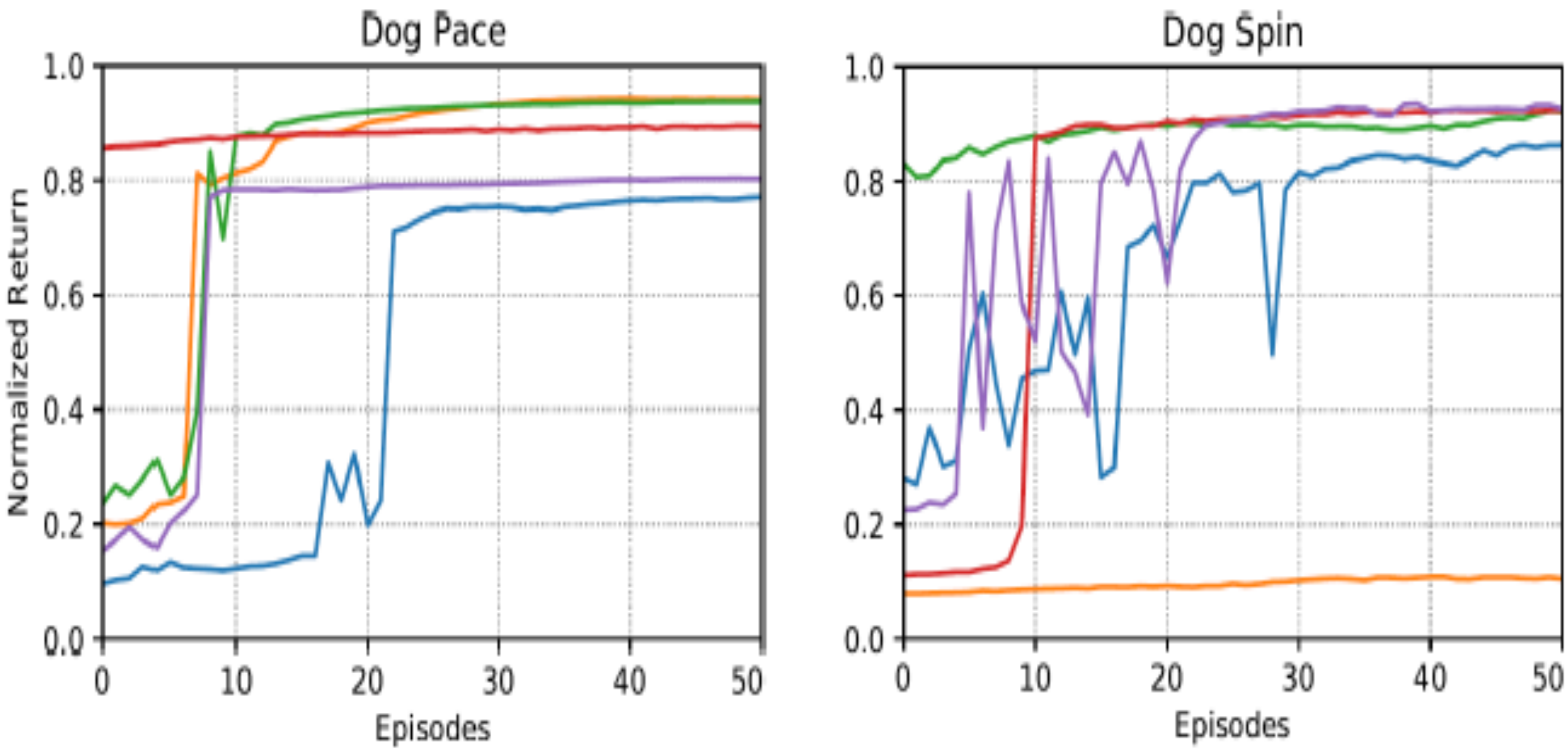
Policy Network



Results and Discussion

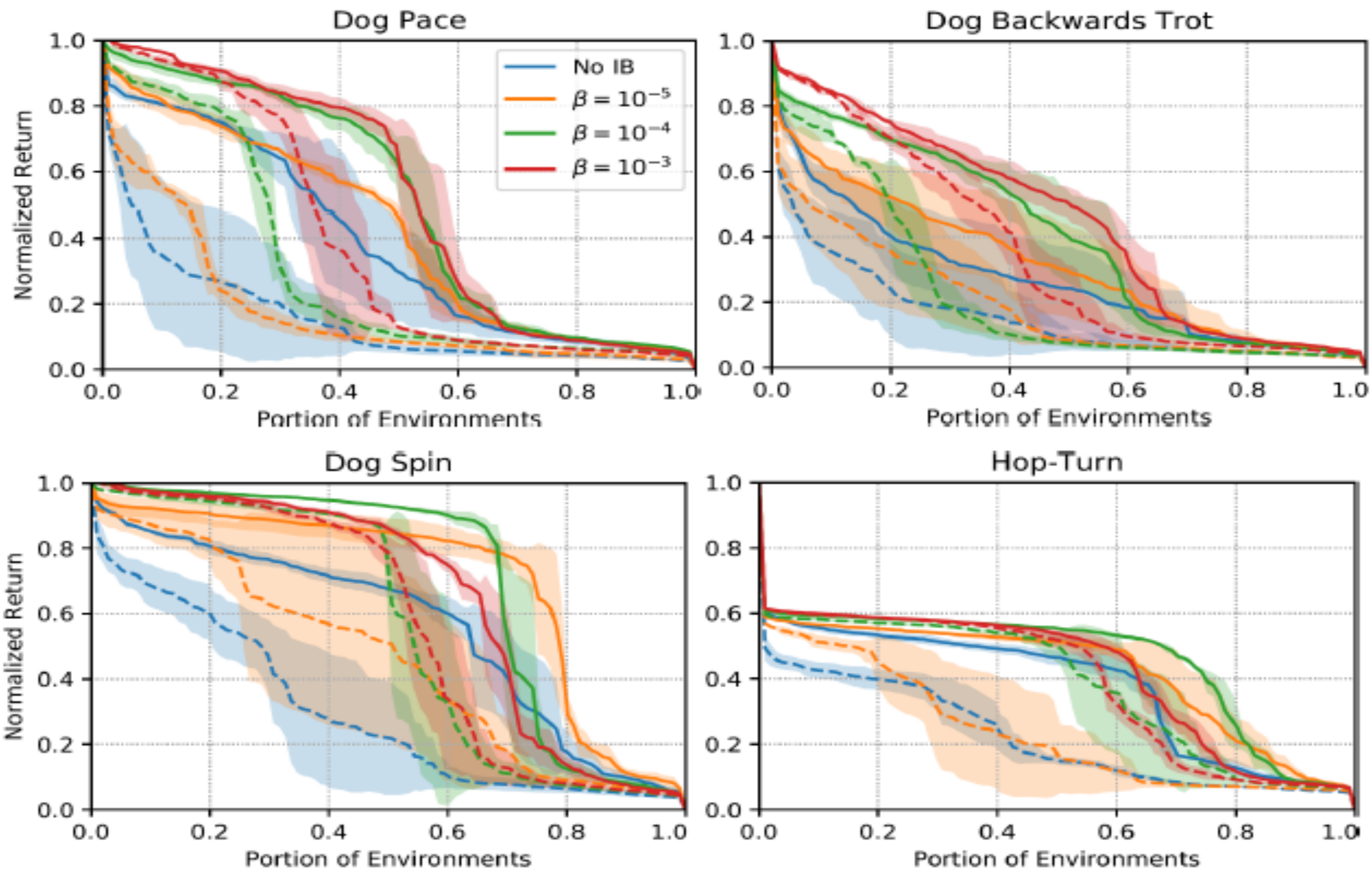


Results and Discussion

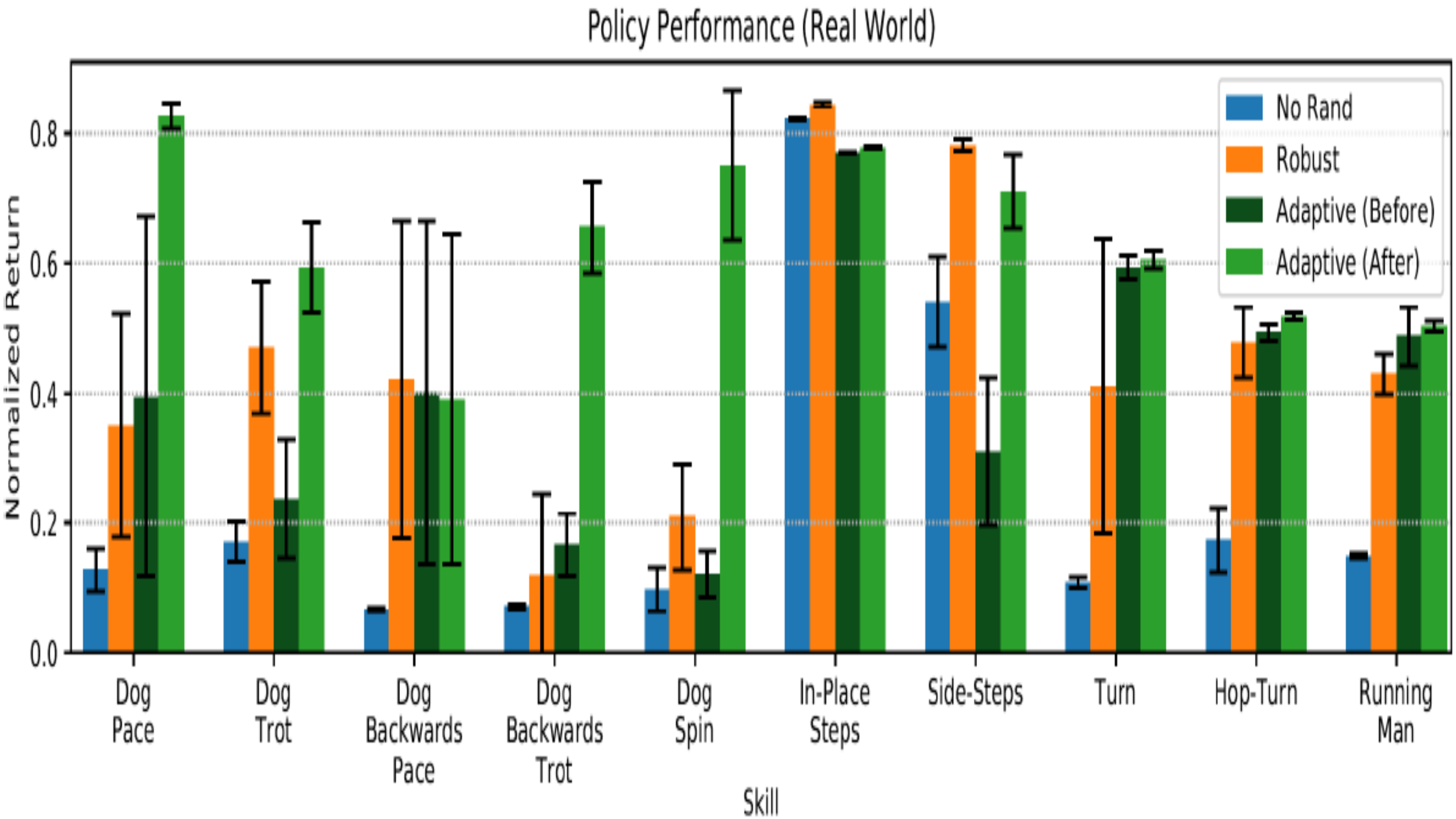


- Faster adaptation to new simulated environments

Results and Discussion



Results and Discussion



Results and Discussion

- Needless to say, the adaptive policies perform way better

- Experiments were conducted to measure normalised return of policies trained without randomisation, with randomisation, and with adaptation.

- Adaptive policies perform better for all kinds of motions.

Conclusion and Future Work

- The behaviours learned by our policies are currently not as stable as the best manually-designed controllers
- More robust controllers
- Learning from other sources of data, like videos

Thanks!!!