2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Safe and Efficient Auto-tuning to Cross Sim-to-real Gap for Bipedal Robot

Yidong Du, Xuechao Chen, Zhangguo Yu, Yuanxi Zhang, Zishun Zhou, Jindai Zhang, Jintao Zhang, Botao Liu and Qiang Huang

***Abstract*— Recent advances in both legged robot locomotion and Reinforcement Learning have shown a promising path for developing bipedal robot controllers. While the difference in dynamics between real world and simulation, also known as reality gap, still hinders the use. In this paper, we focus on sim-to-real bipedal robot locomotion task. We leverage the recent advances in auto-tuning sim-to-real transfer and use it to address sim-to-real bipedal robot locomotion problem. Similar to existing work, we first train a parameter searching model with dataset collected from simulator and use real-world data to tune the simulation parameters. However, the prediction tuning can be unreliable if the training dataset distribution fails to cover the real-world data. We address this problem by formulating this problem as an Out-of-distribution problem and further extending the current framework with a dataset verification model. With extended module, our method is capable of tuning the simulation parameters safely and efficiently. We demonstrate our method outperforms existing work and achieves sim-to-real bipedal robot locomotion on bipedal robot BITeno.**

I. INTRODUCTION

In last forty years, great progress has been made in legged robot locomotion control by dynamical modeling and control theory. Meanwhile, there are also growing interest in learning continuous control tasks, which makes legged robot locomotion control using Reinforcement Learning (RL) an attractive alternative because of its less requirement for expert knowledge. While there are some attempts on learning directly on real robot [1], most current RL algorithms still struggle with collecting large scale data directly on real-world legged robot because it's unsafe and time-consuming. On the other hand, physics simulator is a tool that can provide necessary amount of data safely and quickly. So the current standard paradigm is to train a policy in a physics simulation environment and transfer to real world in zero-shot manner or sim-to-real techniques. However, this transfer is often challenging due to the reality gap [2] caused by imprecise modeling and other factors.

[2] proposes to auto-tune the system parameters of simulation to cross the reality gap using small amount of real-world observations, by training a Searching Parameter Model (SPM) and formulating the parameter identification process as a searching problem. SPM is trained to predict whether the input parameters is higher or lower than the parameters corresponding to the given real-world observations. By shifting parameter distribution using SPM iteratively using real-world data, the simulation becomes more similar to the real-world environment. However, [2] fails to provide a method to ensure the collected real-world observation does not differ from the dataset that SPM is trained on. In legged robot locomotion task, this brings a potential problem that SPM could make unreliable prediction in parameter shift, and then reduce the robot performance in real-world. The reduction of performance in real-world scenarios can be critical since bipedal robot is highly unstable robotic system. The causes of this problem have similarities with the most notable shortcoming of domain randomization method that expert knowledge is needed to determine which and to what extend the parameters should be varied to cover the real-world environment. This potential data unreliability problem is also known as out-of-distribution (OoD) problem [3].

In this paper, to overcome the sim-to-real gap for bipedal robot locomotion task, we borrow the idea of formulating parameter searching problem, and further propose to design a Data Verification Model (DVM), namely an OoD detector. We propose to design the DVM based on Variational Autoencoder (VAE), a generative model commonly deployed for OoD detection [4] for its capability of reconstructing normal samples well and struggling with the OoD samples [4]. Then our method is able to determine if the real-world observations are covered, and effectively prevents unreliable parameter shift problem. Overall. we propose a sim-to-real framework which is capable of auto-tuning the simulation parameters safely and efficiently to improve the simulation performance to match real world for bipedal robot locomotion task.

Our main contribution are threefold:

1. We design a framework for optimizing simulation parameters to cross sim-to-real gap safely and efficiently for bipedal robot locomotion task.
2. We propose to address the problem of potential unreliable parameter shift during parameter tuning by posing it as an OoD problem.
3. We demonstrate our method in both sim-to-sim and sim-to-real bipedal robot locomotion experiments.

II. RELATED WORKS

A. Sim-to-real for Robotics

The idea of domain randomization is commonly used to cross the sim-to-real gap, by randomizing the parameters of simulation environment to train a more robust policy. This idea has been successfully applied to many robotic task including object manipulation [5] and legged robot

∗This work was supported by the National Natural Science Foundation of China under Grant 62073041.

All authors are with School of Mechatronic Engineering, Beijing Institute of Technology, Beijing, 100081, China. Email: chenxuechao@bit.edu.cn

locomotion [6], [7]. But the policy trained with this method tends to be conservative and even struggles to converge if the environment is over randomized [8].

To address the problem of demand for expert knowledge and for better coverage of real-world environment, recent works explore automatic tuning of parameter distribution instead of hand-crafted tuning and heuristics as a variant of domain randomization method. [9] proposes SimOpt, to cross the reality gap by optimizing the distribution simulation parameter via trajectory matching in simulation and real robot. While SimOpt require time-consuming data collection in real world and careful cost function design. [10] proposes active domain randomization which learns a parameter sampling strategy to optimize parameter distribution by leveraging policy performance difference in simulation and real world. [11] shifts the simulation parameter distribution to match the real-world using Bayesian estimation method.However, these works mainly focus on robot manipulation tasks such as grasping or opening drawer.

Other work propose to augment the simulated physics engine to address the inaccuracy in simulation. [12] designed a differentiable physics engine and augment the physics engine via neural networks. [13] corrects the difference between simulation and real world by Gaussian Process to solve the problem of marble moving in a circular maze. [14] proposes a novel residual physics learning method to apply external forces to match simulation and real world to help cross the sim-to-real gap of buoyancy assisted legged robot.

The work most related to ours is [2], which proposes to formulate the parameter tuning as a search problem by training a searching parameter model (SPM), and we borrow the idea of parameter searching model. While [2] mainly focus on image-based robot manipulation task, we focus on legged robot locomotion task. And our method also addresses safe tuning in parameter optimization process.

B. Sim-to-real for Legged Robot Locomotion

There has been many efforts to bridge the sim-to-real gap to achieve legged robot locomotion using RL. Domain randomization is also a common practice in legged robot locomotion task. [6], [7] use domain randomization to overcome sim-to-real gap on quadruped robot. [15], [16] apply domain randomization to bipedal robot Cassie to achieve locomotion task.

Alternatively, some work train policy conditioned on real-world physics parameters via online explicit or implicit system identification. [17] propose to infer physics parameters through a module trained in simulation during real-world deployment. [18] predicts low dimensional latent embedding of system parameters using real-world experience by Bayesian optimization. [8], [19] train adaptation module for online environment identification in simulation and directly deploy on real quadruped and humanoid robot respectively. Instead of achieving sim-to-real by inferring physics parameters in real world, our method adjusts the simulation parameters to match real world.

III. Method

In this section, we propose our transfer algorithm to overcome sim-to-real gap of bipedal robot. The gap between simulation and real world mainly arises from two factors. First, the physics engine of simulation is designed based on a handful of laws and expert knowledge, which makes it often an approximation of real-world physics for computational tractability. And this problem is aggravated by the fact that bipedal robot is a complex dynamical robot system. Second, there are often unpredictable errors in observation such as noise and delays in real-world robotic systems, while in simulation the observation is accurate. A common practice is to use domain randomization by applying a relatively wide range of simulation parameters to address both problem, but this requires expert knowledge and careful manual engineering.

Our algorithm addresses the problem by combining traditional system identification method and learning-based parameter tuning method. We also assume to sample simulation parameter from a parameter distribution and further tune the simulation parameters distribution using real-world robot locomotion data. So it falls into the category of combination of system identification and domain randomization. The framework of our algorithm is shown in Fig. 1, which includes a data distribution verification module and searching parameter module. In the rest of this section, we provide the details of our method.

A. Preliminary and Problem Statement

We model the bipedal robot locomotion problem as a Markov Decision Process (MDP), defined by $M = (S, A, P, R, \gamma)$, where S is the state space, A is action space, $P(s_{t+1}|s_t, a_t, \xi_{Real})$ is the transition function with real-world dynamics parameters ξ_{Real}, $R(s_t, a_t, s_{t+1})$ is the reward function,γ is the discount factor. In practice, a noisy observation space O and observation function $Z(o_t|s_t)$ need to be introduced because of the difficulty of estimating the true states.The objective of RL is to find a policy $\pi_\theta(a_t|s_t)$ which maximizes the cumulative discount rewards $\mathbb{E}[\sum_t^\infty[\gamma^t r_t]]$.In this work, the policy is trained in simulation, so we also introduce simulation parameter ξ_{Sim} which affects the simulation dynamics $P(s_{t+1}|s_t, a_t, \xi_{Sim})$.Normally, ξ_{Sim} differs from ξ_{Real}.Domain randomization hopes to cover ξ_{Real} by sample ξ_{Sim} from Ξ, where Ξ is a handcrafted parameter distribution. While the aim of our method is to tune and get Ξ^* that maximizes $\mathbb{E}_{\pi_\theta}^{\xi_{Real}}[\sum_t^\infty[\gamma^t r_t]]$, where π_θ is trained in simulation parameterized by $\xi_{Sim}^* \sim \Xi^*$.

VAE is a generative model which reconstructs input data using probabilistic latent model. It consists of two attached networks known as encoder and decoder. The encoder and decoder learn posterior distribution $e(z|x)$ and likelihood distribution $d(x|z)$ respectively, where x is input, and z is latent variable (normally encoder and decoder are denoted as p and q). The encoder and decoder are usually parameterized by neural networks. During training, the Evidence Lower Bound (ELBO) is maximized, where ELBO is written as:

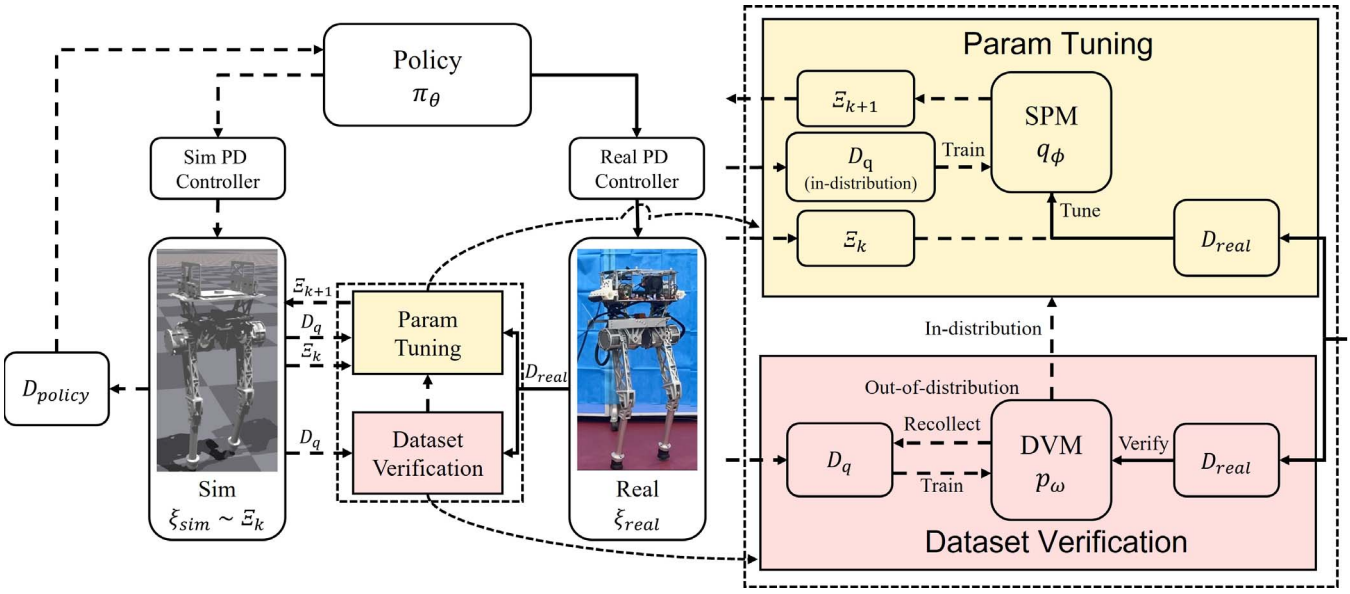


Fig. 1. Method Overview: First D_{policy} is collected in initially parameterized Ξ_k simulated environment to train policy π_θ. Then we use π_θ to collect D_{real} on real-world robot and D_q on simulated robot. D_q can be used to train SPM (q_ϕ) only if the D_{real} is identified as in-distribution by DVM (p_ω), otherwise we expand Ξ and re-collect D_q till the verification passed. Then Ξ_k is updated to Ξ_{k+1} by q_ϕ using real-world robot data D_{real} to match the real world. Dashed lines represent information passing in simulated environment, while solid line is for real world.

$$ELBO(x) = \mathbb{E}_{d(z|x)}\left[\log e(x|z)\right] - D_{KL}\left(d(z|x)\|p(z)\right) \quad (1)$$

where D_{KL} is Kullback-Liebler divergence, and $p(z)$ is prior distribution.

B. Joint Policy Learning and Safe Parameter Auto-tuning

1) Formulating parameter tuning as search problem: Direct system identification by mapping sequences of observations and actions to system parameters are usually challenging, so we train a binary classifier SPM denoted as p_ω to predict from the real-world trajectory whether the current simulation parameter is higher or lower than the target parameters that actually produce the input trajectory. By training such a binary classifier, the parameter identification problem is reformulated as a search problem, and the simulation parameters can be adjusted by SPM. SPM is trained on dataset D_q consisting of simulated trajectory generated from current simulation parameter using current policy and randomly generated system parameters, and used to iteratively update the simulation parameter using dataset D_{real} consisting of trajectory collected from real-world robot. Note that in this paper we focus on bidepal robot locomotion task, so the simulation parameters are related to robot dynamics such as foot friction. The parameters are updated based on the confidence of prediction, either increasing or decreasing the value incrementally.

2) Formulating data unreliability as OoD problem: The most challenging problem of domain randomization is if the randomized environment fails to cover real-world environment, it will be unlikely to get a policy that exhibits good performance after transfer. Similar challenge still exists in SPM method. If SPM is overfitted to a misparametrized simulated environment, there would be risk to make wrong prediction and reduce the robot performance or damage the hardware during further real-world data collection procedure. To prevent this problem, previous work collects D_q separately on a wide distribution Ξ and avoids using less confident data. By doing this, the covered parameter distribution in D_q is effectively expanded, but does not address the underlying problem completely.

To address this problem, we formulate this problem as an OoD detection problem, in which data produced by a covered environment is classified as in-distribution data, otherwise as out-of-distribution data. Specifically, we design the DVM using VAE, denoted as q_ϕ. VAE should be capable of reconstructing data within the training data distribution but not the OoD samples, and thus OoD samples can be detected by its relatively high reconstruction error. Similarly, we train the DVM on the simulated robot locomotion data D_q, and therefore the DVM should be capable of classifying the non-covered real-world robot data as out-of-distribution. We define the statistic of out-of-distribution measurement as:

$$\begin{aligned}\sigma = &\frac{1}{|D_{real}|}\sum_{D_{real}}[ELBO(\{o_t^{real}, a_t^{real}\}_{t=1}^{T}) - \\ &\frac{1}{|Q|}\sum_{Q} ELBO(\{o_t^{sim}, a_t^{sim}\}_{t=1}^{T})]\end{aligned} \quad (2)$$

where Q is the set of N samples whose encoded latent vectors are closest to the test sample's in latent space but within a predefined distance. Intuitively σ is the mean of difference of the ELBOs from the real-world test samples and the mean of ELBOs from the training samples in Q. And hence a high statistic suggests the trained DVM fails to reconstruct D_{real}

and implies the D_{real} is out-of-distribution. So optimizing parameters using SPM trained on current D_q could lead to incorrect parameter shift. Conversely low statistic implies the current real-world robot trajectory is in-distribution and reliable for further parameter tuning.

3) Joint learning and tuning: As simulation updating using SPM, the distribution of parameters Ξ_{k+1} deviates from the distribution Ξ_k that the pretrained SPM and policy is trained on. So the policy and SPM combining with DVM should both be updated iteratively, which forms an alternating learning process. This joint network training and parameter tuning process is detailed in Algorithm 1.

We use separate buffers D_π and D_q to train policy and SPM with DVM respectively. We define the parameter distribution from which the simulation parameter are sampled as uniform distribution $\Xi \sim U(e, r)$, with mean e and range r. First Ξ is initialized with relatively small r. and a threshold of statistic σ is set to verify if the real-world robot dataset D_{real} is in-distribution for SPM dataset before tuning the parameter. If the real-world data is verified to be out-of-distribution, then the r is increased by $r = cr$ to include more diverse data and redo the collection and training procedure, till the verification is passed.

Algorithm 1 Safe Parameter Auto-tuning

Initialize $\Xi_0 \sim U(\mu, r),\ \sigma^{'},\ c(c > 1),\ \alpha,\ K$
Initialize $D_{policy},\ D_q,\ D_{real},\ \pi_\theta,\ p_\omega,\ q_\phi$
for $k = 1 : K$ **do**
  Sample $\xi_{sim} \sim \Xi_k$
  for $t = 1 : T_{policy}$ **do**
    Collect simulated data, store in D_{policy}
    Train π_θ, update θ with (o_t^{sim}, R_t) in D_{policy}
  end for
  Collect real-world data with π_θ, store in D_{real}
  do
    for $t = 1 : T_{policy}$ **do**
      Sample $\epsilon_{sim} \sim U(\mu_k, r)$
      Collect simulated data, store in D_q
    end for
    Train p_ω, update ω with $(\{o_t^{sim}, a_t^{sim}\}_{t=1}^T)$ in D_q
    Calculate σ with
    $\sigma \leftarrow \frac{1}{|D_{real}|} \sum_{D_{real}} [ELBO(\{o_t^{real}, a_t^{real}\}_{t=1}^T) - \frac{1}{|Q|} \sum_{Q} ELBO(\{o_t^{sim}, a_t^{sim}\}_{t=1}^T)]$
    $r \leftarrow r * c$
  while $\sigma > \sigma^{'}$
  Train q_ϕ, update ϕ with $(\{o_t^{sim}, a_t^{sim}\}_{t=1}^T, \epsilon_{sim}^{rand})$ in D_q
  Update $\Xi_{k+1} \sim U(\mu, r)$ using SPM q_ϕ with:
  $\mu_{k+1} = \mu_k + \alpha q_\phi(\{o_t^{real}, a_t^{real}\}_{t=1}^T, \mu_k)$
end for

IV. Experiments

In this section, we want to answer three questions by extensive simulation and real-world robot experiment: 1) Can our method tune the simulation parameter for bipedal robot locomotion task correctly and safely? 2) Can our method outperform baseline methods in sim-to-real problem? 3) Can our method overcome the reality gap better than domain randomization method?

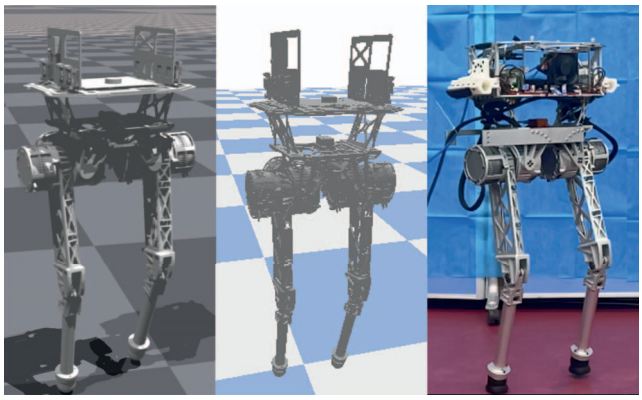


Fig. 2. BITeno robot in different environments. The environments from left to right are 1. Isaac Gym. 2. Pybullet. 3. Real robot.

A. Implementation Details

1) BITeno robot: The robot platform BITeno weighs 15.5kg. Its height is 0.95m, and the length of each leg is 0.65m. BITeno has totally 6 DoFs, and each leg contains abductor/adduction, hip, knee joint. The robot in different environments is shown in Fig. 2. The parameters we randomize are 7-dimensional, including the feet friction ($\mathbb{R}^1$) and mass of each link of robot ($\mathbb{R}^6$).

2) Policy learning and deployment: While theoretically our method can be applied to different tasks, in this paper we focus on bipedal robot locomotion task. So the policy π takes observations $o_t \in \mathbb{R}^{40}$ as input and output robot action $a_t \in \mathbb{R}^6$. o_t is defined as: $o_t = [v_t, c_t, \omega_t, g_t, \hat{q}_t, \hat{\dot{q}}_t, a_{t-1}, e_{t-1}, clock_t]^T$, where $v_t, c_t, \omega_t, g_t, \hat{q}_t, \hat{\dot{q}}_t, a_{t-1}, e_{t-1}, clock_t$ are linear velocity, velocity command, angular velocity, projected gravity vector, joint angle, joint angular velocity, previous action, previous joint error and clock vector respectively. The reward settings are shown Table I in Appendix. Our algorithm can work with any RL algorithm and we use Proximal Policy Optimization (PPO) in this paper. For the policy learning, we use Isaac Gym, a PhysX engine based simulator that is tailored for massive parallel training with GPU acceleration. Our code is extensively based on [20]. During training and deployment, the learned policy produces action commands at 100 Hz, and the commands are mapped to torques by joint level PD controller at 1000Hz.

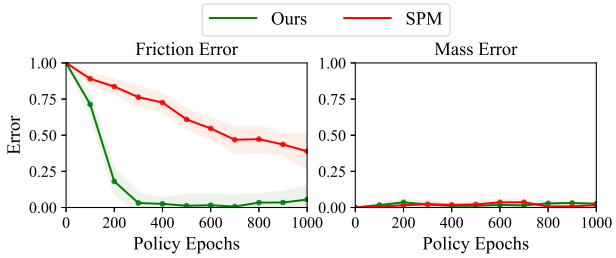
3) DVM learning: In the encoder end, to capture the features of robot locomotion data in both time and dynamic dimension, the sequence of observation and action tuples (o_t, a_t) are stacked and encoded by 4-layer convolutional network. A symmetric decoder is used in the other end of the network. To better learn the locomotion data distribution in a relatively long time span, we stack 50 consecutive tuples as input during training.

Fig. 3. Performance of the bipedal robot locomotion sim-to-sim experiment. We set Issac Gym as source environment and both misparameterized Issac Gym and Pybullet as target environment and report the rewards after transfer. From top to bottom, the four results are from experiments 1. Randomizing friction in Issac Gym. 2. Randomizing robot link mass in Issac Gym. 3. Randomizing parameters in Pybullet. 4. Weakened robot motor in Pybullet. Shown as dots (same in the following figures), the parameters are optimized every 100 epochs. Rewards are evaluated in target environment every 50 epochs. Each experiment is repeated over 3 random seeds. Note that for the some cases, the DR method does not converge. So we report that the converged rewards after 1500 epochs of DR methods are around 60.0, 5.0, 25.0, 20.0 respectively from top to bottom.

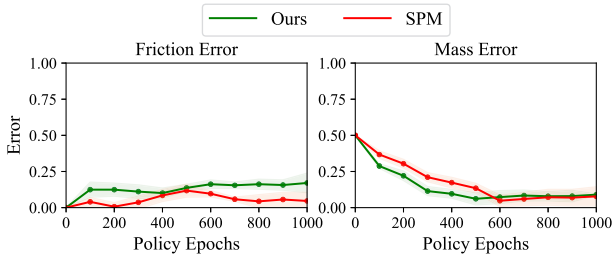
4) SPM learning: SPM and VAE are trained on same dataset. First the sequence of tuples (o_t, a_t) and parameters are flattened and encoded by 3-layer MLP encoder respectively. Note that we also use 50 consecutive tuples here. The labels L are computed from the true parameters ξ_{sim} and randomly sampled parameters ξ_{pred} by $L = \xi_{sim} > \xi_{pred} \in (0, 1)^7$. Then the encoded features of both trajectory and ξ_{pred} are concatenated and fed into separate 2-layer MLPs to train using logistic regression.

B. Sim-to-sim Experiment

For sim-to-sim experiment, we evaluate our method by training in Issac Gym environment as source environment, and transferred to a different target simulated environment on bipedal robot locomotion task. Because different simulated environments differ in physical properties, this experiment



(a) Parameter Error in Friction Randomization Experiment



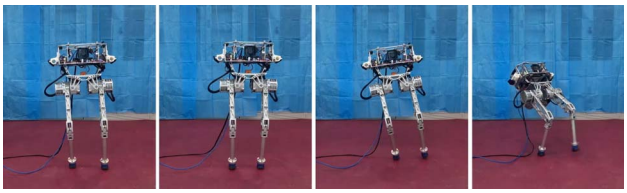
(b) Parameter Error in Mass Randomization Experiment

Fig. 4. Error of parameters during the parameter optimization. (a) shows parameters errors in Isaac Gym friction randomization transfer experiment, and (b) shows parameters errors in Isaac Gym mass randomization transfer experiment.

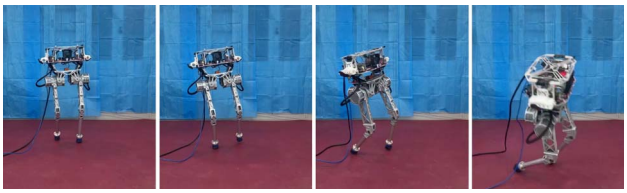
can be part of the proof of the effectiveness of our method. In this part, we use two different simulation environments as target environment. The first is Isaac Gym using different parameter distribution, as a misparameterized simulation environment; We also use Pybullet, a Bullet engine based simulator, as another target environment.

For each experiment, we choose one set of parameters as target environment parameters, while initialize the simulation with different system parameter settings, to create a gap between two environments. Here we set the parameters in misparameterized environment as $0.5X$ or $1.5X$. We take two methods as baselines: domain randomization and original SPM method. For both proposed method and baseline SPM method, the range r of Ξ are initialized as 0.2. Similarly, the range of randomized parameter domain randomization is initialized as 0.4. For better comparison, the performance of the policy directly trained in correctly parameterized environment is also shown and named as pseudo-real. In Fig. 3, we report the performance of transferred policy trained with previously introduced settings in target environment. Our method shows better or comparable performance to both baseline methods. Meanwhile, compared to SPM, our method demonstrate a reduction in the oscillation of rewards. The results suggest our method can automate the simulation parameters tuning to favor policy transfer, and introducing DVM effectively increases safety and efficiency of tuning process.

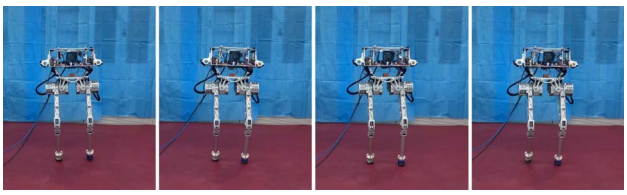
We also test the ability of our method to overcome the reality gap by absorbing the difference caused by other physical properties. Here we evaluate if the weakened power of actuator, a property intuitively related to the mass of robot, can be handled by the proposed method. We simulate the



(a) 0 Epoch Policy transfer Performance



(b) DR 1000 Epochs Policy transfer



(c) Ours 1000 Epochs Policy transfer

Fig. 5. Process of real robot locomotion experiment. (a) is the result of initial pretrained policy. (b) is the result of policy trained after 1000 Epochs using domain randomization. (c) is the result of policy trained after 1000 Epochs using our method (5 iterations of tuning). Our method can achieve stable movement while the pretrained policy and DR method both failed.

weakened power by setting a torque scaling factor, $s = 0.75$, and the actual torque applied to the simulated robot is $\tau = s\tau^*$, where τ^* is the calculated torque mapped by PD controller. The result is shown in Fig. 3 and illustrates the capability of our method to handle difference in other related physical properties such as weakened motor.

Intuitively, less difference in parameter leads to less environmental gap within the same simulated environment. So in Fig. 4 we show the system parameter errors during tuning in experiments using same simulated environment as both source and target environments (Isaac Gym friction randomization and mass randomization transfer experiment) to prove the effectiveness of our method. The results indicate our method can shift the parameter to the optimal parameter setting more efficiently than the original SPM method.

C. Sim-to-real Experiment

In sim-to-real transfer experiment, we apply our method to evaluate the ability to cross sim-to-real gap on real bipedal BITeno robot locomotion task. Similar to the setup in sim-to-sim transfer experiment, the initial policy is trained in a misparametrized simulated environment, which causes a large sim-to-real gap. For the convenience of experiment, we set the mass of simulated robot to $0.5X$ of the mass in the original robot model, and the initial policy trained in this parameter setup fails because of the model mismatch. We compare our method to the Domain randomization baseline and policy trained using the original robot model. The process of experiment is shown in Fig. 5. The results are presented in

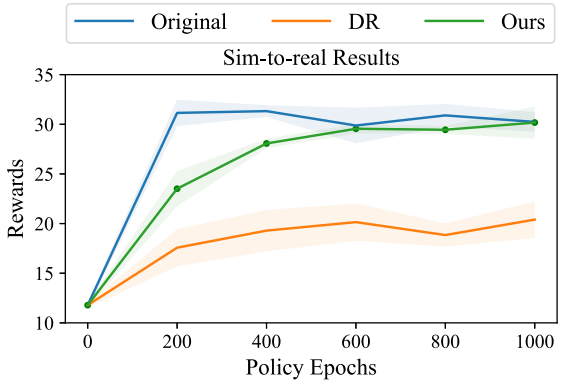


Fig. 6. Performance of the bipedal robot locomotion sim-to-real experiment. In sim-to-real experiment, parameters are optimized every 200 epochs, and policy is evaluated every 200 epochs.

Fig. 6, and indicate our method can successfully overcome the sim-to-real gap and outperform domain randomization. From the result we see the domain randomization method can not cross sim-to-real gap, while our method exhibits performance comparable to original model after sim-to-real tuning.

V. CONCLUSIONS

In this paper, we present a framework to tune simulation parameters using real-world robot trajectory to bridge the sim-to-real gap for bipedal locomotion task. We use SPM to iteratively auto-tune the parameter distribution, and extend the framework to account for unreliable parameter shift problem. This is done by formulating the problem as an OoD problem and introducing a VAE-based data verification model to identify unreliable dataset. We evaluate our method on bipedal BITeno robot in both sim-to-sim and sim-to-real experiments. The results indicate our method show its capability of crossing sim-to-real gap, and also earn gains in safety and efficiency.

The limitation of our method is it has only been tested on robot proprioceptive data. For future work, we can integrate exteroceptive information such as visual information in our framework.

APPENDIX

The complete reward settings for policy training are shown in Table I.

TABLE I

REWARD FUNCTIONS

Reward	Formula	Weight
Linear command tracking	$exp(-4(v_{xy}^{cmd} - v_{xy})^2)$	2.0
Angular command tracking	$exp(-4(\omega_{yaw}^{cmd} - \omega_{yaw})^2)$	1.5
Linear velocity (z)	v_z^2	-1.25
Angular velocity (xy)	ω_{yaw}^2	-0.1
Torque	τ^2	$-5.0 * 10^{-6}$
Orientation	$\|g\|^2$	-10.0
Joint acceleration	$\ddot{q}$	$-2.0 * 10^{-7}$
Action rate	$(a_t - a_{t-1})^2$	-0.1
Joint default postion	$\sum_j (q_j - q_j^{stand})^2$	-0.35
Raibert heuristics	$\sum_j (q_j - q_j^*)^2$	-1.25

REFERENCES

[1] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," in *Conference on Robot Learning*, pp. 1110–1120, PMLR, 2021.

[2] Y. Du, O. Watkins, T. Darrell, P. Abbeel, and D. Pathak, "Auto-tuned sim-to-real transfer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1290–1296, IEEE, 2021.

[3] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *arXiv preprint arXiv:2110.11334*, 2021.

[4] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[5] 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)*, pp. 3803–3810, IEEE, 2018.

[6] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[7] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[8] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[9] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979, IEEE, 2019.

[10] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Conference on Robot Learning*, pp. 1162–1176, PMLR, 2020.

[11] F. Ramos, R. C. Possas, and D. Fox, "Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators," *arXiv preprint arXiv:1906.01728*, 2019.

[12] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "Neuralsim: Augmenting differentiable simulators with neural networks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9474–9481, IEEE, 2021.

[13] K. Ota, D. K. Jha, D. Romeres, J. van Baar, K. A. Smith, T. Semitsu, T. Oiki, A. Sullivan, D. Nikovski, and J. B. Tenenbaum, "Data-efficient learning for complex and real-time physical problem solving using augmented simulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4241–4248, 2021.

[14] N. Sontakke, H. Chae, S. Lee, T. Huang, D. W. Hong, and S. Hal, "Residual physics learning and system identification for sim-to-real transfer of policies on buoyancy assisted legged robots," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 392–399, IEEE, 2023.

[15] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," *arXiv preprint arXiv:2105.08328*, 2021.

[16] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7309–7315, IEEE, 2021.

[17] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint arXiv:1702.02453*, 2017.

[18] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, "Sim-to-real transfer for biped locomotion," in *2019 ieee/rsj international conference on intelligent robots and systems (iros)*, pp. 3503–3510, IEEE, 2019.

[19] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik, "Adapting rapid motor adaptation for bipedal robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1161–1168, IEEE, 2022.

[20] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*, pp. 91–100, PMLR, 2022.