Learning Agile Robotic Locomotion Skills by Imitating Animals

Xue Bin Peng*†, Erwin Coumans*, Tingnan Zhang*, Tsang-Wei Edward Lee*, Jie Tan*, Sergey Levine*†
*Google Research, †University of California, Berkeley
Email: xbpeng@berkeley.edu, {erwincoumans,tingnan,tsangwei,jietan}@google.com, svlevine@eecs.berkeley.edu

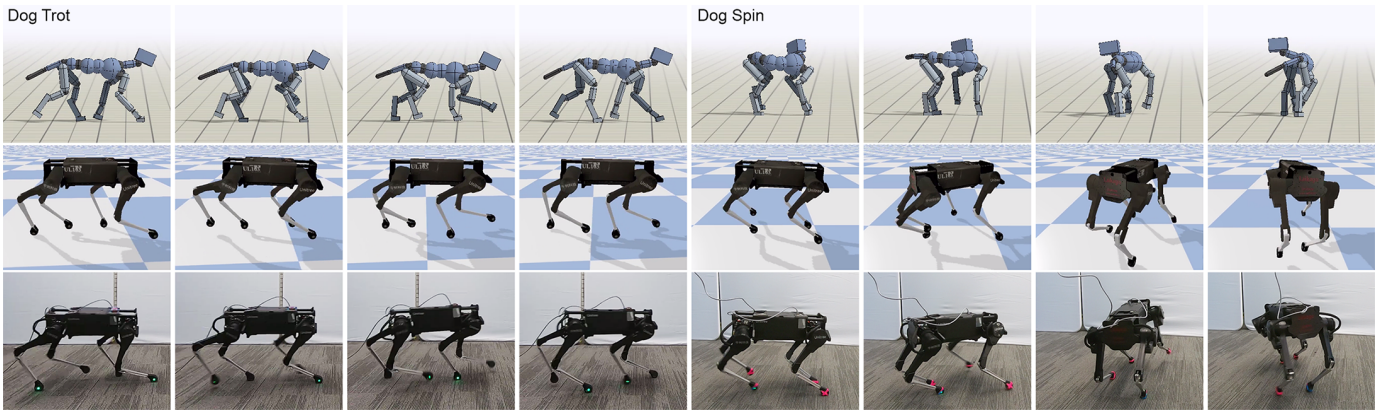

Fig. 1. Laikago robot performing locomotion skills learned by imitating motion data recorded from a real dog. **Top:** Motion capture data recorded from a dog. **Middle:** Simulated Laikago robot imitating reference motions. **Bottom:** Real Laikago robot imitating reference motions.

***Abstract*—Reproducing the diverse and agile locomotion skills of animals has been a longstanding challenge in robotics. While manually-designed controllers have been able to emulate many complex behaviors, building such controllers involves a time-consuming and difficult development process, often requiring substantial expertise of the nuances of each skill. Reinforcement learning provides an appealing alternative for automating the manual effort involved in the development of controllers. However, designing learning objectives that elicit the desired behaviors from an agent can also require a great deal of skill-specific expertise. In this work, we present an imitation learning system that enables legged robots to learn agile locomotion skills by imitating real-world animals. We show that by leveraging reference motion data, a single learning-based approach is able to automatically synthesize controllers for a diverse repertoire behaviors for legged robots. By incorporating sample efficient domain adaptation techniques into the training process, our system is able to learn adaptive policies in simulation that can then be quickly adapted for real-world deployment. To demonstrate the effectiveness of our system, we train an 18-DoF quadruped robot to perform a variety of agile behaviors ranging from different locomotion gaits to dynamic hops and turns. (Video[1])**

I. Introduction

Animals can traverse complex environments with remarkable agility, bringing to bear broad repertoires of agile and acrobatic skills. Reproducing such agile behaviors has been a long-standing challenge in robotics, with a large body of work devoted to designing control strategies for various locomotion skills [37, 49, 54, 18, 3]. However, designing control strategies often involves a lengthy development process, and requires substantial expertise of both the underlying system and the desired skills. Despite the many success in this domain, the capabilities achieved by these systems are still far from the fluid and graceful motions seen in the animal kingdom.

Learning-based approaches offer the potential to improve the agility of legged robots, while also automating a substantial portion of the manual effort involved in the development of controllers. In particular, reinforcement learning (RL) can be an effective and general approach for developing controllers that can perform a wide range of sophisticated skills [7, 43, 25, 44, 34]. While these methods have demonstrated promising results in simulation, agents trained through RL are prone to adopting unnatural behaviors that are dangerous or infeasible when deployed in the real world. Furthermore, designing reward functions that elicit the desired behaviors can itself require a laborious task-specific tuning process.

The comparatively superior agility seen in animals, as compared to robots, might lead one to wonder: can we build more agile robotic controllers with less effort by directly *imitating* animal motions? In this work, we propose an imitation learning framework that enables legged robots to learn agile locomotion skills from real-world animals. Our framework leverages reference motion data to provide priors regarding feasible control strategies for a particular skill. The use of reference motions alleviates the need to design skill-specific reward functions, thereby enabling a common framework to learn a diverse array of behaviors. To address the high sample

[1]Supplementary video: xbpeng.github.io/projects/Robotic_Imitation/

requirements of current RL algorithms, the initial training phase is performed in simulation. In order to transfer policies learned in simulation to the real world, we propose a sample efficient adaptation technique, which fine-tunes the behavior of a policy using a learned dynamics representation.

The central contribution of our work is a system that enables legged robots to learn agile locomotion skills by imitating animals. We demonstrate the effectiveness of our framework on a variety of dynamic locomotion skills with the Laikago quadruped robot [61], including different locomotion gaits, as well as dynamic hops and turns. In our ablation studies, we explore the impact of different design decisions made for the various components of our system.

II. Related Work

The development of controllers for legged locomotion has been an enduring subject of interest in robotics, with a large body of work proposing a variety of control strategies for legged systems [37, 49, 54, 20, 18, 64, 8, 3]. However, many of these methods require in-depth knowledge and manual engineering for each behavior, and as such, the resulting capabilities are ultimately limited by the designer's understanding of how to model and represent agile and dynamic behaviors. Trajectory optimization and model predictive control can mitigate some of the manual effort involved in the design process, but due to the high-dimensional and complex dynamics of legged systems, reduced-order models are often needed to formulate tractable optimization problems [11, 17, 12, 2]. These simplified abstractions tend to be task-specific, and again require significant insight of the properties of each skill.

Motion imitation. Imitating reference motions provides a general approach for robots to perform a rich variety of behaviors that would otherwise be difficult to manually encode into controllers [48, 21, 55, 63]. But applications of motion imitation to legged robots have predominantly been limited to behaviors that emphasize upper-body motions, with fairly static lower-body movements, where balance control can be delegated to separate control strategies [39, 27, 30]. In contrast to physical robots, substantially more dynamic skills can be reproduced by agents in simulation [38, 33, 9, 35]. Recently, motion imitation with reinforcement learning has been effective for learning a large repertoire of highly acrobatic skills in simulation [44, 34, 45, 32]. But due to the high sample complexity of RL algorithms and other physical limitations, many of the capabilities demonstrated in simulation have yet to be replicated in the real world.

Sim-to-real transfer. The challenges of applying RL in the real world have driven the use of domain transfer approaches, where policies are first trained in simulation (source domain), and then transferred to the real world (target domain). Sim-to-real transfer can be facilitated by constructing more accurate simulations [58, 62], or adapting the simulator with real-world data [57, 23, 26, 36, 5]. However, building high-fidelity simulators remains a challenging endeavour, and even state-of-the-art simulators provide only a coarse approximation of the rich dynamics of the real world. Domain randomization can be incorporated into the training process to encourage policies to be robust to variations in the dynamics [52, 60, 47, 42, 41]. Sample efficient adaptation techniques, such as finetuning [51] and meta-learning [13, 16, 6] can also be applied to further improve the performance of pre-trained policies in new domains. In this work, we leverage a class of adaptation techniques, which we broadly referred to as *latent space* methods [24, 65, 67], to transfer locomotion policies from simulation to the real world. During pre-training, these methods learn a latent representation of different behaviors that are effective under various scenarios. When transferring to a new domain, a search can be conducted in the latent space to find behaviors that successfully execute a desired task in the new domain. We show that by combining motion imitation and latent space adaptation, our system is able to learn a diverse corpus of dynamic locomotion skills that can be transferred to legged robots in the real world.

RL for legged locomotion. Reinforcement learning has been effective for automatically acquiring locomotion skills in simulation [44, 34, 32] and in the real world [31, 59, 14, 58, 22, 26]. Kohl and Stone [31] applied a policy gradient method to tune manually-crafted walking controllers for the Sony Aibo robot. By carefully modeling the motor dynamics of the Minitaur quadruped robot, Tan et al. [58] was able to train walking policies in simulation that can be directly deployed on a real robot. Hwangbo et al. [26] proposed learning a motor dynamics model using real-world data, which enabled direct transfer of a variety of locomotion skills to the ANYmal robot. Their system trained policies using manually-designed reward functions for each skill, which can be difficult to specify for more complex behaviors. Imitating reference motions can be a general approach for learning diverse repertoires of skills without the need to design skill-specific reward functions [35, 44, 45]. Xie et al. [62] trained bipedal walking policies for the Cassie robot by imitating reference motions recorded from existing controllers and keyframe animations. The policies are again transferred from simulation to the real world with the aid of careful system identification. Yu et al. [65] transferred bipedal locomotion policies from simulation to a physical Darwin OP2 robot using a latent space adaptation method, which mitigates the dependency on accurate simulators. In this work, we leverage a similar latent space method, but by combining it with motion imitation, our system enables real robots to perform more diverse and agile behaviors than have been demonstrated by these previous methods.

III. Overview

The objective of our framework is to enable robots to learn skills from real animals. Our framework receives as input a reference motion that demonstrates a desired skill for the robot, which may be recorded using motion capture (mocap) of real animals (e.g. a dog). Given a reference motion, it then uses reinforcement learning to synthesize a policy that enables a robot to reproduce that skill in the real world. A schematic

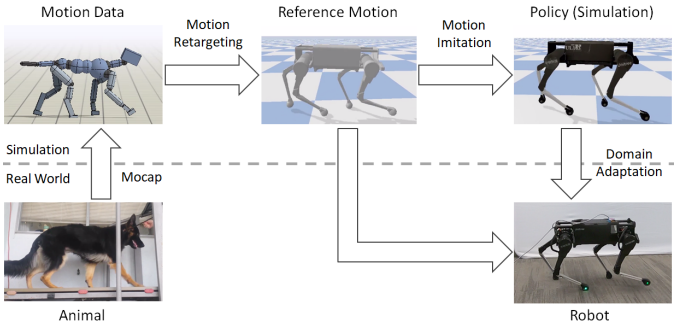

Fig. 2. The framework consists of three stages: motion retargeting, motion imitation, and domain adaptation. It receives as input motion data recorded from an animal, and outputs a control policy that enables a real robot to reproduce the motion.

illustration of our framework is available in Figure 2. The process is organized into three stages: motion retargeting, motion imitation, and domain adaptation. 1) The reference motion is first processed by the motion retargeting stage, where the motion clip is mapped from the original subject's morphology to the robot's morphology via inverse-kinematics. 2) Next, the retargeted reference motion is used in the motion imitation stage to train a policy to reproduce the motion with a simulated model of the robot. To facilitate transfer to the real world, domain randomization is applied in simulation to train policies that can adapt to different dynamics. 3) Finally, the policy is transferred to a real robot via a sample efficient domain adaptation process, which adapts the policy's behavior using a learned latent dynamics representation.

IV. Motion Retargeting

When using motion data recorded from animals, the subject's morphology tends to differ from that of the robot's. To address this discrepancy, the source motions are retargeted to the robot's morphology using inverse-kinematics [19]. First, a set of source keypoints are specified on the subject's body, which are paired with corresponding target keypoints on the robot's body. An illustration of the keypoints is available in Figure 3. The keypoints include the positions of the feet and hips. At each timestep, the source motion specifies the 3D location $\hat{\mathbf{x}}_i(t)$ of each keypoint i. The corresponding target keypoint $\mathbf{x}_i(\mathbf{q}_t)$ is determined by the robot's pose $\mathbf{q}_t$, represented in generalized coordinates [15]. IK is then applied to construct a sequence of poses $\mathbf{q}_{0:T}$ that track the keypoints at each frame,

$$\arg\min_{\mathbf{q}_{0:T}} \sum_t \sum_i ||\hat{\mathbf{x}}_i(t) - \mathbf{x}_i(\mathbf{q}_t)||^2 + (\bar{\mathbf{q}} - \mathbf{q}_t)^T \mathbf{W} (\bar{\mathbf{q}} - \mathbf{q}_t). \quad (1)$$

An additional regularization term is included to encourage the poses to remain similar to a default pose $\bar{\mathbf{q}}$, and $\mathbf{W} = \text{diag}(w_1, w_2, ...)$ is a diagonal matrix specifying regularization coefficients for each joint.

V. Motion Imitation

We formulate motion imitation as a reinforcement learning problem. In reinforcement learning, the objective is to learn a control policy π that enables an agent to maximize its expected return for a given task [56]. At each timestep t, the agent observers a state $\mathbf{s}_t$ from the environment, and samples an action $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$ from its policy π. The agent then applies this action, which results in a new state $\mathbf{s}_{t+1}$ and a scalar reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$. Repeated applications of this process generates a trajectory $\tau = \{(\mathbf{s}_0, \mathbf{a}_0, r_0), (\mathbf{s}_1, \mathbf{a}_1, r_1), ...\}$. The objective then is to learn a policy that maximizes the agent's expected return $J(\pi)$,

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right], \quad (2)$$

where T denotes the time horizon of each episode, and $\gamma \in [0, 1]$ is a discount factor. $p(\tau|\pi)$ represents the likelihood of a trajectory τ under a given policy π,

$$p(\tau|\pi) = p(\mathbf{s}_0) \prod_{t=0}^{T-1} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \pi(\mathbf{a}_t|\mathbf{s}_t), \quad (3)$$

with $p(\mathbf{s}_0)$ being the initial state distribution, and $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ representing the dynamics of the system, which determines the effects of the agent's actions.

To imitate a given reference motion, we follow a similar motion imitation approach as Peng et al. [44]. The inputs to the policy is augmented with an additional goal $\mathbf{g}_t$, which specifies the motion that the robot should imitate. The policy is modeled as a feedforward network that maps a given state $\mathbf{s}_t$ and goal $\mathbf{g}_t$ to a distribution over actions $\pi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{g}_t)$. The policy is queried at 30Hz for a new action at each timestep. The state $\mathbf{s}_t = (\mathbf{q}_{t-2:t}, \mathbf{a}_{t-3:t-1})$ is represented by the poses $\mathbf{q}_{t-2:t}$ of the robot in the three previous timesteps, and the three previous actions $\mathbf{a}_{t-3:t-1}$. The pose features $\mathbf{q}_t$ consist of IMU readings of the root orientation (row, pitch, yaw) and the local rotations of every joint. The root position is not included among the pose features to avoid the need to estimate the root position during real-world deployment. The goal $\mathbf{g}_t = (\hat{\mathbf{q}}_{t+1}, \hat{\mathbf{q}}_{t+2}, \hat{\mathbf{q}}_{t+10}, \hat{\mathbf{q}}_{t+30})$ specifies target poses from the reference motion at four future timesteps, spanning approximately 1 second. The action $\mathbf{a}_t$ specifies target rotations for PD controllers at each joint. To ensure smoother motions, the PD targets are first processed by a low-pass filter before being applied on the robot [4].

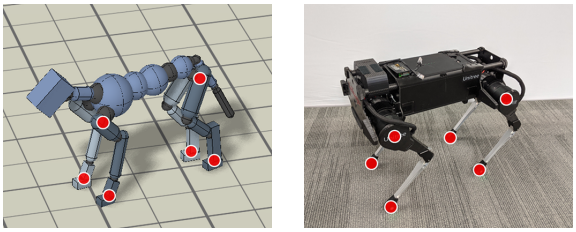

Fig. 3. Inverse-kinematics (IK) is used to retarget mocap clips recorded from a real dog (**left**) to the Laikago robot (**right**). Corresponding pairs of keypoints (red) are specified on the dog and robot's bodies, and then IK is used to compute a pose for the robot that tracks the keypoints.

Reward Function. The reward function encourages the policy to track the sequence of target poses $(\hat{\mathbf{q}}_0, \hat{\mathbf{q}}_1, ..., \hat{\mathbf{q}}_T)$ from the

reference motion at every timestep. The reward function is similar to the one used by Peng et al. [44], where the reward r_t at each timestep is given by:

$$r_t = w^{\text{p}} r_t^{\text{p}} + w^{\text{v}} r_t^{\text{v}} + w^{\text{e}} r_t^{\text{e}} + w^{\text{rp}} r_t^{\text{rp}} + w^{\text{rv}} r_t^{\text{rv}} \tag{4}$$

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

The pose reward r_t^{p} encourages the robot to minimize the difference between the joint rotations specified by the reference motion and those of the robot. In the equation below, $\hat{\mathbf{q}}_t^j$ represents the 1D local rotation of joint j from the reference motion at time t, and $\mathbf{q}_t^j$ represents the robot's joint,

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

Similarly, the velocity reward r_t^{v} is calculated according to the joint velocities, with $\hat{\dot{\mathbf{q}}}_t^j$ and $\dot{\mathbf{q}}_t^j$ being the angular velocity of joint j from the reference motion and robot respectively,

$$r_t^{\text{v}} = \exp\left[-0.1 \sum_j ||\hat{\dot{\mathbf{q}}}_t^j - \dot{\mathbf{q}}_t^j||^2\right]. \tag{6}$$

Next, the end-effector reward r_t^{e}, encourages the robot to track the positions of the end-effectors, where $\mathbf{x}_t^e$ denotes the relative 3D position of end-effector e with respect to the root,

$$r_t^{\text{e}} = \exp\left[-40 \sum_e ||\hat{\mathbf{x}}_t^e - \mathbf{x}_t^e||^2\right]. \tag{7}$$

Finally, the root pose reward r_t^{rp} and root velocity reward r_t^{rv} encourage the robot to track the reference root motion. $\mathbf{x}_t^{\text{root}}$ and $\dot{\mathbf{x}}_t^{\text{root}}$ denotes the root's global position and linear velocity, while $\mathbf{q}_t^{\text{root}}$ and $\dot{\mathbf{q}}_t^{\text{root}}$ are the rotation and angular velocity,

$$r_t^{\text{rp}} = \exp\left[-20||\hat{\mathbf{x}}_t^{\text{root}} - \mathbf{x}_t^{\text{root}}||^2 - 10||\hat{\mathbf{q}}_t^{\text{root}} - \mathbf{q}_t^{\text{root}}||^2\right] \tag{8}$$

$$r_t^{\text{rv}} = \exp\left[-2||\hat{\dot{\mathbf{x}}}_t^{\text{root}} - \dot{\mathbf{x}}_t^{\text{root}}||^2 - 0.2||\hat{\dot{\mathbf{q}}}_t^{\text{root}} - \dot{\mathbf{q}}_t^{\text{root}}||^2\right]. \tag{9}$$

VI. Domain Adaptation

Due to discrepancies between the dynamics of the simulation and the real world, policies trained in simulation tend to perform poorly when deployed on a physical system. Therefore, we propose a sample efficient adaptation technique for transferring policies from simulation to the real world.

A. Domain Randomization

Domain randomization is a simple strategy for improving a policy's robustness to dynamics variations [52, 60, 42]. Instead of training a policy in a single environment with fixed dynamics, domain randomization varies the dynamics during training, thereby encouraging the policy to learn strategies that are functional across different dynamics. However, there may be no single strategy that is effective across all environments, and due to unmodeled effects in the real world, strategies that are robust to different simulated dynamics may nonetheless fail when deployed in a physical system.

B. Domain Adaptation

In this work, we aim to learn strategies that are robust to variations in the dynamics of the environment, while also being able to adapt its behaviors as necessary for new environments. Let $\boldsymbol{\mu}$ represent the values of the dynamics parameters that are randomized during training in simulation (Table I). At the start of each episode, a random set of parameters are sampled according to $\boldsymbol{\mu} \sim p(\boldsymbol{\mu})$. The dynamics parameters are then encoded into a latent embedding $\mathbf{z} \sim E(\mathbf{z}|\boldsymbol{\mu})$ by a stochastic encoder E, and $\mathbf{z}$ is provided as an additional input to the policy $\pi(\mathbf{a}|\mathbf{s}, \mathbf{z})$. For brevity, we have excluded the goal input $\mathbf{g}$ for the policy. When transferring a policy to the real world, we follow a similar approach as Yu et al. [66], where a search is performed to find a latent encoding $\mathbf{z}^*$ that enables the policy to successfully execute the desired behaviors on the physical system. Next, we propose an extension that addresses potential issues due to over-fitting with the previously proposed method.

A potential degeneracies of the previously described approach is that the policy may learn strategies that depend on $\mathbf{z}$ being an accurate representation of the true dynamics of the system. This can result in brittle behaviors where the strategies utilized by the policy for a given $\mathbf{z}$ can overfit to the precise dynamics from the corresponding parameters $\boldsymbol{\mu}$. Furthermore, due to unmodeled effects in the real world, there might be no $\boldsymbol{\mu}$ that accurately models real-world dynamics. Therefore, to encourage the policy to be robust to uncertainty in the dynamics, we incorporate an information bottleneck into the encoder. The information bottleneck enforces an upper bound I_c on the mutual information $I(\mathbf{M}, \mathbf{Z})$ between the dynamics parameters $\mathbf{M}$ and the encoding $\mathbf{Z}$. This results in the following constrained policy optimization objective,

$$\arg\max_{\pi, E} \quad \mathbb{E}_{\boldsymbol{\mu}\sim p(\boldsymbol{\mu})} \mathbb{E}_{\mathbf{z}\sim E(\mathbf{z}|\boldsymbol{\mu})} \mathbb{E}_{\tau\sim p(\tau|\pi,\boldsymbol{\mu},\mathbf{z})} \left[\sum_{t=0}^{T-1} \gamma^t r_t\right] \tag{10}$$

$$\text{s.t.} \quad I(\mathbf{M}, \mathbf{Z}) \leq I_c. \tag{11}$$

where the trajectory distribution is now given by,

$$p(\tau|\pi, \boldsymbol{\mu}, \mathbf{z}) = p(\mathbf{s}_0) \prod_{t=0}^{T-1} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \boldsymbol{\mu}) \pi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z}). \tag{12}$$

Since computing the mutual information is intractable, the constraint in Equation 11 can be approximated with a variational upper bound using the KL divergence between E and a variational prior $\rho(\mathbf{z})$ [1],

$$I(\mathbf{M}, \mathbf{Z}) \leq \mathbb{E}_{\boldsymbol{\mu}\sim p(\boldsymbol{\mu})} \left[\mathrm{D}_{\mathrm{KL}} \left[E(\cdot|\boldsymbol{\mu})||\rho(\cdot)\right]\right]. \tag{13}$$

We can further simplify the objective by converting Equation 11 into a soft constraint, to yield the following information-regularized objective,

$$\arg\max_{\pi, E} \mathbb{E}_{\boldsymbol{\mu}\sim p(\boldsymbol{\mu})} \mathbb{E}_{\mathbf{z}\sim E(\mathbf{z}|\boldsymbol{\mu})} \mathbb{E}_{\tau\sim p(\tau|\pi,\boldsymbol{\mu},\mathbf{z})} \left[\sum_{t=0}^{T-1} \gamma^t r_t\right] - \beta\ \mathbb{E}_{\boldsymbol{\mu}\sim p(\boldsymbol{\mu})} \left[\mathrm{D}_{\mathrm{KL}} \left[E(\cdot|\boldsymbol{\mu})||\rho(\cdot)\right]\right], \tag{14}$$

with $\beta \geq 0$ being a Lagrange multiplier. In our experiments, we model the encoder $E(\mathbf{z}|\boldsymbol{\mu}) = \mathcal{N}(\mathbf{m}(\boldsymbol{\mu}), \boldsymbol{\Sigma}(\boldsymbol{\mu}))$ as a

Algorithm 1 Adaptation with Advantage-Weighted Regression

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

Gaussian distribution with mean $\mathbf{m}(\boldsymbol{\mu})$ and standard deviation $\boldsymbol{\Sigma}(\boldsymbol{\mu})$, and the prior $\rho(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ is given by the unit Gaussian. This objective can be interpreted as training a policy that maximizes the agent's expected return across different dynamics, while also being able to adapt its behaviors when necessary by relying on only a minimal amount of information from the ground-truth dynamics parameters. In our formulation, the Lagrange multiplier β provides a trade-off between robustness and adaptability. Large values of β restrict the amount of information that the policy can access from $\boldsymbol{\mu}$. In the limit $\beta \rightarrow \infty$, the policy converges to a robust but non-adaptive policy that does not access the underlying dynamics parameters. Conversely, small values of $\beta \rightarrow 0$ provides the policy with unfettered access to the dynamics parameters, which can result in brittle strategies where the policy's behaviors overfit to the nuances of each setting of the dynamics parameters, potentially leading to poor generalization to real-world dynamics.

C. Real World Transfer

To adapt a policy to the real world, we directly search for an encoding $\mathbf{z}$ that maximizes the return on the physical system

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} \quad \mathbb{E}_{\tau \sim p^*(\tau|\pi,\mathbf{z})} \left[\sum_{t=0}^{T-1} \gamma^t r_t\right], \tag{15}$$

with $p^*(\tau|\pi, \mathbf{z})$ being the trajectory distribution under real-world dynamics. To identify $\mathbf{z}^*$, we use advantage-weighted regression (AWR) [40, 46], a simple off-policy RL algorithm. Algorithm 1 summarizes the adaptation process. The search distribution is initialized with the prior $\omega_0(\mathbf{z}) = \mathcal{N}(0, I)$. At each iteration k, we sample an encoding from the current distribution $\mathbf{z}_k \sim \omega_k(\mathbf{z})$ and execute an episode with the policy conditioned on $\mathbf{z}_k$. The return $\mathcal{R}_k$ for the episode is recorded and stored along with $\mathbf{z}_k$ in a replay buffer $\mathcal{D}$ containing all samples from previous iterations. $\omega_k(\mathbf{z})$ is then updated by fitting a new distribution that assigns higher likelihoods to samples with larger advantages. The likelihood of each sample $\mathbf{z}_i$ is weighted by the exponentiated-advantage $\exp\left(\frac{1}{\alpha}\left(\mathcal{R}_i - \bar{v}\right)\right)$, where the baselines $\bar{v}$ is the average return of all samples in $\mathcal{D}$, and α is a manually specified temperature parameter. Note that, since $\omega_k(\mathbf{z})$ is Gaussian, the optimal distribution at each iteration (Line 9) can be determined analytically. However, we found that the analytic solution is prone to premature convergence to a suboptimal solution. Instead, we update $\omega_k(\mathbf{z})$ incrementally using a few steps of gradient descent. This process is repeated for $k_{\max}$ iterations, and the mean of the final distribution $\omega_{k_{\max}}(\mathbf{z})$ is used as an approximation of the optimal encoding $\mathbf{z}^*$ for deploying the policy in the real world.

Parameter	Training Range	Testing Range
Mass	$[0.8, 1.2]$ × default value	$[0.5, 2.0]$ × default value
Inertia	$[0.5, 1.5]$ × default value	$[0.4, 1.6]$ × default value
Motor Strength	$[0.8, 1.2]$ × default value	$[0.7, 1.3]$ × default value
Motor Friction	$[0, 0.05]$ Nms/rad	$[0, 0.075]$ Nms/rad
Latency	$[0, 0.04]$ s	$[0, 0.05]$ s
Lateral Friction	$[0.05, 1.25]$ Ns/m	$[0.04, 1.35]$ Ns/m

TABLE I
DYNAMIC PARAMETERS AND THEIR RESPECTIVE RANGE OF VALUES USED DURING TRAINING AND TESTING. A LARGER RANGE OF VALUES ARE USED DURING TESTING TO EVALUATE THE POLICIES' ABILITY TO GENERALIZE TO UNFAMILIAR DYNAMICS.

VII. EXPERIMENTAL EVALUATION

We evaluate our robotic learning system by learning to imitating a variety of dynamic locomotion skills using the Laikago robot [61], an 18 degrees-of-freedom quadruped with 3 actuated degrees-of-freedom per leg, and 6 under-actuated degrees of freedom for the root (torso). Behaviors learned by the policies are best seen in the supplementary video[1], and snapshots of the behaviors are also available in Figure 4. In the following experiments, we aim to evaluate the effectiveness of our framework on learning a diverse set of quadruped skills, and study how well real-world adaptation can enable more agile behaviors. We show that our adaptation method can efficiently transfer policies trained in simulation to the real world with a small number of trials on the physical system. We further study the effects of regularizing the latent dynamics encoding with an information bottleneck, and show that this provides a mechanism to trade off between the robustness and adaptability of the learned policies.

A. Experimental Setup

Retargeting via inverse-kinematics and simulated training is performed using PyBullet [10]. Table I summarizes the dynamics parameters and their respective range of values. The motion dataset contains a mixture of mocap clips recorded from a dog and clips from artist generated animations. The mocap clips are collected from a public dataset [68] and re-targeted to the Laikago following the procedure in Section IV. Figure 5 lists the skills learned by the robot and summarizes the performance of the policies when deployed in the real world. Motion clips recorded from a dog are designated with "Dog", and the other clips correspond to artist animated motions. Performance is recorded as the average normalized return, with 0 corresponding to the minimum possible return per episode and 1 being the maximum return. Note that the maximum return may not be achievable, since the reference motions are generally not physically feasible for the robot. Performance is calculated using the average of 3 policies

(a) Dog Pace (b) Dog Backwards Trot

(c) Side-Steps (d) Turn

(e) Hop-Turn (f) Running Man

Fig. 4. Laikago robot performing skills learned by imitating reference motions. **Top:** Reference motion. **Middle:** Simulated robot. **Bottom:** Real robot.

initialized with different random seeds. Each policy is trained with proximal policy optimization using about 200 million samples in simulation [53]. Both the encoder and policy are trained end-to-end using the reparameterization trick [29]. Domain adaptation is performed on the physical system with AWR in the latent dynamics space, using approximately 50 real-world trials to adapt each policy. Trials vary between 5s and 10s in length depending on the space requirements of each skill. Hyperparameter settings are available in Appendix A.

Model representation. All policies are modeled using the neural network architecture shown in Figure 6. The encoder $E(\mathbf{z}|\boldsymbol{\mu})$ is represented by a fully-connected network that maps the dynamics parameters $\boldsymbol{\mu}$ to the mean $\mathbf{m}_E(\boldsymbol{\mu})$ and standard deviation $\boldsymbol{\Sigma}_E(\boldsymbol{\mu})$ of the encoder distribution. The policy network $\pi(\mathbf{a}|\mathbf{s},\mathbf{g},\mathbf{z})$ receives as input the state $\mathbf{s}$, goal $\mathbf{g}$, and dynamics encoding $\mathbf{z}$, then outputs the mean $\mathbf{m}_\pi(\mathbf{s},\mathbf{g},\mathbf{z})$ of a Gaussian action distribution. The standard deviation $\boldsymbol{\Sigma}_\pi = \text{diag}(\sigma_\pi^1, \sigma_\pi^2, ...)$ of the action distribution is represented by a fixed matrix. The value function $V(\mathbf{s},\mathbf{g},\boldsymbol{\mu})$ receives as input the state, goal, and dynamics parameters.

B. Learned Skills

Our framework is able to learn a diverse set of locomotion skills for the Laikago, including dynamic gaits, such as pacing and trotting, as well as agile turning and spinning motions (Figure 4). Pacing is typically used for walking at slower speeds, and is characterized by each pair of legs on the same side of the body moving in unison (Figure 4(a)) [50]. Trotting is a faster gait, where diagonal pairs of legs move together (Figure 1). We are able to train policies for these different gaits just by providing the system with different reference motions.

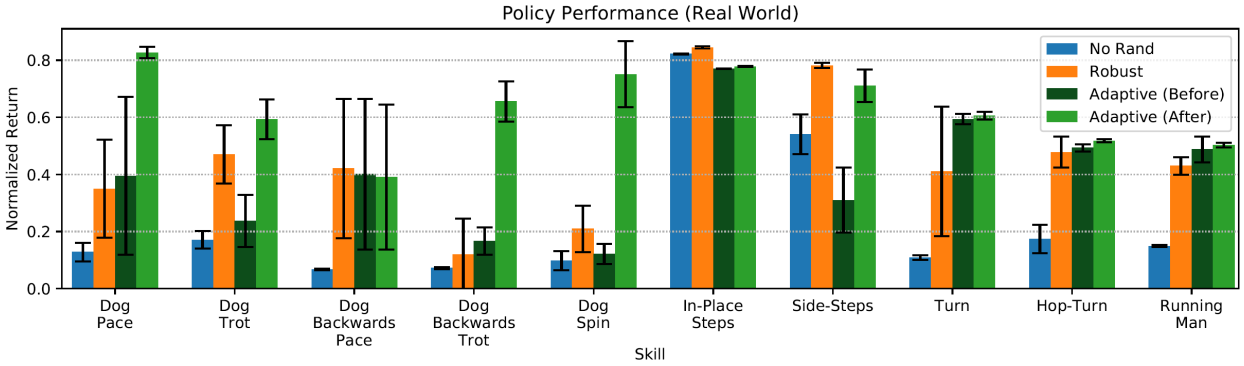

Fig. 5. Performance statistics of imitating various skills in the real world. Performance is recorded as the average normalized return between [0, 1]. Three policies initialized with different random seeds are trained for each combination of skill and method. The performance of each policy is evaluated over 5 episodes, for a total of 15 trials per method. The adaptive policies outperform the non-adaptive policies on most skills.

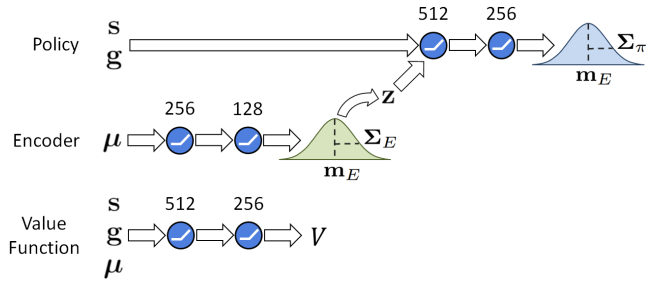

Fig. 6. Schematic illustration of the network architecture used for the adaptive policy. The encoder $E(\mathbf{z}|\boldsymbol{\mu})$ receives the dynamics parameters $\boldsymbol{\mu}$ as input, which are processed by two fully-connected layers with 256 and 128 ReLU units, and then mapped to a Gaussian distribution over the latent space $\mathbf{Z}$ with mean $\mathbf{m}_E(\boldsymbol{\mu})$ and standard deviation $\boldsymbol{\Sigma}_E(\boldsymbol{\mu})$. An encoding $\mathbf{z}$ is sampled from the encoder distribution and provided to the policy $\pi(\mathbf{a}|\mathbf{s}, \mathbf{g}, \boldsymbol{\mu})$ as input, along with the state $\mathbf{s}$ and goal $\mathbf{g}$. The policy is modeled with two layers of 512 and 256 units, followed by an output layer which specifies the mean $\mathbf{m}_\pi(\mathbf{s}, \mathbf{g}, \mathbf{z})$ of the action distribution. The standard deviation $\boldsymbol{\Sigma}_\pi$ of the action distribution is specified by a fixed diagonal matrix. The value function $V(\mathbf{s}, \mathbf{g}, \boldsymbol{\mu})$ is modeled by a separate network with 512 and 256 hidden units.

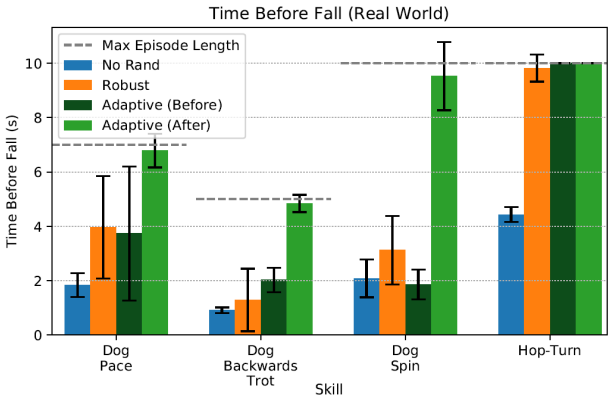

Fig. 7. Comparison of the time elapsed before the robot falls when deploying various policies in the real world. The adaptive policies are often able to maintain balance longer than the other baselines policies, and tend to reach the max episode length without falling.

Furthermore, by simply playing the mocap clips backwards, we are able to train policies for different backwards walking gaits (Figure 4(b)). The gaits learned by our policies are faster than those of the manually-designed controller from the manufacturer. The fastest manufacturer gait reaches a top speed of about 0.84m/s, while the Dog Trot policy reaches a speed of 1.08m/s. The backwards trotting gait reaches an even higher speed of 1.20m/s. In addition to imitating mocap data from animals, our system is also able to learn from artist animated motions. While these hand-animated motions are generally not physically correct, the policies are nonetheless able to closely imitate most motions with the real robot. This includes a highly dynamic Hop-Turn motion, in which the robot performs a 90 degrees turn midair (Figure 4(e)). While our system is able to imitate a variety of motions, some motions, such as Running Man (Figure 4(f)), prove challenging to reproduce. The motion requires the robot to travel backwards while moving in a forward-walking manner. Our policies learn to keep the robot's feet on the ground and shuffle backwards, instead of lifting the feet during each step.

C. Domain Adaptation

To determine the effects of domain adaptation, we compare our method to non-adaptive policies trained in simulation without randomization (No Rand), and robust policies trained with randomization (Robust) but do not perform adaptation in new environments. Real-world performance comparisons of these methods are shown in Figure 5, detailed performance statistics in simulation and the real world are available in Appendix B. When deployed on the real robot, the adaptive policies outperform their non-adaptive counterparts on most skills. For simpler skills, such as In-Place Steps and Side-Steps, the robust policies are sufficient for transfer to the real robot. But for more dynamic skills, such as Dog Pace and Dog Spin, the robust policies are prone to falling, while the adaptive policies can execute the skills more consistently. Policies trained without randomization fail to transfer to the real world for most skills. Figure 7 compares the time elapsed before the robot falls under the various policies. The adaptive policies are often able to maintain balance for a longer period of time than the other methods, with a significant performance improvement after adaptation.

To evaluate the policies' abilities to cope with unfamiliar dynamics, we test the policies in out-of-distribution simulated environments, where the dynamics parameters are sampled from a larger range of values than those used during training. The range of values used during training and testing are detailed in Table I. Figure 8 visualizes the performance of the policies in 100 simulated environments with different dynamics. The vertical axis represents the normalized return, and the

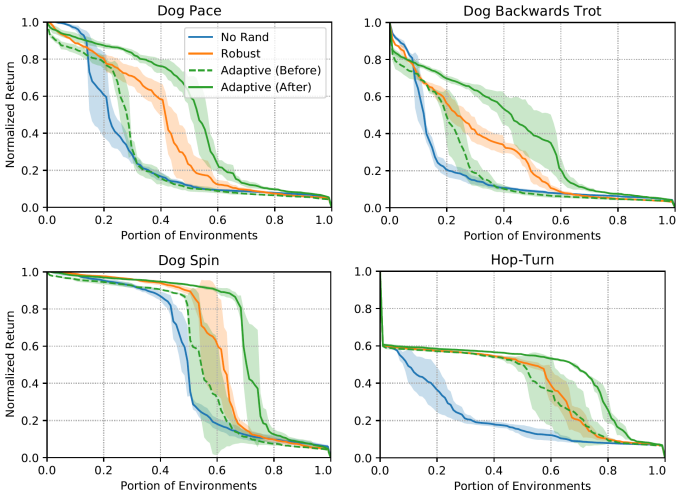

Fig. 8. Performance of policies in 100 simulated environments with different dynamics. The vertical axis represents the normalized return, and the horizontal axis records the portion of environments in which a policy achieves a return higher than a particular value. The adaptive policies achieve higher returns under more diverse dynamics than the non-adaptive policies.

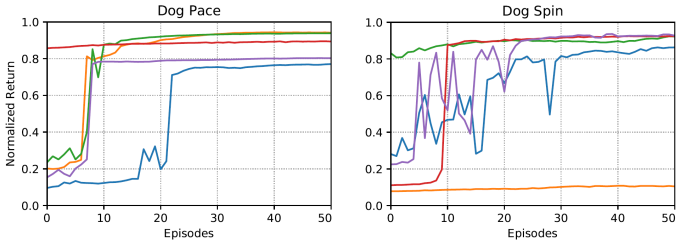

Fig. 9. Learning curves of adapting policies to different simulated environments using the learned latent space. The policies are able to adapt to new environments in a relatively small number of episodes.

horizontal axis records the portion of environments in which a policy achieves a return higher than a particular value. For example, in the case of Dog Pace, the adaptive policies achieve a return higher than 0.6 in 50% of the environments, while the robust policy achieves a return higher than 0.6 in 38% of the environments. The experiments are repeated 3 times for each method using policies initialized with different random seeds. In these experiments, the adaptive policies tend to outperform their non-adaptive counterparts across the various skills. This suggests that the adaptation process is able to better generalize to environments that differ from those encountered during training. To analyze the performance of policies during the adaptation process, we record the performance of individual policies after each update iteration. Figure 9 illustrates the learning curves in 5 different environments for each skill. The policies are generally able to adapt to new environments in a relatively few number of episodes.

D. Information Bottleneck

Next we evaluate the effects of the information bottleneck on adaptation performance. Figure 10 summarizes the performance of policies trained with different values of β for the information penalty. Larger values of β produce policies that access fewer number of bits of information from the dynamics parameters during pre-training. This encourages a policy to be

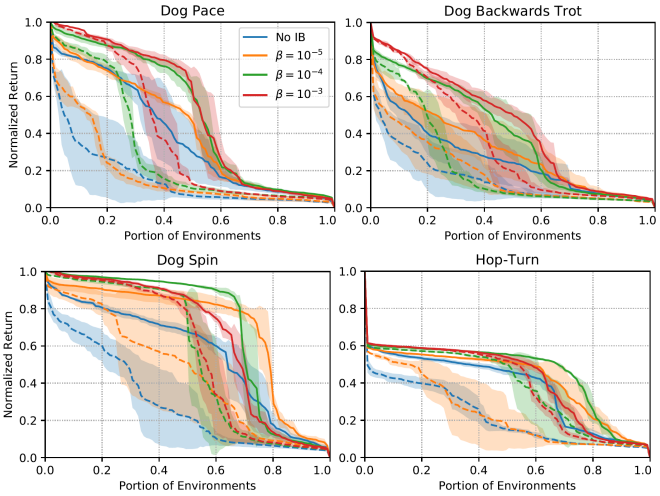

Fig. 10. Performance of adaptive policies trained with different coefficients β for the information penalty. "No IB" corresponds to policies trained without an information bottleneck. The dotted lines represent performance before adaptation, and the solid lines represent after adaptation.

less reliant on precise knowledge of the underlying dynamics, which in turn results in more robust behaviors that attain higher performance before adaptation. However, since the policy's behavior is less dependent on the latent variables, this can also result in less adaptable policies, which exhibit smaller performance improvements after adaptation. Similarly, smaller values of β tend to produce less robust but more adaptive policies, exhibiting lower performance before adaptation, but a larger improvement after adaptation. In our experiments, we find that $\beta = 10^{-4}$ provides a good trade-off between robustness and adaptability. We also compare the information-constrained latent representations to the unconstrained counterparts (No IB). The information-constrained policies generally achieve better performance both before and after adaptation.

VIII. Discussion and Future Work

We presented a framework for learning agile legged-locomotion skills by imitating reference motion data. By simply providing the system with different reference motions, we are able to learn policies for a diverse set of behaviors with a quadruped robot, which can then be efficiently transferred from simulation to the real world. However, due to hardware and algorithmic limitations, we have not been able to learn more dynamic behaviors such as large jumps and runs. Exploring techniques that are able to reproduce these behaviors in the real world could significantly increase the agility of legged robots. The behaviors learned by our policies are currently not as stable as the best manually-designed controllers. Improving the robustness of these learned controllers would be valuable for more complex real-world applications. We are also interested in learning from other sources of motion data, such video clips, which could substantially increase the volume of behavioral data that robots can learn from.

Acknowledgements

We would like to thank Julian Ibarz, Byron David, Thinh Nguyen, Gus Kouretas, Krista Reymann, Bonny Ho, and the Google Robotics team for their contributions to this work.

References

[1] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *CoRR*, abs/1612.00410, 2016. URL http://arxiv.org/abs/1612.00410.

[2] Taylor Apgar, Patrick Clary, Kevin Green, Alan Fern, and Jonathan Hurst. Fast online trajectory optimization for the bipedal robot cassie. 06 2018. doi: 10.15607/RSS.2018.XIV.054.

[3] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252, 2018.

[4] Stephen Butterworth et al. On the theory of filter amplifiers. *Wireless Engineer*, 7(6):536–541, 1930.

[5] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan D. Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *CoRR*, abs/1810.05687, 2018. URL http://arxiv.org/abs/1810.05687.

[6] Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyztsoC5Y7.

[7] Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. Robust task-based control policies for physics-based characters. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 28(5):Article 170, 2009.

[8] Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. Generalized biped walking control. *ACM Transctions on Graphics*, 29(4):Article 130, 2010.

[9] Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel van de Panne. Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics*, 30(4):Article TBD, 2011.

[10] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[11] Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. Feature-Based Locomotion Controllers. *ACM Transactions on Graphics*, 29(3), 2010.

[12] Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[13] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl$ˆ2$: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016. URL http://arxiv.org/abs/1611.02779.

[14] Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning cpg sensory feedback with policy gradient for biped locomotion for a full-body humanoid. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, page 1267–1273. AAAI Press, 2005. ISBN 157735236x.

[15] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 0387743146.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/finn17a.html.

[17] Christian Gehring, Stelian Coros, Marco Hutler, Dario Bellicoso, Huub Heijnen, Remo Diethelm, Michael Bloesch, Péter Fankhauser, Jemin Hwangbo, Mark Hoepflinger, and Roland Siegwart. Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot. *IEEE Robotics & Automation Magazine*, pages 1–1, 02 2016. doi: 10.1109/MRA.2015.2505910.

[18] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Positive force feedback in bouncing gaits? *Proceedings. Biological sciences / The Royal Society*, 270:2173–83, 11 2003. doi: 10.1098/rspb.2003.2454.

[19] Michael Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 33–42, New York, NY, USA, 1998. ACM. ISBN 0-89791-999-8. doi: 10.1145/280814.280820. URL http://doi.acm.org/10.1145/280814.280820.

[20] A. Goswami. Foot rotation indicator (fri) point: a new gait planning tool to evaluate postural stability of biped robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 1, pages 47–52 vol.1, May 1999. doi: 10.1109/ROBOT.1999.769929.

[21] David B. Grimes, Rawichote Chalodhorn, and Rajesh P. N. Rao. Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In Gaurav S. Sukhatme, Stefan Schaal, Wolfram Burgard, and Dieter Fox, editors, *Robotics: Science and Systems*. The MIT Press, 2006. ISBN 0-262-69348-8. URL http://dblp.uni-trier.de/db/conf/rss/rss2006.html#GrimesCR06.

[22] Tuomas Haarnoja, Aurick Zhou, Sehoon Ha, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *CoRR*, abs/1812.11103, 2018. URL http://arxiv.org/abs/1812.11103.

[23] Josiah Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, February 2017.

[24] Zhanpeng He, Ryan Julian, Eric Heiden, Hejia Zhang, Stefan Schaal, Joseph J. Lim, Gaurav S. Sukhatme, and Karol Hausman. Zero-shot skill composition and simulation-to-real transfer by learning task representations. *CoRR*, abs/1810.02422, 2018. URL http://arxiv.org/abs/1810.02422.

[25] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017. URL http://arxiv.org/abs/1707.02286.

[26] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi: 10.1126/scirobotics.aau5872. URL https://robotics.sciencemag.org/content/4/26/eaau5872.

[27] S. Kim, C. Kim, B. You, and S. Oh. Stable whole-body motion generation for humanoid robots to imitate human motions. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2518–2524, Oct 2009. doi: 10.1109/IROS.2009.5354271.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL http://dblp.uni-trier.de/db/journals/corr/corr1312.html#KingmaW13.

[30] Jonas Koenemann, Felix Burget, and Maren Bennewitz. Real-time imitation of human whole-body motions by humanoids. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2812, 2014.

[31] Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *ICRA*, pages 2619–2624. IEEE, 2004. URL http://dblp.uni-trier.de/db/conf/icra/icra2004-3.html#KohlS04.

[32] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. Scalable muscle-actuated human simulation and control. *ACM Trans. Graph.*, 38(4), July 2019. ISSN 0730-0301. doi: 10.1145/3306346.3322972. URL https://doi.org/10.1145/3306346.3322972.

[33] Yoonsang Lee, Sungeun Kim, and Jehee Lee. Data-driven biped control. *ACM Trans. Graph.*, 29(4), July 2010. ISSN 0730-0301. doi: 10.1145/1778765.1781155. URL https://doi.org/10.1145/1778765.1781155.

[34] Libin Liu and Jessica Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Trans. Graph.*, 37(4), July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201315. URL https://doi.org/10.1145/3197517.3201315.

[35] Libin Liu, Michiel van de Panne, and KangKang Yin. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics*, 35(3), 2016.

[36] Kendall Lowrey, Svetoslav Kolev, Jeremy Dao, Aravind Rajeswaran, and Emanuel Todorov. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system. *CoRR*, abs/1803.10371, 2018. URL http://arxiv.org/abs/1803.10371.

[37] Hirofumi Miura and Isao Shimoyama. Dynamic walk of a biped. *The International Journal of Robotics Research*, 3:60 – 74, 1984.

[38] Uldarico Muico, Yongjoon Lee, Jovan Popoviundefined, and Zoran Popoviundefined. Contact-aware nonlinear control of dynamic characters. *ACM Trans. Graph.*, 28(3), July 2009. ISSN 0730-0301. doi: 10.1145/1531326.1531387. URL https://doi.org/10.1145/1531326.1531387.

[39] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 3905–3910 vol.3, Sep. 2003. doi: 10.1109/ROBOT.2003.1242196.

[40] Gerhard Neumann and Jan R. Peters. Fitted q-iteration by advantage weighted regression. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1177–1184. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3501-fitted-q-iteration-by-advantage-weighted-regression.pdf.

[41] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018. URL http://arxiv.org/abs/1808.00177.

[42] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, May 2018. doi: 10.1109/ICRA.2018.8460528.

[43] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Trans. Graph.*, 35(4):81:1–81:12, July 2016. ISSN 0730-0301. doi: 10.1145/2897824.2925881. URL http://doi.acm.org/10.1145/2897824.2925881.

[44] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL

http://doi.acm.org/10.1145/3197517.3201311.

[45] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph.*, 37 (6), November 2018.

[46] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *CoRR*, abs/1910.00177, 2019. URL https://arxiv.org/abs/1910.00177.

[47] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2817–2826, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/pinto17a.html.

[48] Nancy Pollard, Jessica K. Hodgins, M.J. Riley, and Chris Atkeson. Adapting human motion for the control of a humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, May 2002.

[49] M. H. Raibert. Hopping in legged systems — modeling and simulation for the two-dimensional one-legged case. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-14(3):451–463, May 1984. ISSN 2168-2909. doi: 10.1109/TSMC.1984.6313238.

[50] Marc H Raibert. Trotting, pacing and bounding by a quadruped robot. *Journal of biomechanics*, 23:79–98, 1990.

[51] Andrei A. Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 262–270. PMLR, 13–15 Nov 2017. URL http://proceedings.mlr.press/v78/rusu17a.html.

[52] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *CoRR*, abs/1611.04201, 2016. URL http://arxiv.org/abs/1611.04201.

[53] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[54] William J. Schwind and Daniel E. Koditschek. Spring loaded inverted pendulum running: a plant model. 1998.

[55] W. Suleiman, E. Yoshida, F. Kanehiro, J. Laumond, and A. Monin. On human motion imitation by humanoid robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 2697–2704, May 2008. doi: 10.1109/ROBOT.2008.4543619.

[56] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

[57] J. Tan, Z. Xie, B. Boots, and C. K. Liu. Simulation-based design of dynamic controllers for humanoid balancing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2729–2736, Oct 2016. doi: 10.1109/IROS.2016.7759424.

[58] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.010.

[59] Russ Tedrake, Teresa Weirui Zhang, and H. Sebastian Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, volume 3, pages 2849–2854, Piscataway, NJ, USA, 2004. IEEE. ISBN 0-7803-8463-6. URL http://www.cs.cmu.edu/~cga/legs/01389841.pdf.

[60] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

[61] Xingxing Wang. Laikago Pro, Unitree Robotics, 2018. URL http://www.unitree.cc/e/action/ShowInfo.php?classid=6&id=355.

[62] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan Hurst, and Michiel van de Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Proc. Conference on Robot Learning (CORL 2019)*, 2019.

[63] K. Yamane, S. O. Anderson, and J. K. Hodgins. Controlling humanoid robots with human motion data: Experimental validation. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 504–510, Dec 2010. doi: 10.1109/ICHR.2010.5686312.

[64] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. *ACM Trans. Graph.*, 26(3):Article 105, 2007.

[65] Wenhao Yu, Visak C. V. Kumar, Greg Turk, and C. Karen Liu. Sim-to-real transfer for biped locomotion. *CoRR*, abs/1903.01390, 2019. URL http://arxiv.org/abs/1903.01390.

[66] Wenhao Yu, C. Karen Liu, and Greg Turk. Policy transfer with strategy optimization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1g6osRcFQ.

[67] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization, 2019.

[68] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph.*, 37(4):145:1–145:11,

July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201366. URL http://doi.acm.org/10.1145/3197517.3201366.

APPENDIX

A. Hyperparameters

Table II summarizes the hyper-parameter settings for training with proximal-policy optimization (PPO) in simulation, and Table III shows the hyper-parameters for domain adaptation with advantage-weighted regression (AWR). Gradient descent descent updates are performed using Adam [28].

Parameter	Value
Discount factor γ	0.95
Policy Adam learning rate	2×10^{-5}
Value function Adam learning rate	10^{-5}
PPO clip threshold	0.2
PPO batch size	10000
PPO epochs	10
Information penalty coefficient (β)	10^{-4}

TABLE II
HYPER-PARAMETERS USED DURING TRAINING IN SIMULATION WITH PPO.

Parameter	Value
Discount factor γ	1.0
Adam learning rate	5×10^{-3}
Gradient steps per iteration	10
AWR temperature α	0.01

TABLE III
HYPER-PARAMETERS USED FOR DOMAIN ADAPTATION WITH AWR IN THE REAL WORLD.

B. Performance Statistics

Table V summarizes the performance of the different policies when deployed in the real world, and Table IV summarizes performance in simulation. Performance is recorded as the average normalized return, with 0 corresponding to the minimum possible return per episode and 1 being the maximum return. Note that the maximum return may not be achievable, since the reference motions are generally not physically feasible for the robot. Performance is calculated using the average of 3 policies initialized with different random seeds. Performance is evaluated in simulation using the canonical dynamics parameters. Most policies achieve a similar performance in simulation. But when deployed on the real robot, the adaptive policies outperform the non-adaptive policies on most skills. For simpler skills, such as the In-Place Steps and Side-Steps, the robust policy is sufficient for transfer to the real world. But for more dynamic motions, such as Dog Pace and Dog Trot, the robust policy is prone to falling, while the adaptive policies are able to execute the skills more consistently in the real world. Figure 11 compares the time elapsed before the robot falls under the various policies. Falls are detected when the robot's torso makes contact with the ground. The adaptive policies are often able to maintain balance for a longer period of time than the other methods, with a sizable performance improvement after adaptation. The adaptive policies are often able to reach the maximum episode length without falling.

Skill	No Rand	Robust	Adaptive (Before)	Adaptive (After)
Dog Pace	0.128 ± 0.033	0.350 ± 0.172	0.395 ± 0.277	$\mathbf{0.827 \pm 0.020}$
Dog Trot	0.171 ± 0.031	0.471 ± 0.102	0.237 ± 0.092	$\mathbf{0.593 \pm 0.070}$
Dog Backwards Pace	0.067 ± 0.003	$\mathbf{0.421 \pm 0.244}$	0.401 ± 0.264	0.390 ± 0.254
Dog Backwards Trot	0.072 ± 0.004	0.120 ± 0.126	0.167 ± 0.048	$\mathbf{0.656 \pm 0.071}$
Dog Spin	0.098 ± 0.033	0.209 ± 0.081	0.121 ± 0.035	$\mathbf{0.751 \pm 0.116}$
In-Place Steps	0.822 ± 0.002	$\mathbf{0.845 \pm 0.004}$	0.771 ± 0.001	0.778 ± 0.002
Side-Steps	0.541 ± 0.070	$\mathbf{0.782 \pm 0.009}$	0.310 ± 0.114	0.710 ± 0.057
Turn	0.108 ± 0.008	0.410 ± 0.227	0.594 ± 0.018	$\mathbf{0.606 \pm 0.014}$
Hop-Turn	0.174 ± 0.050	0.478 ± 0.054	0.493 ± 0.012	$\mathbf{0.518 \pm 0.005}$
Running Man	0.149 ± 0.004	0.430 ± 0.031	0.488 ± 0.045	$\mathbf{0.503 \pm 0.008}$

TABLE IV
PERFORMANCE STATISTICS OF IMITATING VARIOUS SKILLS IN THE REAL WORLD. PERFORMANCE IS RECORDED AS THE AVERAGE NORMALIZED RETURN BETWEEN [0, 1]. THREE POLICIES INITIALIZED WITH DIFFERENT RANDOM SEEDS ARE TRAINED FOR EACH COMBINATION OF SKILL AND METHOD. THE PERFORMANCE OF EACH POLICY IS EVALUATED OVER 5 EPISODES, FOR A TOTAL OF 15 TRIALS PER METHOD. THE METHOD THAT ACHIEVES THE HIGHEST RETURN FOR EACH SKILL ON THE REAL ROBOT IS HIGHLIGHTED.

Skill	No Rand	Robust	Adaptive (Ours)
Dog Pace	$\mathbf{0.839 \pm 0.002}$	0.820 ± 0.001	0.812 ± 0.004
Dog Trot	$\mathbf{0.752 \pm 0.002}$	0.727 ± 0.002	0.718 ± 0.001
Dog Backwards Pace	$\mathbf{0.843 \pm 0.001}$	0.828 ± 0.001	0.816 ± 0.002
Dog Backwards Trot	$\mathbf{0.768 \pm 0.002}$	0.734 ± 0.001	0.715 ± 0.001
Dog Spin	$\mathbf{0.859 \pm 0.001}$	0.839 ± 0.001	0.839 ± 0.001
In-Place Steps	$\mathbf{0.945 \pm 0.002}$	0.938 ± 0.001	0.935 ± 0.001
Side-Steps	$\mathbf{0.846 \pm 0.001}$	0.808 ± 0.006	0.820 ± 0.002
Turn	$\mathbf{0.715 \pm 0.001}$	0.666 ± 0.009	0.675 ± 0.004
Hop-Turn	$\mathbf{0.628 \pm 0.001}$	0.606 ± 0.005	0.597 ± 0.001
Running Man	$\mathbf{0.585 \pm 0.003}$	0.557 ± 0.004	0.544 ± 0.002

TABLE V

PERFORMANCE STATISTICS OF IMITATING VARIOUS SKILLS IN SIMULATION USING THE CANONICAL DYNAMICS PARAMETERS. PERFORMANCE IN SIMULATION IS SIMILAR ACROSS THE DIFFERENT METHODS.

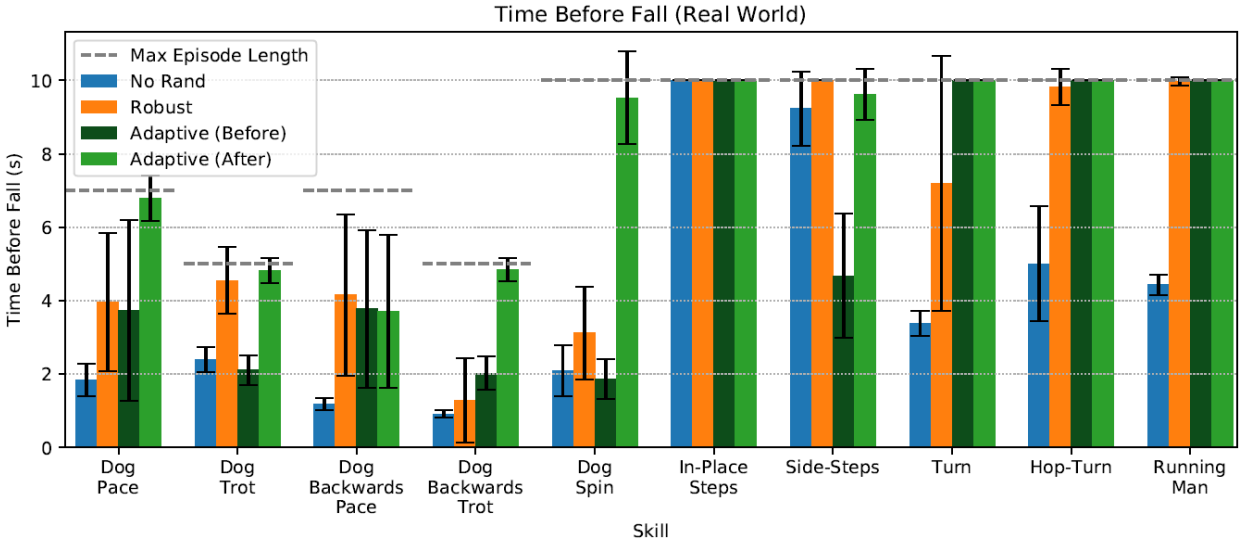

Fig. 11. Comparison of the time elapsed before the robot falls when deploying various policies in the real world. The adaptive policies are generally able to maintain balance longer than the other baselines policies.