EasyChair Preprint

№ 10757

Rapid Bipedal Robot Adaptation via Discriminative Internal Model

Zhibo Zhou

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

August 21, 2023

Rapid Bipedal Robot Adaptation via Discriminative Internal Model

Zhibo Zhou

***Abstract*— —Reinforcement learning (RL) methods play a crucial role in training bipedal robot locomotion. However, there exists a practical challenge in that well-trained robot policies cannot be directly deployed to different robot dynamics, due to the dynamics gap between the training and the application environment, making the policies inflexible for application in various robot tasks. To address this issue, we propose a rapid adaption framework, named Discriminative Internal Model (DIM), which attempts to accelerate the adaption efficiency of RL agents and improve the generalization ability in various dynamic environments. Specifically, DIM first learns a parameterized dynamics model, called internal model (IM), in the training environment. In the adaptation phase, the learned IM uses a small number of transitions to quickly adapt to the new environment. The "fine-tuned" IM can simulate rollouts close to the new environment's distribution to speed up policy adaptation. To avoid generating unreliable rollouts that degrade the performance of the policy, we further proposed a state discriminator. It evaluates the reliability of the internal model in each state to determine the number of augmentation rollouts at that state. To demonstrate the effectiveness of the DIM framework, we conduct experiments on a bipedal robot for dynamics transfer and sim-to-real transfer tasks. Extensive experimental evaluations on bipedal locomotion demonstrate that the proposed DIM outperforms the state-of-the-art model-free RL methods.**

I. INTRODUCTION

Designing robust walking policies for bipedal robots based on classical control requires precise robot dynamics models and human expertise to plan motion trajectories. [1], [2]. Recent model-free reinforcement learning (MFRL) methods have demonstrated promising results on both simulation and physical robots [3], [4]. MFRL could optimize the control policies directly without modeling the environmental dynamics.

However, a notorious weakness is that the MFRL policy is limited to specified dynamics and tasks. The control policies trained in predefined dynamics can not be directly transferred to the target robot [5], [6]. The performance of the policies trained in the source would drop significantly in the target due to the gap between the environments [7].

Existing approaches for tackling the transfer problems in bipedal robotic tasks include dynamics/domain randomization (DR) and system identification (SI) [4], [8] System identification [6], [9].Nonetheless, neither of the two approaches above guarantees a realistic enough simulator. For example, external noise (e.g. temperature, wear, and tear) can cause various physical parameters in the same robot, making it difficult to build accurate simulators. As a result, these methods will fail when policies are deployed in complex environments with more extreme variances, such as large changes in the center of mass (CoM) or friction. And it is impractical to simulate all robot dynamics situations during training.

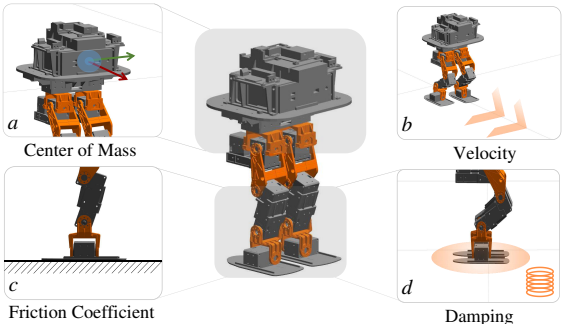

Fig. 1. We set 4 different tasks by varying dynamics parameters. The **center of mass(CoM)** of the robot. The forward **b) velocity** of the robot. The **c) friction** coefficient and the **damping** coefficient of the feet.

In this work, A novel Discriminative Internal Model (DIM) is proposed to address the above-mentioned issues. In the training phase, an internal model is learned while training an MFRL policy, which aims to estimate the state transition of the robot dynamics. In the adaptation phase, the internal model performs transition generation to accelerate MFRL policy adaptation. Intuitively, this is a combination of MFRL and model-based RL (MBRL). However, existing model-based generation usually ignores the reliability of the learned, and applies the model indiscriminately to all states [10], [11]. To mitigate this problem, our DIM employs a more intelligent and efficient way for data augmentation, it evaluates the reliability of the internal model in a state to determine the amount of augmentation data at that state. The contributions of this method can be summarized as follows.

- We developed a framework that combines MFRL and MBRL for transfer learning. To our knowledge, this is the first application of the combination of MBRL and MFRL to a bipedal robot.
- An internal model (IM) is learned and applied to look forward from the current state and generate a series of trajectories to improve policy learning when adapting to new dynamics. DIM first predicts future transitions through a learned environment/dynamics model. The predicted trajectories are mixed into real interaction data to improve policy learning, which can be regarded as a kind of data augmentation.

[1]Zhibo Zhou is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China. zhibozhou09@gmail.com

- We build bipedal robots in PyBullet[1] and Gazebo[2] simulation, respectively, to verify the generalization performance of bipedal locomotion policies, including task generalization and dynamics generalization. Compared with related algorithms, the proposed method achieves state-of-the-art performance, demonstrating the superiority of the proposed model for adapting to a variety of unseen scenarios including extremely low friction situations, and unstable CoM situations.

II. Related work

A. Policy transfer in robotic tasks

Policy transfer for robots aims to study transferring the policies from training environments to testing environments [12], [13]. Existing work can be divided into two categories, one is to improve the robustness of the policy so that it can be applied to new environments, and the other is to train a policy that can adapt quickly.

Leaning a robust policy. Dynamics Randomization (DR) is one of the most popular approaches to improve the robustness of the RL policies [14], [15]. It aims to conduct variety environments or diverse robot dynamics to train the agent for a more robust policy [4], [8], which has achieved promising results in manipulation [16], legged locomotion [17] and visual navigation [18]. Besides, adversarial RL methods also focus on improving the robustness of the model. They train an adversarial controller to generate more difficult and diverse environments. The agent is trained in the environment with disturbances that are generated by the destabilizing adversary [19]. However, it is impractical to simulate or generate all robot dynamics situations during training, which results in the agent's still limited adaptation to the new environment.

Learning to adapt quickly. Instead of learning a robust policy, a different perspective is transfer learning (TL) methods. TL aims at improving the adaptation ability of target learners on target domains by transferring the knowledge acquired in source domains. It provides a feasible solution for online adaptation. Meta-learning focus on "learn-to-learn", which learns a policy with the ability to adapt quickly. K. Arndt et al. [20] let a meta policy search in a latent space for faster adaptation. In contrast to meta-learning, X. B. Penet al. [21] addressed the domain gap through a domain adaptation technique that maps the features of source and target domains to the same space.

Nevertheless, transferring an RL policy is not feasible when there is a large gap between training and testing environments [22], [23]. This is because learning MFRL policies through bootstrapping is unstable [24]. The proposed DIM method focuses on transferring the dynamics model. Learning DIM through supervised learning is more stable than bootstrapping. Thus, transferring the dynamics model is more feasible than transferring the policy directly.

[1]https://pybullet.org
[2]https://gazebosim.org

B. Model-based RL

In contrast to the model-free RL, model-based approaches utilize an approximate dynamics model to mitigate the problem of sample inefficiency [25], [26]. Such a dynamics model can then be used to control through planning or to improve the data efficiency by performing transition augmentation.

Planning. Model predictive control (MPC) [27] achieves great success on numerous robotic tasks by selecting the best action according to model planning trajectories [28]. [29] and [30] construct implicit plans to obtain rollout trajectories by using a learned environment/dynamics model and a model-free policy. The rollout trajectories are then simply encoded into policy networks as additional information to improve MFRL. However, the planning methods tend to fall into a local optimal due to the cumulative prediction error.

Data augmentation. Another usage of model-based methods is to perform data augmentation for model-free algorithms. The underlying idea is to speed up policy training by using model-generated data [31]. Model-based value expansion(MBVE) generates a fixed horizon to update the Q-function [32]. Based on MBVE, [33] further shows that using the rollouts with high model accuracy is more effective than using a fixed horizon. To promote the model generalization, the context-based methods are employed to train the dynamics model and proved to be efficient [11], [34]. Our DIM method makes use of the advantages of context-based methods that capture the features from history trajectory by Recurrent Neural Networks (RNN) [35]. Moreover, we consider model accuracy in the target environment when performing the data augmentation.

III. Preliminaries

In this section, we will briefly introduce the definition and notation of the RL framework. Specifically, bipedal locomotion is formulated as a Markov Decision Process (MDP) problem. Standard reinforcement learning includes an agent interacting with an environment E and receiving a reward r every time step t. MDP can be denoted as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is the transition probability function, and $\mathcal{R}$ is the reward function, $\gamma \in (0, 1]$ is the discount factor. The objective of reinforcement learning is to optimize the policy π by maximizing the expected return from the initial state. Therefore, the basic policy gradient can be denoted as:

$$\min_{\theta} \nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} \mathcal{R}(s_{t'}, a_{t'}, s_{t'+1})] \tag{1}$$

where $\tau = (s_0, a_0, s_1, ..., s_{T+1})$ is the finite length trajectories collect with policy π_θ. The return of action can be simplified to consider only the rewards after the agent takes the action, not the rewards before the action. To reduce the variance in the sample estimate for the policy gradient, the advantage function is introduced as an alternative optimiza-

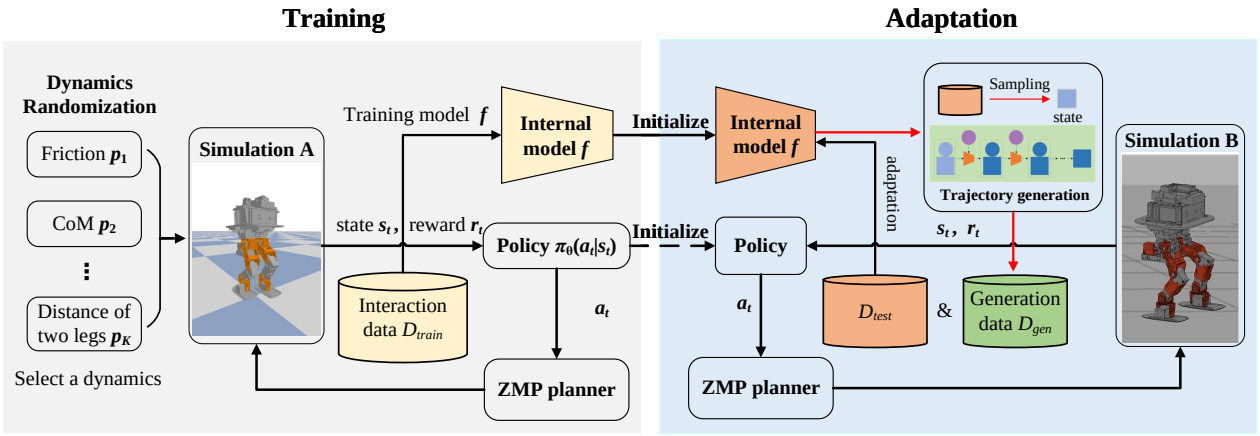

Fig. 2. Overview of the proposed method. In the training phase, the policy π and the internal model f are trained simultaneously in the dynamics of randomized environments. During adaptation, the internal model and the policy are both initialized with the pre-trained model. The policy then utilizes the data from $\mathcal{D}_{test}$ combined with $\mathcal{D}_{gen}$ generated by the internal model for adaptation.

tion objective.

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A_\pi(s_t, a_t)] \quad (2)$$

The advantage of an action, defined by $A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$, describes how much better or worse it is than other actions on average. Q and V represent the state-action value and state value, respectively. The baseline policy used in this work, Proximal Policy Optimization (PPO) [36], is derived from Eq. 2.

IV. Method

The framework of our method is presented in Fig. 2. The agent observes the robot state s_t from the environment and feeds it to the actor to predict a control signal, which is a set of parameters of Zero Moment Point (ZMP) [1] controller. The ZMP controller then parses the control trajectory and sends it back to the robot. After that, the agent will receive a reward r_t from the environment. In the learning phase, we employ dynamic randomization to learn policies for various robot dynamics and learn an internal model that approximates the environment dynamics. In the adaptation phase, the internal model is continued to be trained and used to augment the state to generate virtual trajectories. Then the virtual data is mixed with real interaction data to train the policy. However, some bad augmentation data may be harmful to policy training. Therefore, we further propose to generate data in a discriminative manner for states.

A. Learning internal model

Learning a dynamics model through supervised learning is more stable than learning the value function through bootstrapping [24]. Thus transferring the dynamics model is more feasible than transferring the policy directly. In this work, we learn the environment dynamics through end-to-end neural networks. A recurrent network is introduced to capture the context of recent experiences. The intuition is that the dynamics information underlying transitions can be captured from interaction trajectories.

Learning a dynamic model that approximates the environment dynamics is the core of model-based reinforcement learning. The proposed DIM consists of a transition estimator, a reward estimator, and a "done" estimator, parameterized by θ^ϕ, θ^φ and θ^ξ, respectively. As illustrated in Figure 3. The transition module receives the state s_t, the action a_t, as well as the latent h_t encoded from the RNN network, to predict the next state $\hat{s}_{t+1}$. It can be formulated as:

$$\hat{s}_{t+1} = f_\theta^\phi(s_t,\ a_t,\ h_t) \quad (3)$$

The reward module estimates the state-action reward for executing the action a_t at state s_t. Similarly, the reward module receives the state s_t, the action a_t, and the latent vector h_t to predict the state-action reward $\hat{r}_t$, which can be formulated as:

$$\hat{r}_t = f_\theta^\varphi(s_t,\ a_t,\ h_t) \quad (4)$$

A complete environment model should be interactive, that is, given a policy, the agent can obtain complete rollouts by interacting with the environment model. Therefore, in addition to the transition module and reward module mentioned above, a "done" estimator is learned to predict whether the trajectory is terminal after executing the action a_t at state s_t.

$$\hat{d}_t = f_\theta^\xi(s_t,\ a_t,\ h_t) \quad (5)$$

where $\hat{d}_t$ is the predicted "done" flag. The environment model is trained with tuples $(s_t,\ a_t,\ r_t,\ s_{t+1},\ d_t)$ sampled from experience data, which can be easily obtained through interaction. We optimize the environment model by minimizing the following loss functions:

$$L(\theta^\phi) = ||s_{t+1} - f_\theta^\phi(s_t,\ a_t,\ h_t)||^2 \quad (6)$$

$$L(\theta^\varphi) = ||r_t - f_\theta^\varphi(s_t,\ a_t, s_{t+1})||^2 \quad (7)$$

$$L(\theta^\xi) = ||d_t - f_\theta^\xi(s_t,\ a_t, s_{t+1})||^2 \quad (8)$$

With the environment model, for a given state s_t, the agent has the ability to simulate the subsequent trajectory under the current policy. In this work, the predicted trajectories are mixed into real interaction data to improve policy learning, which can be regarded as a kind of data augmentation.

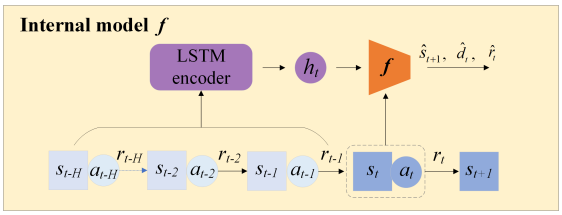

Fig. 3. Illustration of the internal model. The internal model **f** accepts the state s_t, action a_t and a latent vector h_t at time step t to predict the next state $\hat{s}_{t+1}$, reward r_t and determine whether it is "done".

B. Accelerating adaptation with DIM

The internal model can be applied to augment the state, which can then be used for planning or to generate more transitions. The proposed method falls into the latter category, focusing on leveraging the learned environment model to generate a series of transitions that are close to the new tasks domain to accelerate policy adaptation.

The DIM is first trained in the training task to obtain an initialized model. For new environments/tasks, with the context encoder, the DIM can quickly adapt with few transitions. This allows the agent to interact with the internal model to obtain virtual transitions that are similar to the new environment. Assuming that the agent receives the state s_t and takes action at time step t, according to Equation (3), (4) and (5), the internal model will predict the next state $\hat{s}_{t+1}$ and the corresponding reward $\hat{r}_t$, as well as the "done" flag. The augmentation process can be repeated based on the new predicted state s_{t+1}. The augmentation process at every time step can be formulated as:

$$\begin{aligned} \hat{s}_{t+k} &= f_\theta^\phi(\hat{s}_{t+k-1},\ \hat{a}_{t+k-1},\ \hat{h}_{t+k-1}) \\ \hat{r}_{t+k} &= f_\theta^\varphi(\hat{s}_{t+k},\ \hat{a}_{t+k},\ \hat{h}_{t+k}) \\ \hat{d}_{t+k} &= f_\theta^\xi(\hat{s}_{t+k},\ \hat{a}_{t+k},\ \hat{h}_{t+k}) \end{aligned} \tag{9}$$

where k is the augmentation depth. As the augmentation depth increases, the errors will increase simultaneously.

C. State discriminator

Augmentation for all states is unreliable, because the dynamic model may have inaccurate predictions for some states. The inaccurate augmentation may drop the performance of the agent. To alleviate this problem, we further propose to discriminate the model reliability of the states. Then we sample states with better model reliability with a higher probability to perform state augmentation.

Specifically, we calculate the prediction error (MSE in this work) $\|e(s_t)\|^2$ as the reliability criterion for each state s_t in the buffer. The sampling probability is defined as:

$$p_i = Softmax(-\|e(s_t)\|^2)$$

Intuitively, a lower prediction error means the internal model is more "familiar" with the state, and the model can generate more accurate transitions in that state.

TABLE I

DEFINITION OF THE REWARD FUNCTIONS

Reward term	formulation	weight
Step reward r_s	$\|q_t - q_{t-1}\|^2 cos\theta$	$w_s = 100.0$
Effort reward r_e	$-\sum \tau^2$	$w_e = 0.25$
Orientation reward r_o	$-\sum(min(\|\alpha_i\|, 15))$	$w_o = 0.05$
Height reward r_h	$-\|h_t - h_0\|$	$w_h = 20.0$
Fall reward r_f	-50 if robot falls down else 0	$w_f = 1.0$

Algorithm 1 DIM adaptation

Input: Initial policy:π;
Pre-train IM: f_θ;
Augmentation depth k;
Data nums: N;
Output: Adaptation policy: π
while $epoch \leq max\_epoch$ **do**
Collect trajectories $\{\tau_i\}$ using π.
Train f_θ with $\{\tau_i\}$ by Eq. 6.
For each trajectory τ_i calculate sampling probability:

$$P_i = Softmax(-\|e(s_t)\|^2), s_t \in \tau_i$$

Sample N states $s \sim P_i$
Generate k-depth trajectories $\{\hat{\tau}\}$ using f_θ.
Update the policy π using $\{\hat{\tau}\}$
end while

V. EXPERIMENTAL SETUP

In this section, we present the experimental setup and implementation details of the proposed framework. We first describe the bipedal robot in Sec. V-A. We then introduce the simulation environment and the definition of the tasks in Sec. V-B. Finally, the evaluation metrics and training details are elaborated in the V-C and Sec. V-D, respectively.

A. Robot setup

To verify the effectiveness of the proposed method, we conduct our simulator on Gazebo and PyBullet to support the training and adaptation validation of bipedal locomotion.

Robot. The bipedal robot contains 10 degrees of freedom (DoF), and each leg has 5 DoF. They correspond to five joints hip roll, hip pitch, knee, ankle roll, and ankle pitch. The CoM is maintained at the center of the pelvis.

State and Action Space. The state consists of joint positions, joint velocities, body angular velocity, and body angular acceleration, which is a 20-dimension vector. The action is a set of parameters of ZMP [1] controller [3], which contains the gait length, the height of the feet raise, the amplitude of the body swings (in x and z directions) and the gait period. The ZMP controller outputs the planning trajectory to control the robot at a frequency of 240 Hz, while the frequency of the RL controller is 8Hz.

Reward function The reward functions employed in this work are shown in Tab. I. The robot position at step t is represented by $q_t \in \mathbb{R}^3$. θ is the angle of the $q_t - q_{t-1}$ and the target direction d. τ in the effort reward is the joint

[3] https://github.com/ROBOTIS-GIT/ROBOTIS-OP3/

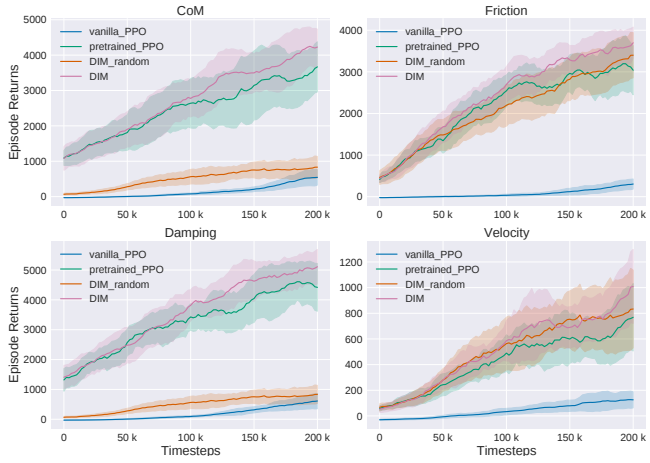

Fig. 4. Adaptation training curves on 4 tasks. The solid line represents the mean and the shadow represents the STD across 5 random seeds.

torque. α_i in the orientation reward is the robot orientation in the Euler angle. i represents the rotation direction, which includes roll, pitch, and yaw. h_0 in the height reward is the desired height of the robot. Therefore, the reward function is defined as the sum of these sub-rewards: $r_t = w_s r_s + w_e r_e + w_o r_o + w_h r_h + w_f r_f$.

B. Environments and Tasks

We build the bipedal robot simulator in PyBullet and define 4 tasks to evaluate the adaptation performance of different approaches, as shown in Figure 1. We randomize the dynamics parameters and divide them into two disjoint groups, one for the training and the other for the adaptation. The dynamics parameters include CoM, friction coefficient, and damping coefficient. Moreover, we define different walking velocities in the training and adaptation environments to verify the task transfer ability of policy. More implementation details are described below.

CoM. The CoM of the robot is randomized in the x-axis and y-axis directions, respectively. Specifically, we change the CoM with a random offset distance in these 2 directions, denoted by COM_x and COM_y. In implementation, $\text{COM}_x \in [-3.0, 3.0]$ and $\text{COM}_y \in [-5.0, 5.0]$. The unit is m. **Friction:** The range of friction coefficient in training is $[0.2, 0.8]$, while $[0.1, 0.2]$ in adaptation. **Damping:** The damping of the ground, is set as $[6, 12]$ in training and $[5, 6] \cup [12, 13]$ in adaptation. The unit is N/m. **Velocity**.The walking velocity of the bipedal robot. The range of velocity is $[60, 75]$ in training and $[75, 85]$ in adaptation. The unit is cm/s. For all the tasks, there is no intersection between the training and adaptation sets.

C. Evaluation Metrics

Since the objective of the RL agent is to maximize cumulative reward, the average episode reward is used to evaluate the performance of adaptation.

$$R = \frac{1}{N} \sum_{t=0}^{T} r(s_t, a_t)$$

where N is the number of test episodes. We also introduce success rates to compare the varying models. An episode is considered a success if the robot can walk stable for 2 minutes and walk forward 5 meters from the starting point. The success rate is calculated by:

$$SR = \frac{\mathcal{X}(step > L \cap len > D)}{N} \times 100\%$$

where $\mathcal{X}$ is the indicative function and L is the time step threshold. len is the walking distance and the D is the distance threshold.

D. Training details

Internal Model. The internal model consists of 3 fully connected layers. The concatenating vector (h_t, s_t, a_t) passes through 3 fully connected layers of size $(33, 300)$, $(300, 300)$ and $(300, 22)$ to predict next state $\hat{s}_{t+1}$, reward $\hat{r}_t$ and "done" flag $\hat{d}_t$. The context encoder consists of a LSTM [35] module. The input trajectory length and the latent size of LSTM are 8 and 64, respectively. Then a fully connected layer outputs the latent code to 8. The learning rate of training is set as $1e-3$.

MFRL training. The policy and value network both contain 2 hidden layers of 64 units. The learning rate of policy and value function are all set as $3e-4$. The batch size in our experiment is set as 64.

VI. Experimental Results and Analysis

This section qualitatively and quantitatively analyzes the adaptation performance of the proposed DIM with vanilla PPO [36], pre-trained PPO, and DIM-random.

- **Vanilla PPO** is training the PPO policy from scratch in adaptation environments.
- **pretrained PPO** is training PPO with a pre-trained policy that is trained in the training environments.
- **DIM** is the proposed method. The internal model and the policy are both pre-trained. In the adaptation phase, DIM is employed to generate transitions to improve adaptation efficiency.
- **DIM-random** is similar to DIM. The only difference is that the pre-trained internal model is replaced with a randomly initialized model. We consider this setting to verify the effectiveness of the prior knowledge acquired by the internal model on the training task.

A. Dynamics Transfer Results

The adaptation results are shown in Figure 1. Our method significantly outperforms the baseline method in all tasks. Especially for the velocity task, our method achieves a mean reward of 928.89, while the vanilla PPO only gets 732.17. It demonstrates that DIM generation is efficient to accelerate the adaptation of RL agent.

It can be seen that vanilla PPO has poor performance in all tasks, while the pre-trained PPO and DIM perform better. This indicates that pre-training is essential for the agent to gain task-relevant knowledge. It is also worth noting that DIM-random achieves competitive results in CoM and Friction tasks. Although the random initialized internal model

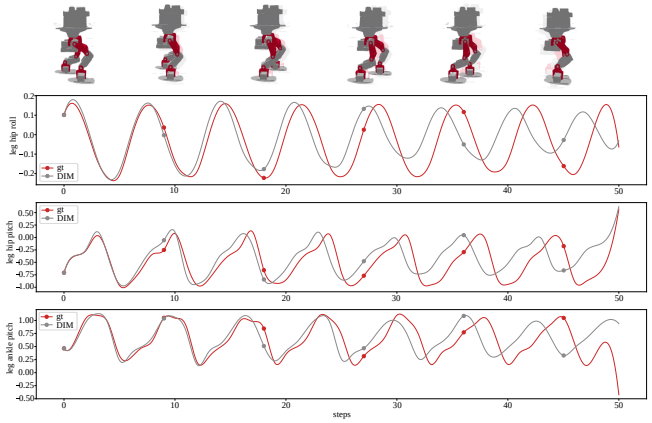

Fig. 5. The trajectory predicted by the internal model. The first row illustrates the robot pose. The 3 curves show the joint trajectories of the hip roll, hip pitch, and ankle pitch on the left leg. The x-axis represents the time steps.

TABLE II

THE COMPARISON OF SUCCESS RATES ON 4 TASKS.

	Friction	Velocity	Damping	CoM
vanilla PPO	68%	18%	92%	81%
DIM-random	72%	20%	30%	32%
DIM w/o SD	80%	24%	93%	80%
DIM	**81%**	**24%**	**98%**	**84%**

may be imperfect, the proposed SD is helpful to reduce the generation of inaccurate transitions. This prevents the policy from being attacked by these transitions.

We also measure the success rates of different methods, as shown in Tab. II. In practice, we calculate the success rate by testing 100 episodes on each task using the best-trained model. DIM yields a higher success rate than other methods, which further proves the effectiveness of the proposed DIM.

B. Ablation studies

In this section, we conduct experiments to evaluate the importance of the key components of our algorithm, including We also validate the state discriminator, the prediction accuracy of the internal model, and the number of augmentation transitions.

The prediction accuracy of the internal model. To verify the performance of the internal model, we visualize the predicted future trajectory (contains 50 predicted steps), as shown in Fig. 5. The predicted transitions can be translated to the robot poses, as shown in the first row of Fig. 5, the shadow represents the predicted robot pose while the solid represents the actual pose. The corresponding joint trajectories are also shown in Fig. 5. For brevity, we have chosen only three joints. It shows that the trajectory prediction of the internal model is accurate within the first 10 steps. As augmentation depth increases, the prediction error will gradually increase. **The effect of the augmentation ratio.** To evaluate the number of augmentation transitions, we set different ratios of the augmentation and evaluate the episodic returns. The results are shown in Tab. III. It can be seen that the best results on 3 tasks are obtained when the ratio is set as 1. We believe that too much augmentation may

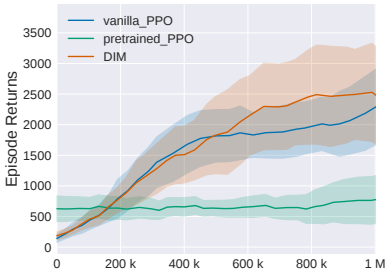

Fig. 6. Adaptation training curves for Gazebo to PyBullet transfer. We report the mean episode returns across 5 random seeds.

TABLE III

THE REWARD FOR DIFFERENT AUGMENTATION RATIOS ON FOUR TASKS.

	tasks			
Ratio	**Velocity**	**Damping**	**Friction**	**CoM**
0×	1132	5710	4201	4075
0.5×	**1467**	6058	4257	4449
1×	1283	**6070**	**4849**	**5208**
2×	1038	5676	4558	4868
4×	1431	5772	4239	4859

lead to a lot of similar transitions, which are redundancy for the policy.

C. Gazebo to PyBullet

To evaluate the generalization of the DIM model. We also build a robot using Gazebo simulation to validate the transfer performance across different simulators. The state and action space are consistent with the PyBullet simulator. The biggest difference is that Gazebo uses the ODE physics engine, while PyBullet uses the Bullet.

Results. Figure 6 illustrates the effectiveness of the proposed method across different simulation environments. The large simulation gap makes policy trained on the gazebo suffer from catastrophic failure when directly adapting to the PyBullet environment. The policy initialized by the pre-train model even underperforms the randomly initialized policy. This issue also occurs in [22], [23]. Thus, we directly use the randomly initialized policy in the adaptation phase.

VII. CONCLUSIONS

We presented an adaptable internal model equipped with a state discriminator to accelerate the policy adaptation for a bipedal robot. In this work, we propose Discriminative Internal Model (DIM), to accelerate the adaption efficiency of MFRL agents and improve the generalization ability in various dynamics environments. To avoid bad augmentation, we further propose a state discriminator that evaluates the reliability of the internal model in a state to determine the amount of augmentation data at that state. We demonstrate the adaptation performance on the bipedal robot with various dynamics parameters. Experimental results demonstrate that the DIM enables the agents to rapidly adapt across different dynamics and tasks. Our future work will focus on utilizing the DIM for sim-to-real adaptation.

REFERENCES

[1] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to humanoid robotics*. Springer, 2014, vol. 101.

[2] A. Ames, J. Rogers, F. Hammond III, Y. Wardi, A. Thomaz *et al.*, "Dynamic humanoid locomotion: Hybrid zero dynamics based gait optimization via direct collocation methods," Ph.D. dissertation, Georgia Institute of Technology, 2016.

[3] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Proceedings of the Conference on Robot Learning*, 2020, pp. 317–329.

[4] 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.

[5] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1241–1246.

[6] 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)*, 2019, pp. 3503–3510.

[7] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[8] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2811–2817.

[9] M. Kaspar, J. D. M. Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4383–4388.

[10] S. Racanière, T. Weber, D. Reichert, L. Buesing, A. Guez, D. Jimenez Rezende, A. Puigdomènech Badia, O. Vinyals, N. Heess, Y. Li *et al.*, "Imagination-augmented agents for deep reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[11] K. Lee, Y. Seo, S. H. Lee, H. Lee, and J. Shin, "Context-aware dynamics model for generalization in model-based reinforcement learning." *arXiv: Learning*, 2020.

[12] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," *ieee symposium series on computational intelligence*, 2020.

[13] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250.

[14] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning*, 2017, pp. 334–343.

[15] A. Vandesompele, G. Urbain, H. Mahmud, F. Wyffels, and J. Dambre, "Body randomization reduces the sim-to-real gap for compliant quadruped locomotion," *Frontiers in neurorobotics*, 2019.

[16] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 334–343.

[17] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *robotics science and systems*, 2018.

[18] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[19] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning*, 2017, pp. 2817–2826.

[20] K. Arndt, M. Hazara, A. Ghadirzadeh, and V. Kyrki, "Meta reinforcement learning for sim-to-real domain adaptation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2725–2731.

[21] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv: Robotics*, 2020.

[22] M. Wulfmeier, I. Posner, and P. Abbeel, "Mutual alignment transfer learning," in *Conference on Robot Learning*, 2017, pp. 281–290.

[23] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342*, 2015.

[24] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," *arXiv preprint arXiv:2006.16712*, 2020.

[25] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *neural information processing systems*, 2019.

[26] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," *international conference on robotics and automation*, 2017.

[27] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[28] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv: Learning*, 2018.

[29] S. Racanière, T. Weber, D. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li *et al.*, "Imagination-augmented agents for deep reinforcement learning," in *Advances in neural information processing systems*, 2017, pp. 5690–5701.

[30] X. Wang, W. Xiong, H. Wang, and W. Yang Wang, "Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation," in *European Conference on Computer Vision*, 2018, pp. 37–53.

[31] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *International conference on machine learning*, 2018, pp. 1407–1416.

[32] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, "Model-based value estimation for efficient model-free reinforcement learning," *arXiv preprint arXiv:1803.00101*, 2018.

[33] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," *Advances in neural information processing systems*, 2018.

[34] Y. Seo, K. Lee, I. C. Gilaberte, T. Kurutach, J. Shin, and P. Abbeel, "Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning," *neural information processing systems*, 2020.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.