Error-Aware Policy Learning: Zero-Shot Generalization in Partially Observable Dynamic Environments

Visak Kumar
School of Interactive Computing
Georgia Institute of Technology
Email: visak3@gatech.edu

Sehoon Ha
School of Interactive Computing
Georgia Institute of Technology
Email: sehoonha@gatech.edu

C. Karen Liu
Computer Science Department
Stanford University
Email: karenliu@cs.stanford.edu

Abstract—Simulation provides a safe and efficient way to generate useful data for learning complex robotic tasks. However, matching simulation and real-world dynamics can be quite challenging, especially for systems that have a large number of unobserved or unmeasurable parameters, which may lie in the robot dynamics itself or in the environment with which the robot interacts. We introduce a novel approach to tackle such a sim-to-real problem by developing policies capable of adapting to new environments, in a zero-shot manner. Key to our approach is an error-aware policy (EAP) that is explicitly made aware of the effect of unobservable factors during training. An EAP takes as input the predicted future state error in the target environment, which is provided by an error-prediction function, simultaneously trained with the EAP. We validate our approach on an assistive walking device trained to help the human user recover from external pushes. We show that a trained EAP for a hip-torque assistive device can be transferred to different human agents with unseen biomechanical characteristics. In addition, we show that our method can be applied to other standard RL control tasks.

I. INTRODUCTION

Simulation has a growing role in learning-based methods to design control policies for robots as it provides a safe and efficient way to generate useful data. However, for robotic agents that are governed by complex dynamics or interacting with a complex environment, it is challenging to identify a model that captures the real-world dynamics accurately, giving rise to the so called sim-to-real problem in transferring control policies. Among many factors responsible for the sim-to-real gap, we are interested in addressing the challenges involving unobserved or unmeasurable model parameters, which may lie in the robot dynamics itself or in the environment with which the robot interacts. For example, for assistive robots that aid in locomotion, the dynamics of the wearable robot is closely coupled with humans, who show remarkably large variations in their movements, such as variations in muscle activation dynamics, muscle maximum force, fatigue, robot-human interaction dynamics, delay in sensing/actuation, all of which are difficult to measure and parameterize accurately. Similarly, other types of robots, such as quadrupeds and bipeds, also require accurate complex contact dynamics, contact parameters, motor dynamics, delay in the system, difficult to measure or model correctly.

Two broad approaches have been proposed to address the sim-to-real issue: 1) Domain randomization and domain adaptation methods [22, 33, 27, 23, 28] which aim to learn robust or universal policies by training them with variations in the model parameters. These methods require manual engineering of the range in which the parameters are varied. For highly complex systems and environments, domain randomization or adaptation often determines a subset of parameters to be observable and leaves the unpredictable effect of unobsered parameters to chance. 2) System identification methods [35, 32, 14, 4], aim to identify accurate models of the real robot to bring simulation closer to reality. System identification can be interleaved with policy learning—deploying the current policy in the target environment to collect more data to further improve the dynamic model [39, 26, 2]. Since the task-relevant training data is difficult to acquire from the real world, system identification also needs to determine a subset of parameters to be observable to avoid over fitting. Neither approach has demonstrated reliable ability to transfer policies to the real world when the combined dynamics of agent and environment is highly unobservable.

In this work, we introduce a novel approach to tackle sim-to-real problems in which the environment dynamics has high variance and is highly unobservable. While our approach is motivated by physical assistive robotic applications, the method can be applied to other tasks in which many dynamic parameters are challenging to model. We propose to train a policy explicitly aware of the effect of unobservable factors during training, called an Error-Aware policy (EAP). Akin to the high-level idea of meta learning, we divide the dynamical environments into training and validation sets and "emulate" a reality gap in simulation. Instead of estimating the model parameters that give rise to the emulated reality gap, we train a function that predicts the deviation (i.e. error) of future states due to the emulated reality gaps. Conditioned on the error predictions, the error-aware policies (EAPs) can learn to overcome the reality gap, in addition to mastering the task.

The main application in this work is to learn an error-

aware policy for assistive device control, such as a hip-exoskeleton that helps the user to recover balance during locomotion. From biomechanical data of human gait, we model multiple virtual human walking agents, each varying in physical characteristics as well as parameters that affect the dynamics such as joint damping, torque limits, ground friction, and sensing and actuation delay. We then train a single policy on this group of human agents and show that that the learned EAP works effectively when tested on a different human agent without needing additional data. We extend the prior work, [18], that trained a control policy for push-recovery assistive device for just one simulated human agent, and develop an algorithm that enables the learned policy to transfer to other human agents with unseen biomechanical characteristics.

We evaluate our approach on assistive wearable device by quantifying the stability and gait characteristics generated by an unseen human agent wearing the device with the trained EAP. We present a comprehensive study of the benefits of our approach over prior zero-shot methods such as universal policy (UP) and domain randomization (DR). We also provide results on some standard RL environments, such as Cartpole, Hopper, 2D walker and a quadrupedal robot.

II. Related work

A. Transfer of RL policies

A popular approach to transfer control policies is Domain randomization (DR). DR methods [22, 33, 27, 23, 28] propose to train policies that are robust to variations in the parameters that affect the system dynamics. Although some of these methods have been validated in the real world [22, 23], DR often requires manual engineering of the range in which the parameters are varied to make sure that the true system model lies within the range of variation. For a complex robotic system, it is often challenging to estimate the correct range of all the parameters because a large range of variation could lead to lower task performance, whereas a smaller range leads to less robust policies. To address the demanding sample budget issue with domain randomization, [21] presented a data-efficient domain randomization algorithm based on bayesian optimization. The algorithm presented in Mehta et al [20] actively adapts the randomization range of variation to alleviate the need for exhaustive manual engineering. Ramos et al [29] proposed an approach to infer the distribution of the dynamical parameters and showed that policies trained with randomization within this distribution can transfer better.

Careful identification of parameters using data from the real world, popularly known as system identification, has also shown promising results in real-world robots. Tan et al [32] and Hwangbo et al [14] carefully identified the actuator dynamics to bring the source environment closer to the target, Xie et al [35] also demonstrated that careful system identification techniques can transfer biped locomotion policies from simulation to real-world. Jegorova

et al [16] presented a technique that improves on existing system identification techniques by borrowing ideas from generative adversarial networks (GAN) and showed improved ability to identify the parameters of a system. Similarly, Jiang et al [17] presented a SimGAN algorithm that identifies a hybrid physics simulator to match the simulated trajectories to the ones from the target domain to enable policy adaptation. Yu et al [38] developed a method that combines online system identification and universal policy to enable identifying dynamical parameters in an online fashion. Citing the difficulty in obtaining meaningful data for system identification, [43] developed an algorithm that probes the target environment to provide more information about the dynamics of the environment. A few model based approaches have also been successful in transferring policies to a target domain [31, 6, 37].

Another popular approach of transferring policies includes utilizing data from the target domain to improve the policy. Chebotar et al [4] presented a method that interleaves policy learning and system identification, however this requires deploying the policy in the target domain every few iterations. This method would be impractical for a system that interacts closely with a human because of safety concerns. Yu et al [39] and Peng et al [25] presented latent space adaptation techniques where the policy is adapted in the target domain by searching for a latent space input to the policy that enables successful transfer. Exarchos et al [7] also presented an algorithm that achieved policy transfer using only kinematic domain randomization combined with policy adaptation in the target domain, similar to [39].

Yu et al [40] proposed Meta Strategy Optimization, a meta-learning algorithm for training policies with latent variables that can quickly adapt to new scenarios with a handful of trials in the target environment. Among the methods that use data from the target domain also include meta-learning approaches like Bhelkale et al [2], in which a model-based meta-reinforcement learning algorithm was presented to account for changing dynamics of an aerial vehicle carrying different payloads. In this approach, the parameters causing the variations in the dynamics are inferred by deploying the policy in the target domain, which in turn helps improve the policy's performance. In Ignasi et al. [5], the idea of model-agnostic meta-learning [9] was extended to modelling dynamics of a robot. The authors presented an approach to quickly adapt the model of the robot in a new test environment while using a sampling-based controller MPPI to compute the actions. [36] developed a zero-shot transfer for policy by combining reinforcement learning and a robust tracking controller with a disturbance observer in the target environment. The validated the approach on a vehicle driving task. Similarly, [8, 11] presented an approach to combine bayesian learning and adaptive control by learning model error and uncertainty.

For tasks such as assistive device control for human locomotion, it is potentially unsafe and prohibitive to collect sufficient task-relevant data in the real world which pre-

vents us from using methods such as system identification or transfer learning approaches that need data in the target environment. In addition to this, human dynamics exhibit large variations due to many unobserved parameters, this makes it challenging to define the right parameters for the system model in simulation and also in finding the right range of parameter variation for an approach like DR.

B. Adaptation for Assistive Devices

Assistive devices such as exoskeletons provide unique challenges for domain adaptation due to the large variations between individuals who pilot the device. Zhang et al [42] reported a human-in-the-loop optimization approach for ankle exoskeletons to account for this variability, however, this approach takes a few hours per individual to find the optimal control law. Jackson et al [15] presented a unique heuristic-based approach to design a control law that adapts to the person's muscle activity. While these methods work well for steady-state walking, the large number of data required to optimize for in the case of [42] and the complex muscle responses involved during push recovery make it an infeasible application. Several recent works have incorporated a learning-based approach to tackling the problem of adaptation, Peng et al [26] adopted a reinforcement learning approach to learn assistive walking strategies for Hemiplegic patients, which was tested on real human patients and showed robustness and adaptability. However, it requires online data to update the actor-critic network. This process involves deploying a policy on a patient to collect data, for a task like push recovery it might be challenging to collect relevant data required for updating the policy without compromising the patient's safety. Both [13] and [41] combined dynamic motion primitives (DMPs) and learning approaches to adapt control strategies for different individuals. Majority of the work with assistive devices have primarily focused on walk assistance and not on push-recovery like this paper addresses.

III. ZERO-SHOT TRANSFER WITH ERROR-AWARE POLICY

We present a method to achieve the zero-shot transfer of control policies in partially observable dynamical environments. We consider robotic systems and environments with unobservable or unmeasurable model parameters, which make building accurate simulation models difficult.

We present a novel policy architecture, an Error-Aware Policy (EAP), that is explicitly aware of errors induced by unobservable dynamics parameters and self-corrects its actions according to the errors. An EAP takes the current state, observable dynamic parameters, and predicted errors as inputs and generates corrected actions. We learn an additional error-prediction function that outputs the expected error. Both the error-aware policy and the error-prediction function, are iteratively learned using model-free reinforcement learning and supervised learning.
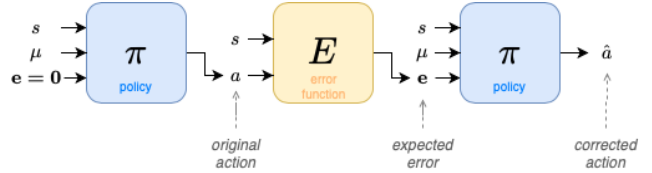

Figure 1: Overview of An Error-aware Policy (EAP). An EAP takes the "expected" future state error as an additional input. The expected error is predicted based on the current state $\mathbf{s}$, observable parameters $\boldsymbol{\mu}$, and an uncorrected action $\mathbf{a}$ that assumes zero error.

A. Problem Formulation

We formulate the problem as Partially Observable Markov Decision Processes (PoMDPs), $(S, O, A, P, R, \rho_0, \gamma)$, where S is the state space, O is the observation space, A is the action space, P is the transition function, R is the reward function, ρ_0 is the initial state distribution and γ is a discount factor. In our formulation, we make a clear distinction between observable model parameters $\boldsymbol{\mu}$ and unobservable parameters $\boldsymbol{\nu}$ of the agent and environment. Observable quantities are parameters that can be easily measured such as masses or link lengths, whereas unobserved quantities are challenging to estimate, such as circuit dynamics or backlash. Therefore, both $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ affect the transition function $P(\mathbf{s}'|\mathbf{a}, \mathbf{s}, \boldsymbol{\mu}, \boldsymbol{\nu})$. Since we can configure our simulator with both $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, we can randomly sample $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ and create a list of K different environments $\mathbf{D} = \{(\boldsymbol{\mu}_0, \boldsymbol{\nu}_0), (\boldsymbol{\mu}_1, \boldsymbol{\nu}_1), \cdots, (\boldsymbol{\mu}_K, \boldsymbol{\nu}_K)\}$, but it is hard to obtain $\boldsymbol{\nu}$ at testing time. In this case, the transition function will be abbreviated as $P(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \boldsymbol{\mu})$.

Instead of estimating the values of unobserved quantities, we capture the effect of these parameters by defining a metric called a *state-error*. When transferring from one environment to another, the action $\mathbf{a}$ applied at a given state $\mathbf{s}$ will produce different next states due to the differences in both $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, in other words, a state-error.

We hypothesize that a policy which is explicitly aware of the state-error would be able to make better decisions by self-correcting its action. We call this an error-aware policy $\pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\mu}, \mathbf{e})$ (EAP), which takes in observable parameters $\boldsymbol{\mu}$ as well as the "expected" future state error in a new environment $\mathbf{e}$ as input (Figure 1).

We present a novel training methodology using model-free reinforcement learning that involves learning two functions: an error-aware policy and an error prediction function. First, we learn an error-aware policy that takes the output of error prediction function E as an input and has the ability to generalize to novel environments in a zero-shot manner. Simultaneously, we learn an error-prediction function, which takes as inputs the state $\mathbf{s}$, an uncorrected action $\mathbf{a}$ and observable parameters $\boldsymbol{\mu}$, and outputs the expected state error $\mathbf{e}$ when a policy trained in one environment is deployed to a different one $E: (\mathbf{s}, \mathbf{a}, \boldsymbol{\mu}) \mapsto \mathbb{R}^n$. We will discuss more details of training in the following

sections.

Algorithm 1: Train an Error Aware Policy.
1: **Input:** Environments $\mathbf{D} = \{(\boldsymbol{\mu}_0, \boldsymbol{\nu}_0), \cdots, (\boldsymbol{\mu}_K, \boldsymbol{\nu}_K)\}$
2: Pre-train $\pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\mu}_0, \mathbf{e} = 0)$ for $P(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \boldsymbol{\mu}_0)$ reference environment with $\mathbf{e} = 0$
3: **while** not done **do**
4:   Sample an environment with $(\boldsymbol{\mu}, \boldsymbol{\nu})$ from $\mathbf{D}$
5:   **for** each policy update iteration **do**
6:     Initialize buffer $\mathbf{B} = \{\}$
7:     Update an error function E using Algorithm 3
8:     $\mathbf{B}$ = Generate rollouts using Algorithm 2
9:     Update policy π using $\mathbf{B}$ with PPO.
10:   **end for**
11: **end while**
12: **return** $\pi(\mathbf{a}, |\mathbf{s}, \boldsymbol{\mu}, \mathbf{e})$

B. Training an Error-aware Policy

Training Procedure. The training process of an error-aware policy is summarized in Algorithm 1. Assume that we have an oracle error function $E(\mathbf{s}, \mathbf{a}, \boldsymbol{\mu})$ that outputs the expected state error in a novel environment, which will be explained in the following section. First, the policy is pre-trained to achieve the desired behavior only in the reference environment $(\boldsymbol{\mu}_0, \boldsymbol{\nu}_0)$ assuming there is no state error, $\pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\mu}_0, \mathbf{e} = 0)$. Once the policy is trained in the reference environment, we sample dynamics parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\nu}_i$ $(i > 0)$ uniformly from the data set $\mathbf{D}$ and evaluate the EAP in this new environment. The policy parameters are updated using a model-free reinforcement learning algorithm, Proximal Policy Optimization [30]. Sampling new testing environments and updating policy parameters are repeated until the convergence.

Algorithm 2: Generate Rollouts
1: **Input:** Observable dynamics parameters $\boldsymbol{\mu}$, Transition function P, Current policy π and error function E, Replay buffer $\mathbf{B}$
2: Sample state $\mathbf{s}$ from initial state distribution ρ_0
3: **while** not done **do**
4:   $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\mu}, \mathbf{e} = 0)$ // original action
5:   $\mathbf{e} = E(\mathbf{s}, \mathbf{a}, \boldsymbol{\mu})$ // predicted error
6:   $\hat{\mathbf{a}} \sim \pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\mu}, \mathbf{e})$ // error-aware action
7:   $\mathbf{s}' \sim P(\mathbf{s}'|\mathbf{s}, \hat{\mathbf{a}}, \boldsymbol{\mu})$
8:   $r = R(\mathbf{s}, \hat{\mathbf{a}})$
9:   $B = B \cup \{(\mathbf{s}, \hat{\mathbf{a}}, r, \mathbf{s}', \boldsymbol{\mu})\}$
10:   $\mathbf{s} = \mathbf{s}'$
11: **end while**
12: **return** B

Rollout Generation. A roll-out generation procedure is described in Algorithm 2. Given a state $\mathbf{s}$ in this environment $(\boldsymbol{\mu}, \boldsymbol{\nu})$, we query an action from policy π as if the policy is being deployed in the reference environment with $\mathbf{e} = 0$. This action $\mathbf{a}$ is fed into the error function E which predicts the expected state error in this environment, then the state error is passed into the error-aware policy to query a corrected action $\hat{\mathbf{a}}$ which will be applied to the actual system. The task reward $R(\mathbf{s}, \mathbf{a})$ guides the policy optimization to find the best "corrected" action that maximizes the reward.

Algorithm 3: Train an Error Prediction Function.
1: **Input:** Reference environment with $\boldsymbol{\mu}_0$
2: **Input:** Target environment with $\boldsymbol{\mu}$
3: **Input:** Replay Buffer $\mathbf{B}$
4: **Input:** Dataset $\mathbf{Z}$
5: **Input:** Error Horizon T
6: **while** not done **do**
7:   Sample the initial state $\mathbf{s}_0^0$ from $\mathbf{B}$
8:   $\mathbf{s}^0 = \mathbf{s}_0^0$
9:   **for** $t = 0 : T - 1$ **do**
10:     `// Simulation in Reference Env`
11:     $\mathbf{a}_0^t \sim \pi(\mathbf{a}|\mathbf{s}_0^t, \boldsymbol{\mu}_0, \mathbf{e} = 0)$
12:     $\mathbf{s}_0^{t+1} \sim P(\mathbf{s}_0^t, \mathbf{a}_0^t, \boldsymbol{\mu}_0)$
13:     `// Simulation in Validation Env`
14:     $\mathbf{a}^t \sim \pi(\mathbf{a}|\mathbf{s}^t, \boldsymbol{\mu}, \mathbf{e} = 0)$
15:     $\mathbf{s}^{t+1} \sim P(\mathbf{s}^t, \mathbf{a}^t, \boldsymbol{\mu})$
16:   **end for**
17:   $\mathbf{Z} = \mathbf{Z} \cup \{(\mathbf{s}^0, \mathbf{a}^0, \mathbf{s}^T, \mathbf{s}_0^T, \boldsymbol{\mu})\}$
18: **end while**
19: minimize the $L(\phi)$ in Eq. 1 using $\mathbf{Z}$.
20: **return** ϕ

C. Training an Error Function

In reality, we do not have an oracle error function that can predict the next state due to the lack of unobservable parameters $\boldsymbol{\nu}$. To this end, we will learn this function simultaneously with EAP, by splitting the dataset $\mathbf{D}$ into the training and validation sets. Similar to training methodology followed in meta-learning algorithms, we repeatedly apply the trained policy into sampled environments from the validation set. Because our nominal behavior is pre-trained in the reference environment $(\boldsymbol{\mu}_0, \boldsymbol{\nu}_0)$, we compute the errors by measuring the differences in the reference environment $(\boldsymbol{\mu}_0, \boldsymbol{\nu}_0)$ and the validation environment $(\boldsymbol{\mu}, \boldsymbol{\nu})$: $\mathbf{e} = (\bar{\mathbf{s}}' - \mathbf{s}') \in \mathbb{R}^n$, generated by two dynamic models $P(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \boldsymbol{\mu}_0)$ and $\bar{P}(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \boldsymbol{\mu})$.

Horizon of Error Prediction. In practice, we found that the error accumulated during one step is often not sufficient to provide useful information to the EAP. To overcome this challenge, we take the state in the collected trajectory and further simulate it for T steps in both the reference environment $P(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \boldsymbol{\mu}_0)$ and the validation environment $\bar{P}(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \boldsymbol{\mu})$. We provide analysis on the effect of horizon length from $T = 1$ to $T = 8$ in the Section IV.

Loss Function. Since the differences between the two dynamical environments reflects the reality gap caused by
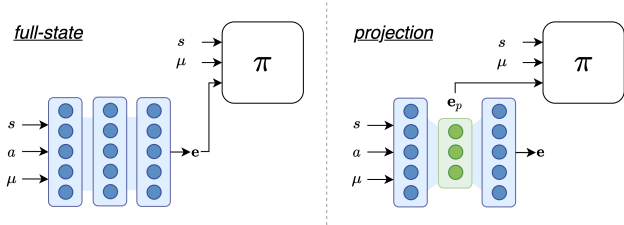
Figure 2: **Left :** A full state error representation input into the policy vs **Right :** Projected error representation as an input to the policy

unobservable parameters, the error prediction function E enables us to learn the effect of the unobserved parameters captured through the state error. We train our error prediction function E to learn this "emulated" sim-to-real gap by minimizing the following loss:

$$L(\phi) = \sum_{(\mathbf{s}^0,\mathbf{a}^0,\mathbf{s}^T,\mathbf{s}_0^T,\mu)\in\mathbf{Z}} ||E(\mathbf{s}^0,\mathbf{a}^0,\mu) - (\mathbf{s}_0^T - \mathbf{s}^T)||^2, \quad (1)$$

where $\mathbf{Z}$ is the collected dataset and ϕ is the parameters or the neural net representing E. Algorithm 3 summarizes the training procedure.

Reduced Representations. We experiment with two different representations of the error input to the policy. First, we input the full state error $\mathbf{e} = \mathbf{s}_0^T - \mathbf{s}^T$ (with the same dimension as the state) approximated by a MLP neural network, into the policy. Second, we use a network architecture with an information bottle neck, as illustrated in Figure 2, and input the latent representation $\mathbf{e}_p$ into the policy. The same loss function L is used to train both the functions.

IV. EVALUATION

We design experiments to validate the performance of error-aware policies. We aim to answer the following research questions.

1) Does an EAP show better zero-shot transfer on unseen environments compared to the baseline algorithms?
2) How does the choice of hyperparameters affect the performance of an EAP?

A. Baseline Algorithms

We compare our method with two baselines commonly used for sim-to-real policy transfer, Domain Randomization (DR)[22, 23] and Universal Policy (UP) [38]. DR aims to learn a more robust policy for zero-shot transfer, by training with randomly sampled dynamics parameters (in our case, both μ and ν). UP extends DR by taking dynamics parameters as additional input. UP often transfer to target environments better than DR, but it explicitly requires to know dynamics parameters, where ν is assumed to be unobservable in our scenario. We did not compare EAPs against meta-learning algorithms [2, 9, 5], which require additional samples from the validation environment.
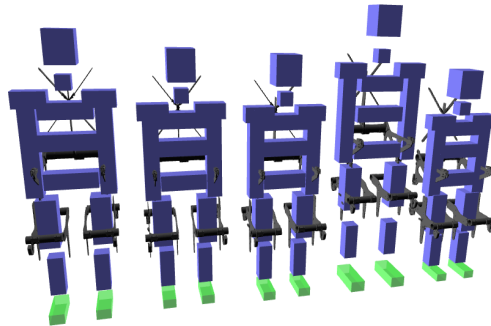

Figure 3: Five different test subjects for the assistive walking experiment with varying height, mass, leg length and foot length from the biomechanical gait dataset [10].

B. Tasks

We evaluate the performance of error-aware policies on five different tasks. The first task is about push-recovery of an assistive walking device for simulated humans, inspired by the work of Kumar et al [18]. The second task is locomotion of a quadrupedal robot, Aliengo Explorer[1]. The rest three tasks are CartPole, Hopper, and Walker2D, which are from the OpenAI benchmark suite [3].

1) Assistive walking device for push recovery: In this task, the goal is to learn a policy for an assistive wearable device (i.e. exoskeleton) to help a human recover balance after an external push is applied. (inset figure). We use a hip exoskeleton that applies torques in 2-degrees of freedom at each hip joint. Our algorithm begins by training 15 human agents using public biomechanical gait data [10] to walk in a steady-state gait cycle, similar to the approach presented in [24]. The 15 agents vary in mass, height, leg length, and foot length according to the biomechanical data used to train their corresponding policies, which formulate the four-dimensional observable parameters μ (Figure 3). We also vary each human agent's joint damping, maximum joint torques, PD gains, and sensory delay as the four dimensional unobservable parameters ν. We split the 15 human agents into 10 for the training set and 5 as the testing set.

Human Behavior Modeling. First, we capture the human behavior by training a human-only walking policy π_h that mimics the reference motion which frames are denoted as $\bar{\mathbf{q}}$. Each human model has 23 actuated joints along with a floating base. The state space has 53 dimensions, $\mathbf{s}_h = [\mathbf{q}, \dot{\mathbf{q}}, \mathbf{v}_{com}, \omega_{com}, \psi]$, which represent joint positions, joint velocities, linear and angular velocities of the center of mass, and a phase variable ψ that indicates the target frame in the reference biomechanical gait cycle. The action $\mathbf{a}$ is defined as the offset to the reference biomechanical joint trajectory $\bar{\mathbf{q}}(\psi)$, which results in the target angles: $\mathbf{q}^{target} = \bar{\mathbf{q}} + \mathbf{a}$. The reward function encourages mimicking

Task	Observable Params. $\boldsymbol{\mu}$	Unobservable Params. $\boldsymbol{\nu}$	Net. Arch.	Err. Dim. $|\mathbf{e}_p|$
Assitive Walking	mass, height, leg length, and foot length	joint damping, max torques, PD gains and delay	(64, 32)	6
Aliengo	PD gains, link masses	sensor delay, joint damping, ground friction	(64, 32)	6
Cartpole	pole length, pole mass, cart mass	joint damping, joint friction	(32, 16)	2
Hopper	thight mass, foot mass, shin length	joint damping, ground friction	(32, 16)	4
Walker 2D	link masses, shin length	sensing delay, joint damping, ground friction	(64, 32)	5

Table I: Tasks and Network Architectures

Observable Params. $\boldsymbol{\mu}$		
Parameter	Training range	Testing range
Mass	[45,76] kg	[55,95] kg
Height	[143,182] cm	[155,197] cm
Leg-length	[70,88] cm	[80,95] cm
foot length	[21,24] cm	[24,26] cm

Table II: Ranges of variation for observable parameters during training and testing in the assistive walking task.

Unobservable Params. $\boldsymbol{\nu}$		
Parameter	Training range	Testing range
Joint damping	[0.3,0.6]	[0.5,0.8]
Max torques	[120,180]	[155,200]
PD gains (P,D)	[(500,25),(750,50)]	[(650,30),(800,50)]
Delay	[30,60] ms	[45,70] ms

Table III: Ranges of variation for unobservable parameters during training and testing in the assistive walking task.

the reference motions from public biomechanical data:

$$R_{human}(\mathbf{s}_h,\mathbf{a}_h) = w_q(\mathbf{q}-\bar{\mathbf{q}}) + w_v(\dot{\mathbf{q}}-\bar{\dot{\mathbf{q}}})$$
$$+ w_c(\mathbf{c}-\bar{\mathbf{c}}) + w_p(\mathbf{p}-\bar{\mathbf{p}}) - w_\tau||\boldsymbol{\tau}||^2, \quad (2)$$

where the terms include the reference joint positions $\bar{\mathbf{q}}$, joint velocities $\bar{\dot{\mathbf{q}}}$, end-effector locations $\bar{\mathbf{p}}$, contact flags $\mathbf{c}$, and the joint torques $\boldsymbol{\tau}$. During training, we exert random forces to the agent during policy training. Each random force has a magnitude uniformly sampled from $[0,800]$ N and a direction uniformly sampled from $[-\pi/2,\pi/2]$, applied for 50 milliseconds on the agent's pelvis in parallel to the ground. The maximum force magnitude induces a velocity change of roughly $0.6-0.8$ m/sec. This magnitude of change in velocity is comparable to experiments found in biomechanics literature such as [34],[19] and [12]. We also randomize the time when the force is applied within a gait cycle. The forces are applied once at a randomly chosen time in each trajectory rollout. Similar to [18], we enforce joint torque constraints and introduce sensing delays during training to prevent the human agent to adapt to external disturbance really well.

MDP Formulation. Once the human agents are trained, we begin learning the push-recovery EAP for the assistive device. The objective is to stabilize the human gait from external perturbations. The 17 dimensional state of robot is defined as $\mathbf{s}_e = [\boldsymbol{\omega},\boldsymbol{\alpha},\ddot{\mathbf{x}},\mathbf{q}_{hip},\dot{\mathbf{q}}_{hip}]$, which comprises angular velocity, orientation, linear acceleration, hip joint positions, and hip joint velocity. The four dimensional action $\mathbf{a}_e$ consists of torques at two hip joints. The reward function maximizes the quality of the gait while minimizing the

impact of an external push.

$$R_{exo}(\mathbf{s}_h,\mathbf{s}_e,\mathbf{a}_e) = R_{human}(\mathbf{s}_h)$$
$$- w_1\|\mathbf{v}_{com}\| - w_2\|\boldsymbol{\omega}_{com}\| - w_3\|\mathbf{a}_e\|, \quad (3)$$

where R_{human} is defined in equation 2 , and $\mathbf{v}_{com}$ and $\boldsymbol{\omega}_{com}$ are the global linear and angular velocities of the pelvis. The last term penalizes the torque usage. We use the same weight $w_1 = 2.0$, $w_2 = 1.2$ and $w_3 = 0.001$ for all our experiments.

2) Quadrupedal Locomotion: In our second task, we learn a control policy that generates a walking motion for a quadrupedal robot, Aliengo Explorer [1]. For this task, the 17 observable parameters ($\boldsymbol{\mu}$) are PD gains of the joints, link and root masses and the 10 unobservable parameters $\boldsymbol{\nu}$ include sensing delay, joint damping of thigh and knee joints and ground friction. The 39-dimensional state space consists of torso position and orientation and corresponding velocities, joint position and velocities, foot contact variable that indicates when each foot should be in contact with the ground, while the 12-dimensional action space consists of joint velocity targets which is fed into a PD controller that outputs torques to each joint.

The reward function is designed to track the target motion that walks at 0.8 m/s:

$$r(\mathbf{s},\mathbf{a}) = w_1 e^{-k_1*(\mathbf{q}-\bar{\mathbf{q}})} + w_2 e^{-k_2*(\dot{\mathbf{q}}-\bar{\dot{\mathbf{q}}})}$$
$$+ w_3\min(\dot{x},0.8) + \sum_{i=1}^{4}||c_i-\bar{c}_i||^2. \quad (4)$$

In this equation, the first term encourages to track the desired joint positions, the second term is to track the desired joint velocities, the third term is for matching the forward velocity $\dot{x}$ to a target velocity of 0.8 m/s. and the four term tracks the predefined contact flags. We use the same weight $k_1 = 35, w_1 = 0.75$, $k_2 = 2, w_2 = 0.20$, and $w_3 = 1.0$ for all experiments.

3) OpenAI Environments: We test our method on three OpenAI environments: CartPole, Hopper and Walker2D. While using the same state spaces, action spaces, and the reward functions described in the benchmark [3], we additionally define observable and unobservable dynamics parameters as follows:

1) **Cartpole.** Observable parameters $\boldsymbol{\mu} \in R^3$ includes the length of the pole, the mass of the pole, and the mass of cart. Unobservable parameters $\boldsymbol{\nu} \in R^3$ include the damping at the rotational joint, the friction at the rotational joint, and the friction at the translational joint.

(a) Hopper

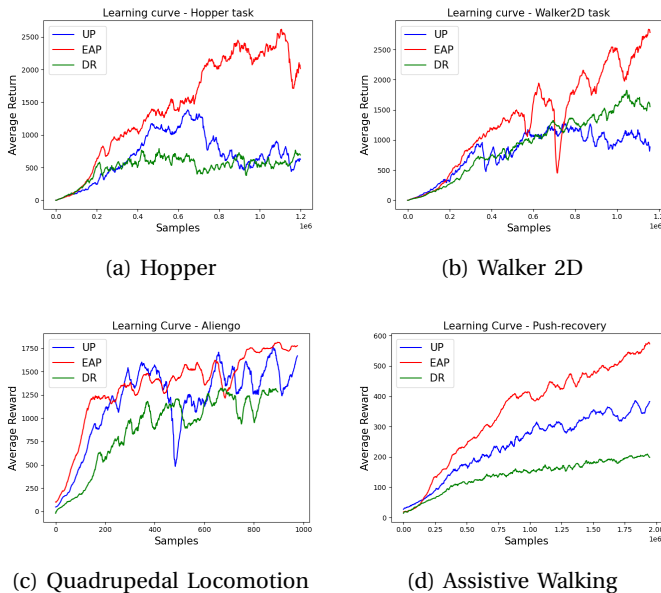(b) Walker 2D

(c) Quadrupedal Locomotion

(d) Assistive Walking

Figure 4: Learning curves for four tasks. The number of samples for EAP include the ones generated for training an error function.

2) **Hopper.** Observable parameters $\mu \in R^3$ include the mass of the thigh and foot and the length of the shin bodynode. Unobservable parameters $\nu \in R^3$ include joint damping of shin and foot joints and ground friction.

3) **Walker 2D.** Observable parameters $\mu \in R^6$ include the masses of thigh and foot for both legs, the mass of pelvis, and the length of shin. Unobservable parameters $\nu \in R^4$ include joint damping of foot joints, the delay in observation, and ground friction.

C. Zero-shot Transfer with EAPs

In this section, we compare the zero-shot transfer of error-aware policies against two other baseline algorithms, Domain Randomization (DR) and Universal Policies (UP).
Learning Curves. First, we compare the learning curves of the EAP, DR, and UP approaches on four selected tasks in Figure 4. We set the same ranges of the observable and unobservable parameters for all three algorithms. In our experience, EAPs learn faster than DR and UP for three tasks, the Hopper, Walker2D, and assistive walking tasks, while showing comparable performance for the quadrupedal locomotion task. Note that, to make the comparison fair to baselines, we also include the samples for training error functions (Algorithm 3) when we evaluate the performance of EAPs. We do not include the experiment on the Cart-Pole environment for brevity but the EAP outperforms the baselines as well

Zero-shot Transfer. Then we evaluate the learned policies on unseen validation environments, where their dynamics parameters μ and ν are sampled from the outside of the training range. We conduct the experiments for the CartPole, Hopper, Walker2D and quadrupedal locomotion tasks and compare the *normalized* average returns (the average return divided by the maximum return). The results
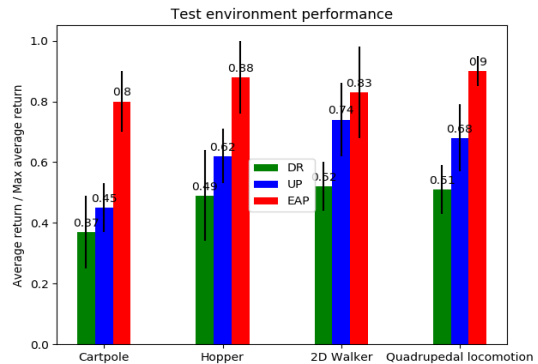

Figure 5: Comparison of EAP and baselines DR and UP. The error bars represent the variation in the average return of the policy in the target environment when trained with 4 different seeds.
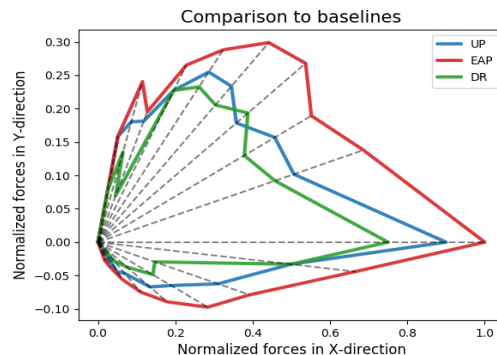

Figure 6: Average stability region in five test subjects. The results indicate the better zero-shot transfer of EAP over DR and UP.

are plotted in Figure 5, which indicate that EAP outperforms DR by 60% to 116% and UP by 12% to 77%. Note that UP may perform well for the real-world transfer due to the lack of the unobservable parameters. We also observe that UP is consistently better than DR by being aware of the dynamics parameters, μ and ν, which meets our expectation.

For evaluating the zero-shot transfer for the assistive walking task, we define an additional metric "stability region", which depicts the ranges of maximum perturbations in all directions that can be handled by the human with the EAP-controlled exoskeleton. We train policies for 10 training human subjects and test the learned policies for 5 new human subjects. Figure 6 compares the average performance of EAP with DR and UP. The larger area of stability region indicates that EAP significantly outperforms two baselines. The ranges of variation for observable and unobservable parameters during training and testing phases are included in tables II and III.

D. Ablation study

We further analyze the performance of EAPs by conducting a set of ablation studies. We studied four categories of parameters: choices of observable parameters, reference dynamics, error prediction horizons, and error representations.
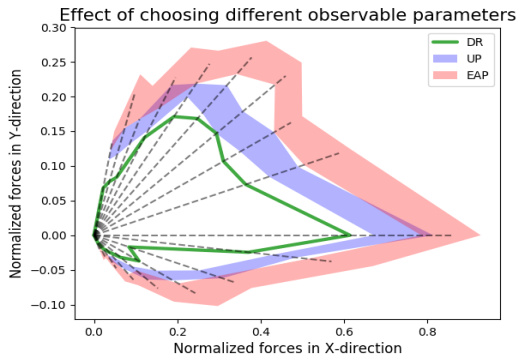
Figure 7: Ablation study with choosing different observable parameters as $\boldsymbol{\mu}$. The result indicates that our approach (EAP) shows more reliable zero-shot transfers for all different scenarios.
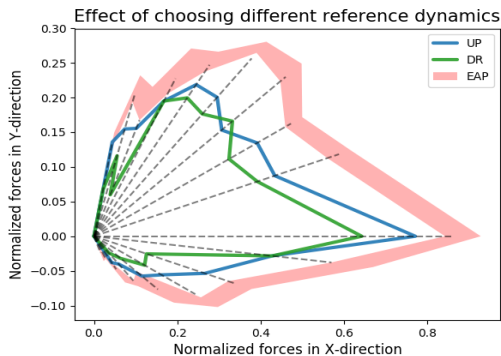

Figure 8: Ablation study with different reference dynamics. The results indicate that our algorithm is robust against the choice of different references.

Choice of Observable and Unobservable Parameters. We check the robustness of EAPs by testing with different choices of observable and unobservable parameters. We randomly split the parameters into $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ and test three different splits. Figure 7 shows the stability regions for all three algorithms for three different scenarios. In all cases, EAPs are more robust than the baseline algorithms.

Choice of Reference Dynamics. In this study, we analyze the effect of choosing three different reference dynamics $P(\mathbf{s}'|\mathbf{a}, \mathbf{s}, \boldsymbol{\mu}_0, \boldsymbol{\nu}_0)$ on the performance of EAP. We randomly choose three different human agents as the reference dynamics and follow the learning procedure of EAPs to train three different policies. These policies are then deployed on the same test subjects along with UP and DR policies. Figure 8 shows that all the EAPs outperforming the baselines by having larger stability regions, although EAPs have slightly larger variances.

Horizon of Error Prediction. As we motivated in Section III-C, one step error might be too subtle to inform the learning of EAPs and we may need T step expansion to enlarge them. We studied the effect of the error prediction horizon T in Algorithm 3 by varying its value from $T = 1$ to $T = 8$ for the assistive walking task. Figure 9 shows the normalized average return over T gradually changes over the different values of T and peaks at $T = 5$. Therefore, we
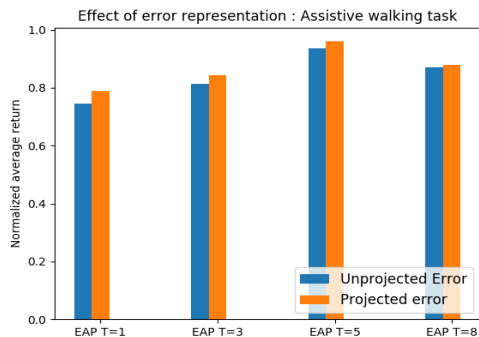

Figure 9: Ablation study with different parameter setting for EAP training.

set $T = 5$ for all the experiments.

The error representation. We also compare the effect of the error representation. Figure 9 also plots the normalized average returns of the unprojected errors (blue) and projected errors (orange), where projected errors show slightly better performance for all the different T values.

V. CONCLUSIONS

We presented a novel approach to train an error-aware policy (EAP) that transfers effectively to unseen target environments in a zero-shot manner. Our method learns an EAP for an assistive wearable device to help a human recover balance after an external push is applied. We show that a single trained EAP is able to assist different human agents with unseen biomechanical characteristics. We also validate our approach by comparing EAP to common baselines like Universal Policy and Domain randomization to show our hypothesis that a policy which explicitly takes future state error as input can enable better decision making. Our approach outperforms the baselines in all the tasks. We also evaluated the performance of our algorithm through a series of ablation studies that sheds some light on the importance of parameters such as error horizon length, error representation, choice of observable parameters and choice of reference dynamics. We find that EAP is not sensitive to either the choice of observable parameters or the reference dynamics, and outperforms the baselines with variations in these quantities as well.

Our work has a few limitations. At the core, our algorithm relies on the error function to make predictions of the expected state errors. The accuracy of this prediction can be improved by better function approximators such as recurrent neural networks (RNN) that takes a history of states as input, we leave this for future work. We also aim to test our approach on real-world robot.

REFERENCES

[1] Unitree's aliengo quadrupedal robot. URL https://www.unitree.com/products/aliengo/.
[2] Suneel Belkhale, Rachel Li, Gregory Kahn, Rowan McAllister, Roberto Calandra, and Sergey Levine.

Model-based meta-reinforcement learning for flight with suspended payloads, 2020.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[4] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience, 2018.

[5] Ignasi Clavera, Anusha Nagabandi, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. *CoRR*, abs/1803.11347, 2018. URL http://arxiv.org/abs/1803.11347.

[6] Siddharth Desai, Haresh Karnan, Josiah P. Hanna, Garrett Warnell, and Peter Stone. Stochastic grounded action transformation for robot learning in simulation, 2020.

[7] Ioannis Exarchos, Yifeng Jiang, Wenhao Yu, and C. Karen Liu. Policy transfer via kinematic domain randomization and adaptation, 2020.

[8] David D. Fan, Jennifer Nguyen, Rohan Thakker, Nikhilesh Alatur, Ali akbar Agha-mohammadi, and Evangelos A. Theodorou. Bayesian learning-based adaptive control for safety critical systems, 2019.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[10] Claudiane Fukuchi, Fukushi Reginaldo, and Marcos Duarte. A public dataset of overground and treadmill walking kinematics and kinetics in healthy individuals. *Peer journal*, 2018.

[11] Aditya Gahlawat, Pan Zhao, Andrew Patterson, Naira Hovakimyan, and Evangelos Theodorou. L1-gp: L1 adaptive control with bayesian learning. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger, editors, *Proceedings of Machine Learning Research*, volume 120, pages 826–837, The Cloud, 10–11 Jun 2020. PMLR. URL http://proceedings.mlr.press/v120/gahlawat20a.html.

[12] AL Hof, SM Vermerris, and WA Gjaltema. Balance responses to lateral perturbations in human treadmill walking. *Journal of Experimental Biology*, 213(15):2655–2664, 2010.

[13] Zhicong Huang, Junqiang Liu, Z. Li, and C. Su. Adaptive impedance control of robotic exoskeletons using reinforcement learning. *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 243–248, 2016.

[14] 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.

[15] R. W. Jackson and S. H. Collins. Heuristic-based ankle exoskeleton control for co-adaptive assistance of human locomotion. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(10):2059–2069, 2019. doi: 10.1109/TNSRE.2019.2936383.

[16] Marija Jegorova, Joshua Smith, Michael Mistry, and Timothy Hospedales. Adversarial generation of informative trajectories for dynamics system identification. *arXiv preprint arXiv:2003.01190*, 2020.

[17] Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C. Karen Liu, Sergey Levine, and Jie Tan. Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning, 2021.

[18] Visak C V Kumar, Sehoon Ha, Gergory Sawicki, and C. Karen Liu. Learning a control policy for fall prevention on an assistive walking device. *Internation Conference of Robotics and Animation*, 2019.

[19] D. Martelli, V. Vashista, S. Micera, and S. K. Agrawal. Direction-dependent adaptation of dynamic gait stability following waist-pull perturbations. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(12):1304–1313, Dec 2016. ISSN 1558-0210. doi: 10.1109/TNSRE.2015.2500100.

[20] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J. Pal, and Liam Paull. Active domain randomization, 2019.

[21] Fabio Muratore, Christian Eilers, Michael Gienger, and Jan Peters. Data-efficient domain randomization with bayesian optimization, 2021.

[22] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, 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, 2018.

[23] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *CoRR*, abs/1710.06537, 2017. URL http://arxiv.org/abs/1710.06537.

[24] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018)*, 2018.

[25] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems*, 07 2020. doi: 10.15607/RSS.2020.XVI.064.

[26] Zhinan Peng, Rui Luo, Rui Huang, Jiangping Hu, Kecheng Shi, Hong Cheng, and Bijoy Kumar Ghosh. Data-driven reinforcement learning for walking assistance control of a lower limb exoskeleton with hemiplegic patients. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9065–9071.

IEEE, 2020.

[27] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *CoRR*, abs/1703.02702, 2017. URL http://arxiv.org/abs/1703.02702.

[28] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles, 2017.

[29] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators, 2019.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] Yuda Song, Aditi Mavalankar, Wen Sun, and Sicun Gao. Provably efficient model-based policy adaptation, 2020.

[32] 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. *CoRR*, abs/1804.10332, 2018. URL http://arxiv.org/abs/1804.10332.

[33] 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.

[34] Yang Wang and Manoj Srinivasan. Stepping in the direction of the fall: The next foot placement can be predicted from current upper body state in steady-state walking. *Biology Letters*, 10(9), 2014. ISSN 1744957X. doi: 10.1098/rsbl.2014.0405.

[35] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel van de Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of Machine Learn-*

[39] Wenhao Yu, Visak CV Kumar, Greg Turk, and C. Karen

ing Research, volume 100, pages 317–329. PMLR, 30 Oct–01 Nov 2020. URL http://proceedings.mlr.press/v100/xie20a.html.

[36] Z. Xu, C. Tang, and M. Tomizuka. Zero-shot deep reinforcement learning driving policy transfer for autonomous vehicles based on robust control. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2865–2871, 2018. doi: 10.1109/ITSC.2018.8569612.

[37] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. *CoRR*, abs/1907.03613, 2019. URL http://arxiv.org/abs/1907.03613.

[38] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

Liu. Sim-to-real transfer for biped locomotion, 2019.

[40] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5(2):2950–2957, 2020.

[41] Y. Yuan, Z. Li, T. Zhao, and D. Gan. Dmp-based motion generation for a walking exoskeleton robot using reinforcement learning. *IEEE Transactions on Industrial Electronics*, 67(5):3830–3839, 2020. doi: 10.1109/TIE.2019.2916396.

[42] Juanjuan Zhang, Pieter Fiers, Kirby A. Witte, Rachel W. Jackson, Katherine L. Poggensee, Christopher G. Atkeson, and Steven H. Collins. Human-in-the-loop optimization of exoskeleton assistance during walking. *Science*, 356(6344):1280–1284, 2017. ISSN 0036-8075. doi: 10.1126/science.aal5054. URL https://science.sciencemag.org/content/356/6344/1280.

[43] Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies, 2019.