An Efficient Model-Based Approach on Learning Agile Motor Skills without Reinforcement

Haojie Shi[1,2*], Tingguang Li[1*], Qingxu Zhu[1], Jiapeng Sheng[1], Lei Han[1] and Max Q.-H. Meng[3†], *Fellow, IEEE*

***Abstract*— Learning-based methods have improved locomotion skills of quadruped robots through deep reinforcement learning. However, the sim-to-real gap and low sample efficiency still limit the skill transfer. To address this issue, we propose an efficient model-based learning framework that combines a world model with a policy network. We train a differentiable world model to predict future states and use it to directly supervise a Variational Autoencoder (VAE)-based policy network to imitate real animal behaviors. This significantly reduces the need for real interaction data and allows for rapid policy updates. We also develop a high-level network to track diverse commands and trajectories. Our simulated results show a tenfold sample efficiency increase compared to reinforcement learning methods such as PPO. In real-world testing, our policy achieves proficient command-following performance with only a two-minute data collection period and generalizes well to new speeds and paths.**

I. INTRODUCTION

Learning-based methods [1], [2], [3], [4], [5], [6], [7], [8] have recently demonstrated significant advantages in acquiring agile motor skills for quadrupedal robots. In particular, model-free deep Reinforcement Learning (RL) algorithms enables them to mimic animal motions and displays a natural behavior [3], [4], [9], [8], [10], [11].

However, model-free RL algorithms [12], [13], [14] usually require substantial on-policy data to improve their performance. Given the cost of collecting data in simulation compared to the real world, these algorithms often train policies in simulation and then deploy them on physical robots through zero-shot transfer. However, the policies learned in simulation may not consistently perform well in real-world scenarios due to the persistent sim-to-real gap. Researchers have attempted to mitigate this gap using techniques like domain randomization [15] and domain adaptation within simulation environments to enhance policy robustness. Nevertheless, these techniques do not provide a fundamental solution and cannot guarantee successful transfer. [16] argues that dynamics randomization and adaptation approaches may not consistently address sim-to-real transfer challenges, leaving the sim2real gap unresolved. On the other hand, an alternative approach is to train or fine-tune the policy directly on a real robot, which can effectively address the problem. [17] utilizes a model-free off-policy reinforcement learning algorithm for policy fine-tuning in the real world, albeit it still necessitates more than 2 hours data for fine-tuning. To increase sample efficiency, [18] adopts a model-based reinforcement learning approach, which enables direct policy training on the real robot. Nevertheless, since the policy network is trained through a model-free reinforcement learning algorithm, this method still requires over one hour to train a basic policy for walking towards predefined directions.

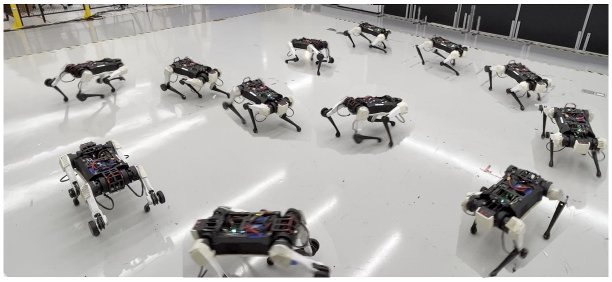

Fig. 1: Our robot Max follows the U-shape path after fine-tuned in the real world.

In the realm of computer graphics, ControlVAE [19] has demonstrated superior sample efficiency compared to deep reinforcement learning. It achieves this by co-training a world model with a VAE-based policy network [20]. Building on this concept, we introduce a model-based learning framework to close the sim2real gap by directly fine-tuning policies on real robots. First we train a world model capable of predicting several consecutive states of the robot. Leveraging the differentiability of the world model, we can train an end-to-end control policy by direct backpropagation. This policy imitates reference trajectories obtained from real dogs by interacting with the world model. Additionally, we develop a high-level policy for generating latent variable within the VAE [20]. This empowers the robot to follow various high-level commands and track diverse paths.

In simulated experiments, our method exhibits a tenfold improvement in sample efficiency compared to PPO [13], both during training and adaptation. In real robot experiments, our policy effectively tracks a oblong path at speeds

† Corresponding author

∗ Equal Contribution.

[1] Haojie Shi, Tingguang Li, Qingxu Zhu, Jiapeng Sheng, and Lei Han are affiliated with Tencent Robotics X, China, (email: haojieshi, teaganli,qingxuzhu,kevinsheng,lxhan@tencent.com)

[2] Haojie Shi is from the Chinese University of Hong Kong, and this work was done during internship at Tencent Robotics X Lab.

[3] Max Q.-H. Meng is with Shenzhen Key Laboratory of Robotics Perception and Intelligence and the Department of Electronic and Electrical Engineering at Southern University of Science and Technology in Shenzhen, China. He is a Professor Emeritus in the Department of Electronic Engineering at The Chinese University of Hong Kong in Hong Kong and was a Professor in the Department of Electrical and Computer Engineering at the University of Alberta in Canada. (email: max.meng@ieee.org)

of 0.6m/s, 0.9m/s, and 1.2m/s with just 2 minutes of fine-tuning. Furthermore, we also evaluate our policy generalization ability in new speed commands and unseen paths, highlighting our method's robust generalization capability.

In conclusion, the main contributions of this paper are:

1) We present a model-based learning framework to acquire agile skills in quadrupedal robots within simulations and fine-tune them on real robots, substantially enhancing the sample efficiency of learning-based methods in the robotics domain.
2) We assess our approach in both simulation and the real robot, demonstrating that with only 2 minutes of fine-tuning, our robot effectively executes reference commands.
3) We establish the generalization capability of our method with real robot experiments, as the fine-tuned policy follows previously unseen commands and paths.

II. RELATED WORK

As model-free deep reinforcement learning algorithms [12], [13], [14] continue to advance rapidly, recent research has achieved notable success in training expert policies for quadrupedal locomotion. In contrast to classical control methods [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], learning-based approaches harness the power of deep neural networks to acquire agile motor skills. Notably, [3], [4] employ PPO [13] to emulate natural motion patterns observed in animals. Moreover, [8] introduces a novel approach that incorporates terrain information while mimicking the behavior of real animals.

However, model-free deep reinforcement learning algorithms demand an enormous volume of interaction data, making it infeasible to collect on a real robot. Consequently, they often train the policy in simulation and attempt zero-shot transfer into the real world. To address the sim2real gap, [4] introduces an environmental encoding approach, optimizing it on the real robot by maximizing total returns for swift adaptation. It's crucial to note that the effectiveness of this adaptation strategy hinges on the degree of similarity between the simulation and real-world environments and may not be universally applicable across all tasks. Furthermore, [8], [31], [32], [15] employ domain randomization techniques to cultivate a robust policy and employ domain adaptation to address the sim2real gap. While these methods can alleviate the impact of the sim2real gap, they do not offer a fundamental solution. Notably, [16] demonstrates that their policy can be transferred to the real robot without necessitating domain randomization, questioning the necessity of this technique. In summary, the challenge of transferring model-free reinforcement learning policies from simulation to reality remains an open problem.

To increase sample efficiency for reinforcement learning, recent years have witnessed great progress in model-based reinforcement learning algorithms [33], [34], [35]. They first learn a dynamics model in the simulation and then improve their policy using model-free RL with imagined data produced by the learned model. To increase the prediction power of the world model, further research focuses on learning a compact latent space of world model [36], [37], [38], and also succeeds in the real robot training [18]. In this way, the sim2real gap does not exist since they train the policy directly in the real robot. While they can train walk policy in a real robot in one hour, it is still not evaluated how much data it will take to train a more complex policy like imitating an animal or following a desired path in our task. And since the policy is trained by the model-free reinforcement learning algorithm, the sample efficiency is still limited. Meanwhile, training directly in the real robot from scratch fails to take advantage of the simulation environment. In contrast, our method trains both the world model and control policy in a supervised manner, resulting in significantly enhanced sample efficiency. Additionally, we adopt a two-stage approach that involves training the policy in simulation to create a warm-up policy, followed by fine-tuning it in the real world using just two minutes of data. This significantly reduces the amount of real-world data required and enables the learning of more sophisticated motor skills.

ControlVAE [19] is an innovative technique in computer graphics that utilizes a VAE-based policy, supervised by a differentiable world model. This approach provides significantly higher sample efficiency than deep reinforcement learning algorithms, but it mainly concentrates on policy training for human motion generation within a simulation context. To expand on this concept, our proposed framework combines world model and policy learning in a supervised manner, resulting in a learning framework that enhances training efficiency during the fine-tuning stages on a real robot with a regularization term. Consequently, our approach allows for deployment on a real quadrupedal robot system with only a 2-minute fine-tuning period.

III. METHODOLOGY

Our framework contains two parts, i.e. a world model and a control policy, as shown in Fig. 2. The world model learns to approximate the unknown dynamics of the simulation and the reality. Given current robot state and action, it predicts next state. The control policy learns agile behaviors by imitating motions from real animals. Instead of interacting with a simulator, it directly collects samples predicted by the trained world model. Both the world model and the control policy are updated in a supervised manner and trained iteratively: we first collect state-action pairs under a fixed control policy to fit the system dynamics using the world model. Then the control policy is updated by interacting with the fixed world model. The whole process repeats until the control policy converges.

A. World Model Learning

We commence by training the world model f_{w}. It predicts the next state based on the current state and action, utilizing a residual form as follows:

$$\hat{s}_{t+1} = f_{\mathrm{w}}(\Delta s_t | o_t, a_t, \pi) + s_t, \tag{1}$$

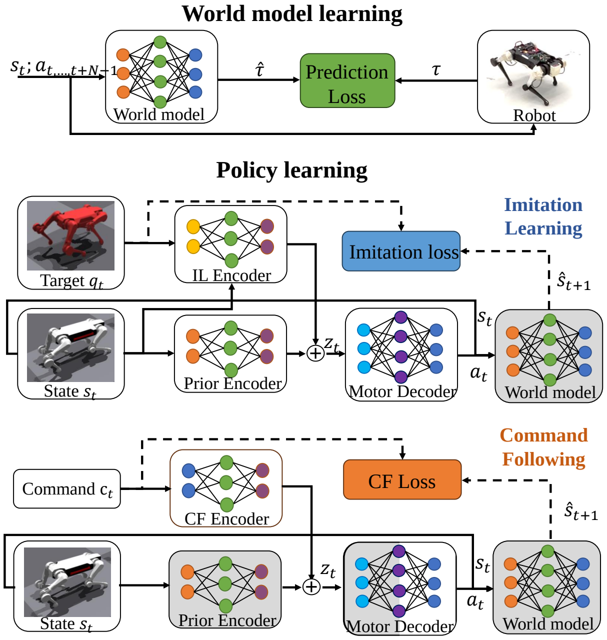

Fig. 2: Overview of our learning framework. The gray block represents fixed parameters. For the command following task, the Motor Decoder is fixed when training from scratch and becomes trainable during real-world fine-tuning.

where $f_{\mathrm{w}}(\Delta s_t|o_t, a_t, \pi)$ is a neural network parameterized by θ_{w}, s_t represents the robot state at time t, encompassing robot position, orientation, linear velocity, angular velocity, joint positions and joint velocities. o_t corresponds to the robot observation, including robot linear velocity, angular velocity, joint position, and joint velocity at robot local frame. a_t is the target angle for each joint which can be converted to joint torques through a PD controller. $\hat{s}_{t+1}$ represents the predicted state at the next time step $t+1$. When training the world model, the robot interacts with the simulator under a fixed control policy to collect state-action sequences $\tau = \{s_0, a_0, s_1, a_1, ..., s_n, a_n\}$. The world model is trained in supervised learning manner with the n-step prediction loss that is conducive to prediction in a long time horizon:

$$L_t^{\mathrm{w}} = \sum_{t=1}^{n} \|\hat{s}_t - s_t\|, \tag{2}$$

where $\hat{s}_t$ is the predicted robot state and s_t represents the ground truth robot state either from simulator or from the real robot during the fine-tuning stage.

B. Imitation Learning

In the context of the imitation task, our objective is to imitate motion sequences collected from real animals. We formulate this problem into an encoder-decoder architecture, where we encode the reference motion sequence into a latent embedding and decode this latent embedding together with robot observation into joint motor action. We take a VAE-based [20] architecture with an Imitation Learning Encoder $\pi_{\mathrm{IL}}(z_t|o_t, \boldsymbol{q}_t)$ to encode the observation o_t and a sequence of future reference motions $\boldsymbol{q}_t$ into a latent variable z_t. The Motor Decoder $\pi_{\mathrm{M}}(a_t|o_t, z_t)$ takes o_t and z_t and produces the action a_t. Besides, we incorporate an additional state-conditional prior $\pi_{\mathrm{prior}}(z_t|o_t)$ to disentangle distinct skills within the latent space, as emphasized in [19]. We model the latent variable's prior distribution $p(z_t|o_t)$ and posterior distribution $q(z_t|o_t, \boldsymbol{q}_t)$ as Gaussian distribution:

$$\begin{aligned} p(z_t|o_t) &\sim \mathcal{N}(\pi_{\mathrm{prior}}(z_t|o_t), \sigma^2 I), \\ q(z_t|o_t, \boldsymbol{q}_t) &\sim \mathcal{N}(\pi_{\mathrm{IL}}(z_t|o_t, \boldsymbol{q}_t) + \pi_{\mathrm{prior}}(z_t|o_t), \sigma^2 I), \end{aligned} \tag{3}$$

where $\pi_{\mathrm{IL}}(z_t|o_t, \boldsymbol{q}_t)$ and $\pi_{\mathrm{prior}}(z_t|o_t)$ are neural networks parameterized by θ_{IL} and θ_{prior}. I is a identity matrix, and σ is a fixed standard deviation for simplicity.

The imitation learning loss is defined as follows:

$$L_t^{\mathrm{I}} = 0.6L_t^{\mathrm{jpos}} + 0.05L_t^{\mathrm{jvel}} + 0.3L_t^{\mathrm{bpos}} + 0.05L_t^{\mathrm{bvel}}, \tag{4}$$

where the joint position loss L_t^{jpos}, joint velocity loss L_t^{jvel}, base position loss L_t^{bpos} and base velocity loss L_t^{bvel} are similar to the reward function in [8]:

$$\begin{aligned} L_t^{\mathrm{jpos}} &= 1 - \exp(-\|\hat{j}_t - \bar{j}_t\|^2), \\ L_t^{\mathrm{jvel}} &= 1 - \exp(-\|\hat{\dot{j}}_t - \bar{\dot{j}}_t\|^2), \\ L_t^{\mathrm{bpos}} &= 1 - \exp(-20\|\hat{p}_t^{\mathrm{base}} - \bar{p}_t^{\mathrm{base}}\|^2 - 10\|\hat{i}_t^{\mathrm{base}} - \bar{i}_t^{\mathrm{base}}\|^2), \\ L_t^{\mathrm{bvel}} &= 1 - \exp(-2\|\hat{\dot{p}}_t^{\mathrm{base}} - \bar{\dot{p}}_t^{\mathrm{base}}\|^2 - 0.2\|\hat{\dot{i}}_t^{\mathrm{base}} - \bar{\dot{i}}_t^{\mathrm{base}}\|^2), \end{aligned} \tag{5}$$

where j and $\dot{j}$ are joint position and joint velocity, p^{base} and i^{base} represent base position and orientation, $\dot{p}^{\mathrm{base}}$ and $\dot{i}^{\mathrm{base}}$ denote base velocity and base angular velocity. $\hat{(\cdot)}$ represents the states predicted by the world model and $\bar{(\cdot)}$ denotes the reference motion. Since the world model is differentiable, the gradient of the imitation loss can be calculated end-to-end through the differential dynamics.

To ensure the latent space is well formed so that we can further find an appropriate latent variable in the downstream command following task, we incorporate a KL-divergence loss for regularization:

$$\begin{aligned} L_t^{\mathrm{KL}} &= D_{\mathrm{KL}}(q(z_t|o_t, \boldsymbol{q}_t)||p(z_t|o_t)) \\ &= \|\pi_{\mathrm{IL}}(o_t, \boldsymbol{q}_t)\|^2 / 2\sigma^2. \end{aligned} \tag{6}$$

To make our policy focus on not only the next state but also a long-term horizon, the final loss term is calculated as the sum of n-step imitation loss, where the n-step roll out is predicted by the world model

$$L_t^{\mathrm{IL}} = \sum_{t=1}^{n} (L_t^{\mathrm{I}} + 0.1L_t^{\mathrm{KL}}). \tag{7}$$

C. Command Following

The next step is to train a policy that follows linear velocity and angular velocity specified by users. We introduce the Command Following Encoder $\pi_{\mathrm{CF}}(z_t|o_t, c_t)$ to encode the commands into the latent space. Given a random command $c_t = [\bar{v}_t, \bar{\omega}_t]$, where $\bar{v}_t$ and $\bar{\omega}_t$ represent the desired linear velocity in forward direction and the desired angular velocity, the posterior distribution of latent variable z_t is computed as:

$$q(z_t|o_t, c_t) \sim \mathcal{N}(\pi_{\mathrm{prior}}(z_t|o_t) + \pi_{\mathrm{CF}}(z_t|o_t, c_t), \sigma^2 I), \tag{8}$$

Algorithm 1 Fine-tune on the real robot

```
Require: Network parameters $\theta_\mathrm{w}, \theta_\mathrm{prior}, \theta_\mathrm{CF}, \theta_\mathrm{M}$, learning
    rates $\alpha_\mathrm{w}, \alpha_\pi$, update number $n_\mathrm{w}, n_\pi$, max iterations $N$,
    samples $n_\mathrm{sample}$, rollout length $n$, replay buffer $\mathcal{B}$, batch
    size $M$
for i in $\{0...N-1\}$ do
    send updated control policy to the real robot
    roll out $n_\mathrm{sample}$ steps on the real robot, send to $\mathcal{B}$
    // Update the world model
    for j in $\{1...n_\mathrm{w}\}$ do
        sample trajectories $(s_0, a_0, ..., s_n, a_n)_{1...M}$ from $\mathcal{B}$
        compute prediction loss $L^\mathrm{w}$ in Eq. 2
        $\theta_\mathrm{w} \leftarrow \theta_\mathrm{w} + \alpha_\mathrm{w} \nabla_{\theta_\mathrm{w}} L^\mathrm{w}$
    end for
    // Update the control policy
    for j in $\{1...n_\pi\}$ do
        sample states and commands $(s_0, c_0)_{1...M}$ from $\mathcal{B}$
        roll out $n$ steps predicted by world model $f_\mathrm{w}$
        compute $L^\mathbf{CF} \leftarrow \sum_{t=1}^{n}(L_t^\mathrm{CF} + 0.1 L_t^\mathrm{reg})$ in Eq. 9, 12
        $\theta_\mathrm{CF} \leftarrow \theta_\mathrm{CF} + \alpha_\pi \nabla_{\theta_\mathrm{CF}} L^\mathbf{CF}$
        $\theta_\mathrm{M} \leftarrow \theta_\mathrm{M} + \alpha_\pi \nabla_{\theta_\mathrm{M}} L^\mathbf{CF}$
    end for
end for
```

where $\pi_\mathrm{CF}(z_t|o_t, c_t)$ is the neural network parameterized by θ_CF. Since our goal is to make the robot follow the command, the command following loss comprises both linear velocity loss L_t^v and angular velocity loss L_t^ω:

$$L_t^\mathrm{CF} = 2L_t^\mathrm{v} + L_t^\omega, \tag{9}$$

$$L_t^\mathrm{v} = 1 - \exp(-2|\bar{v}_t - \hat{v}_t|), \tag{10}$$

$$L_t^\omega = 1 - \exp(-2|\bar{\omega}_t - \hat{\omega}_t|), \tag{11}$$

where $\bar{(\cdot)}$ represents the user command while $\hat{(\cdot)}$ is the robot states predicted by the world model. To preserve the naturalness of the robot behavior, we exclusively update the command following network $\pi_\mathbf{CF}$ while keeping the prior network π_prior and motor decoder π_M fixed during training.

D. Fine-tune on a Real Robot

Owing to the sim-to-real gap, the policy learned from the simulation may fail when deployed on the real robot. Hence, we fine-tune both the Command Following Encoder and the Motor Decoder on the real robot to follow the desired paths. To preserve the natural behavior originating from the original Motor Encoder π_M^{ori}, we introduce a regularization term:

$$L_t^\mathrm{reg} = \|\pi_\mathrm{M}^\mathrm{ori}(a_t|o_t, z_t) - \pi_\mathrm{M}(a_t|o_t, z_t)\|. \tag{12}$$

The algorithm for fine-tuning in the real world is outlined in Algorithm 1.

IV. EXPERIMENT RESULTS

In this section, we report experimental results to address the following pivotal questions: (i) How effective is our approach in improving sample efficiency, compared with RL methods? (ii) How well is our fine-tuning process on the real robot can help to close the sim-to-real gap? (iii) Does our fine-tuned policy exhibit sufficient generalization capacity on previously unseen tasks? We conduct experiments both in simulation and real world. We compare our method with a RL baseline in terms of sample efficiency. In the real world experiment, we conduct the fine-tuning process on a real quadrupedal robot. To further demonstrate the generalization ability, we performance path following task on four unseen paths.

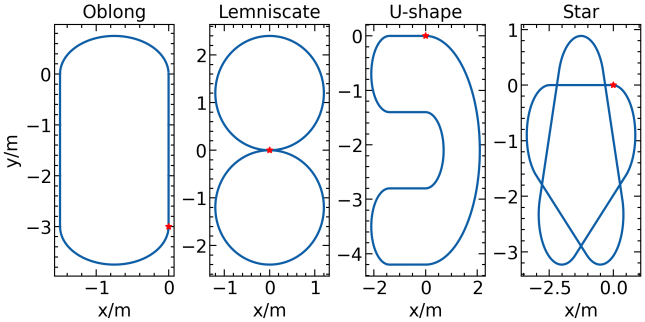

Fig. 3: Four types of desired paths. The red star represents the starting point.

A. Evaluation in Simulation Environments

Sample Efficiency in Imitation Learning Task. To address the first question regarding sample efficiency, we initially train the imitation task from scratch using Isaac Gym [39]. Isaac Gym is a GPU-based physical simulator simulating a batch of agents concurrently. In this task, we simultaneously employ 128 agents for training. We compare our method with PPO algorithm [13] with respect to the number of samples collected from the simulator. The reward function for PPO is defined as $r_t = 1 - L_t^\mathrm{I}$. We maintain an identical policy network structure for both methods to facilitate a meaningful comparison. The mean reward during training is reported in Fig. 4(a). It demonstrates that our method achieves a mean reward of 0.8 with approximately 5 million samples, as indicated in the read dashed line. In contrast, the PPO algorithm requires over 70 million samples to achieve similar results. This showcases that our method's sample efficiency surpasses that of PPO by over tenfold.

Sample Efficiency in Adapting to New Environments. Directly training PPO on a real robot is dangerous and may easily damage the robot. To compare the sample efficiency in adapting to new environments, we introduce variations to physical parameters in simulation and perform the fine-tuning process to make the adaptation. For the imitation task, we alter a number of physical parameters as shown in *Env1*, TABLE I. For example, we significantly increase the robot's mass from 5.74 kg to 14 kg, which makes the new environment extremely difficult for the original policy. To emulate a scenario akin to real-world robot data collection, we employ 2 agents in the simulation environment for both methods. In our approach, each training iteration accumulates 3000 samples, equivalent to 1 minute of data collection given the control frequency of 50 Hz. For PPO, policy updates are conducted every 32 steps. Fig. 4(b) depicts the training

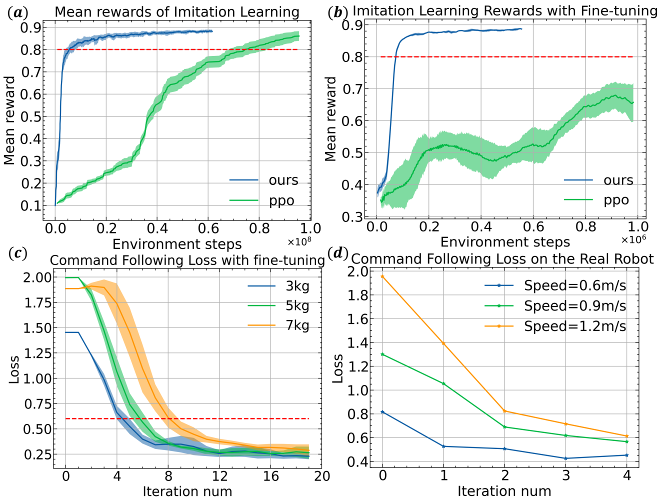

Fig. 4: (a) Training curves of the imitation learning in the simulation. (b) Training curves of fine-tuning the imitation learning policy in the modified simulation environment. (c) Mean loss of fine-tuning the path following policy in three workloads within the modified simulation environment. (d) Mean loss of fine-tuning the path following policy under various speeds on the real robot.

TABLE I: The physical parameters of the original and new environments, where Ctrl Lat represents Control Latency.

	Mass (kg)	Kp	Ctrl Lat (ms)	Max Torque (Nm)
Original	5.74	50.0	0.0	18.0
Env1	14.0	40.0	6.0	16.2
Env2	5.74+3.0	50.0	6.0	18.0
Env3	5.74+5.0	50.0	6.0	18.0
Env4	5.74+7.0	50.0	6.0	18.0

curves. The plot highlights that our method attains a mean reward of 0.8 with roughly 50,000 samples (equivalent to approximately 17 minutes of data) in this challenging setting. Conversely, PPO algorithm remains subpar even with ten times the sample size.

To further investigate the performance of command following, we extend this task to path following, where the robot aims at following predefined paths, as shown in Fig. 3. We employ the pure pursuit algorithm [40] to convert the path information to commands. In this experiment, we follow the *Oblong* with a target speed of 0.9 m/s. We also create three distinct environments, *Env2*, *Env3*, *Env4*, as shown in TABLE I. To emulate the fine-tuning process in the real-world environment, each training iteration involves collecting 1500 samples (30 seconds data). Fig. 4(c) depicts the training curves with the loss term defined in Eq. 9. From the plot, we observe that our approach, under workloads of 3kg, 5kg, and 7kg,requires approximately 4 iterations (2 minutes), 6 iterations (3 minutes), and 8 iterations (4 minutes) of data to achieve a loss of less than 0.6. This result indicates a relatively good performance at these speeds. In comparison, the loss of PPO remains nearly unchanged with such a limited amount of samples, and thus we did not draw the result. In this way, we can demonstrate the high sample efficiency of our approach for both training and fine-tuning, adapting to different environments in both imitation learning and path following tasks.

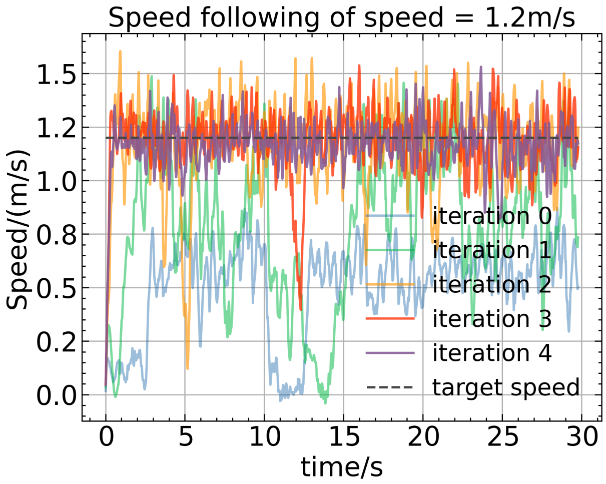

Fig. 5: Speed following at 1.2 m/s along the oblong path on the real robot with real-world adaptation.

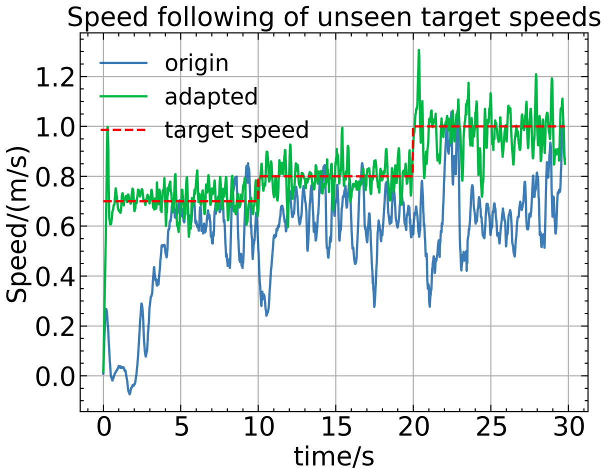

Fig. 6: Speed following along the oblong path on the real robot using the original policy and the adapted policy.

B. Evaluation on Real World Experiments

Adapting from Simulation to Reality. To address the second question, we perform physical experiments using the real robot Max. Due to the sim2real gap, the policy trained in the simulation may fail to follow the path with the desired speed and can exhibit significant lag behind the target speed, especially at high target speeds. This underscores the necessity of real-world fine-tuning. We perform three adaptation experiments on *Oblong* with target speeds of 0.6m/s, 0.9m/s, and 1.2m/s. To fine-tune the policy in the real world, each iteration involves collecting 30 seconds of data (1500 samples) to train the world model, followed by updating the policy network using data predicted by the adapted world model. Fig. 4(d) displays the command following loss L^{CF} for four iterations (2 minutes data) with target speeds of 0.6m/s, 0.9m/s, and 1.2m/s on the real robot. TABLE II presents the averaged linear velocity error $e_v = \frac{1}{n}\sum_{t=1}^{n}|\bar{v}_t - \hat{v}_t|$ and angular velocity loss $e_\omega = \frac{1}{n}\sum_{t=1}^{n}|\bar{\omega}_t - \hat{\omega}_t|$ computed

TABLE II: The averaged linear velocity error e_v and angular velocity loss e_ω computed in a trajectory of 30s after each iteration. Iter0 refers to the original policy without any fine-tuning.

	Speed=0.6m/s		Speed=0.9m/s		Speed=1.2m/s	
	e_v	e_ω	e_v	e_ω	e_v	e_ω
Iter0	0.088	0.587	0.250	0.612	0.696	0.501
Iter1	0.055	0.241	0.194	0.565	0.431	0.319
Iter2	0.047	0.232	0.098	0.297	0.148	0.276
Iter3	**0.038**	0.190	0.078	0.269	0.103	0.286
Iter4	0.047	**0.189**	**0.063**	**0.249**	**0.081**	**0.240**

TABLE III: The averaged linear velocity error e_v, angular velocity error e_ω, and distance error e_p computed across four paths with unseen target speeds equal to 0.7m/s, 0.8m/s and 1.0m/s.

	Oblong			Lemniscate		
	e_v	e_ω	e_p	e_v	e_ω	e_p
origin	0.269	0.578	2.031	0.224	0.642	2.190
adapted	0.052	0.239	0.901	0.057	0.199	0.725

	U-shape			Star		
	e_v	e_ω	e_p	e_v	e_ω	e_p
origin	0.233	0.572	1.560	0.287	0.631	2.031
adapted	0.053	0.210	0.771	0.050	0.234	0.901

within a 30-second trajectory after each iteration during the real-world adaptation. It is evident that after the first iteration, there is a significant decreasing in losses. Particularly for a speed of 1.2m/s, the speed error decreases by more than 0.26m/s. After four iterations, the losses appear to converge, and the final performance is highly effective in tracking the commands. For example, Fig. 5(a) depicts speed tracking at 1.2m/s on the real robot with real-world adaptation. It is evident that in the initial policy (iteration 0), the actual speed lags considerably behind the target speed. After the first iteration, the actual speed can somewhat follow the target, but it exhibits significant fluctuations. In iteration 4, the policy effectively tracks the target speed with minimal vibration.

Generalization Ability on Unseen Scenarios. To answer the last question, we evaluate our policy on unseen command velocities and paths. In the previous experiment, we collect real robot data, totaling 7.5 minutes of data with target speeds of 0.6m/s, 0.9m/s, and 1.2m/s. We utilize this data for off-policy fine-tuning to derive the adapted policy. We test the generalization ability on unseen target velocities of 0.7m/s, 0.8m/s, and 1.0m/s on all paths including unseen *Lemniscate*, *U-shape*, and *Star*. TABLE III reports averaged linear velocity error (e_v), angular velocity error (e_ω), and distance error (e_p) computed over four paths lasting 30 seconds each. The distance error is defined as $e_p = \frac{1}{n}\sum_{t=1}^{n}\|p_t - p_t^*\|$, where p_t, p_t^* are robot position and the target position at time t. p_t^* is derived by integrating the target speed with respect to time. From the table, it's evident that after off-policy adaptation, all of the errors have decreased by over one-half. Fig. 5 vividly demonstrates the speed tracking to follow the oblong path on the real robot under the original policy and the adapted one. The original policy lags behind the target unseen speeds, whereas our adapted policy can follow them effectively with averaged linear velocity error of around 0.05m/s. Fig. 7 displays the real trajectories for tracking the path at different unseen target speeds. From the plot, it's evident that the original policy lags significantly behind the target trajectory, while our adapted policy can effectively track it, performing even slightly faster at higher speeds. In conclusion, the experimental results demonstrate that our adapted policy can successfully handle unseen commands and track unfamiliar paths, highlighting the generalization capability of our approach.

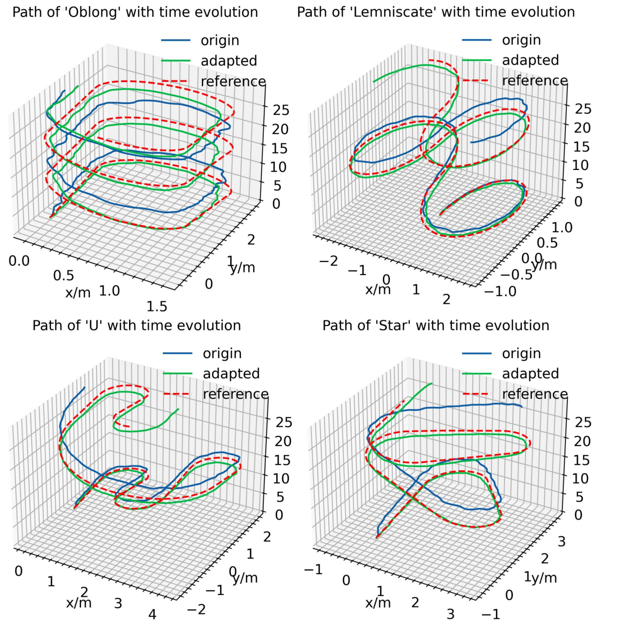

Fig. 7: Path following on unseen paths under the original policy and the adapted one. The z-axis represents the time evolution, and the reference path is computed by integrating the target speed with respect to time.

V. CONCLUSIONS

In summary, we have introduced an efficient learning framework designed to mimic the natural behavior of animals and enable path tracking for quadrupedal robots. Our approach begins by training a world model and a policy network, effectively turning it into an auto-encoder that utilizes the differential dynamics from the world model. This strategy significantly boosts sample efficiency, outperforming model-free deep reinforcement learning algorithms by over tenfold. Additionally, our method facilitates rapid policy fine-tuning on real robots, requiring only 2 minutes of data, and demonstrates robust generalization capabilities. Future directions could include developing a world model with perception information, allowing the framework to adapt to visual locomotion across challenging terrains. The fine-tuning algorithm can narrow the sim2real gap further and improve the success rate of visual locomotion in challenging environments. In conclusion, our work opens up exciting possibilities for training complex motor skills on real robots.

References

[1] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg, "Learning a contact-adaptive controller for robust, efficient legged locomotion," *arXiv preprint arXiv:2009.10019*, 2020.

[2] 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.

[3] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[4] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[5] 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.

[6] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, 2020.

[7] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," in *Conference on Robot Learning*, pp. 773–783, PMLR, 2022.

[8] T. Li, Y. Zhang, C. Zhang, Q. Zhu, W. Chi, C. Zhou, L. Han, *et al.*, "Learning terrain-adaptive locomotion with agile behaviors by imitating animals," *arXiv preprint arXiv:2308.03273*, 2023.

[9] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 25–32, IEEE, 2022.

[10] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, pp. 1–20, 2021.

[11] L. Han, Q. Zhu, J. Sheng, C. Zhang, T. Li, Y. Zhang, H. Zhang, Y. Liu, C. Zhou, R. Zhao, *et al.*, "Lifelike agility and play on quadrupedal robots using reinforcement learning and generative pre-trained models," *arXiv preprint arXiv:2308.15143*, 2023.

[12] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

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

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[15] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30, IEEE, 2017.

[16] Z. Xie, X. Da, M. Van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4955–4961, IEEE, 2021.

[17] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 1593–1599, IEEE, 2022.

[18] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Conference on Robot Learning*, pp. 2226–2240, PMLR, 2023.

[19] H. Yao, Z. Song, B. Chen, and L. Liu, "Controlvae: Model-based learning of generative controllers for physics-based characters," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 6, pp. 1–16, 2022.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3359–3365, IEEE, 2017.

[22] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2245–2252, IEEE, 2018.

[23] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization with implicit hard contacts," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3316–3323, 2018.

[24] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization for legged robots with slipping motions," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3013–3020, 2019.

[25] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 1–9, IEEE, 2018.

[26] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5761–5768, IEEE, 2018.

[27] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, pp. 1441–1460, 2018.

[28] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2017.

[29] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[30] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 93–100, IEEE, 2017.

[31] 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," *arXiv preprint arXiv:1804.10332*, 2018.

[32] H. Shi, B. Zhou, H. Zeng, F. Wang, Y. Dong, J. Li, K. Wang, H. Tian, and M. Q.-H. Meng, "Reinforcement learning with evolutionary trajectory generator: A general approach for quadrupedal locomotion," *IEEE Robotics and Automation Letters*, pp. 3085–3092, 2022.

[33] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma, "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees," *arXiv preprint arXiv:1807.03858*, 2018.

[34] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," in *Conference on Robot Learning*, pp. 617–629, PMLR, 2018.

[35] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," *arXiv preprint arXiv:1802.10592*, 2018.

[36] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering atari with discrete world models," *arXiv preprint arXiv:2010.02193*, 2020.

[37] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[38] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," *arXiv preprint arXiv:2301.04104*, 2023.

[39] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[40] R. C. Coulter *et al.*, *Implementation of the pure pursuit path tracking algorithm*. Carnegie Mellon University, The Robotics Institute, 1992.