Learning a Single Policy for Diverse Behaviors on a Quadrupedal Robot using Scalable Motion Imitation

Arnaud Klipfel[1] Nitish Sontakke[1] Ren Liu[2] Sehoon Ha[1]

***Abstract*— Learning various motor skills for quadrupedal robots is a challenging problem that requires careful design of task-specific mathematical models or reward descriptions. In this work, we propose to learn a single capable policy using deep reinforcement learning by imitating a large number of reference motions, including walking, turning, pacing, jumping, sitting, and lying. On top of the existing motion imitation framework, we first carefully design the observation space, the action space, and the reward function to improve the scalability of the learning as well as the robustness of the final policy. In addition, we adopt a novel adaptive motion sampling (AMS) method, which maintains a balance between successful and unsuccessful behaviors. This technique allows the learning algorithm to focus on challenging motor skills and avoid catastrophic forgetting. We demonstrate that the learned policy can exhibit diverse behaviors in simulation by successfully tracking both the training dataset and out-of-distribution trajectories. We also validate the importance of the proposed learning formulation and the adaptive motion sampling scheme by conducting experiments.**

I. Introduction

Quadrupedal robots can achieve various autonomous missions by overcoming rough terrains that wheeled robots cannot traverse, but the control is not straightforward due to its high-dimensional state space and under-actuated dynamics. Roboticists have studied various approaches for legged robot control, ranging from model-based control [1]–[4] to learning-based approaches [5]–[8], which have demonstrated impressive agility and robustness on various quadrupedal robots. However, most of the prior works have focused on the given specific task, such as robust walking, running, or jumping, because they are governed by very different dynamics. These task-specific controllers often require manual engineering based on the expert's prior knowledge, which can be either developing mathematical models for model-based controllers or shaping reward functions for learning-based algorithms. It requires even more effort if the developer wants to improve the naturalness of the behavior.

One interesting approach is to develop a motion imitation controller that can track the given reference motion, which defines the task implicitly. For instance, walking and jumping are two very different tasks, but motion imitation treats them as the same task of tracking the corresponding motion. If the reference is captured by a human or an animal, motion imitation can also allow us to develop natural behaviors from the original actor. One notable early work is DeepMimic proposed by Peng et al. [9], which shows an impressive motion-tracking performance on a simulated humanoid character. Many researchers have extended this work to imitate a wide range of motions on a simulated character by investigating novel policy architectures [10] or introducing adversarial learning [11], [12]. This motion imitation approach has been investigated in the context of robotics as well to develop natural motions [13], but it is limited to tracking a single reference motion.

This paper investigates a scalable motion imitation framework for a quadrupedal robot to track various motor skills

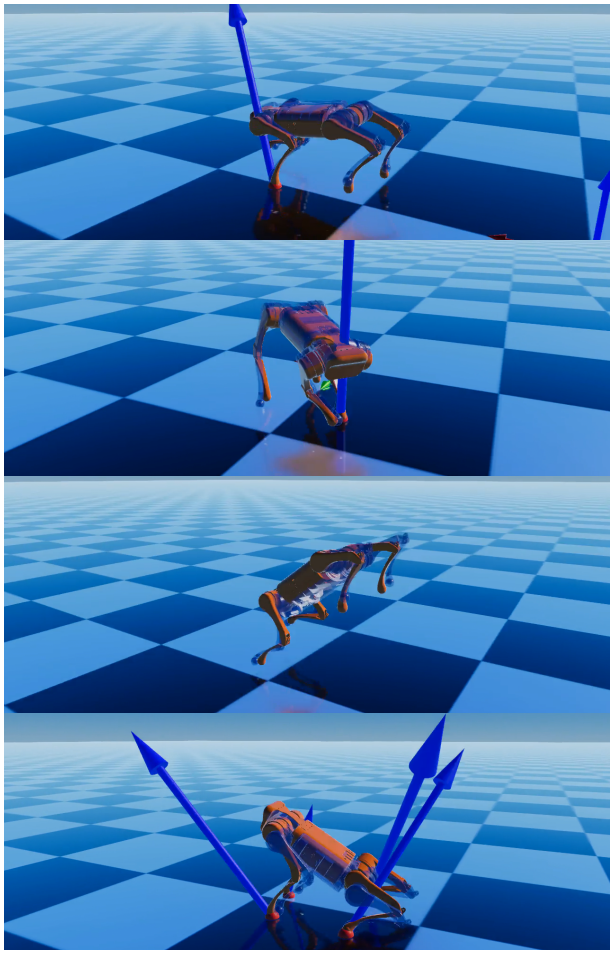

Fig. 1. Our scalable motion imitation framework can learn a single policy to execute many motor skills, from running (**1st**), turning (**2nd**), jumping (**3rd**), and sitting (**4th**).

[1]School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, 30308, USA. aklipfel3@gatech.edu, nitishsontakke@gatech.edu, sehoonha@gatech.edu

[2] Meta Platforms, Inc., USA, renl@meta.com. Work done while at Georgia Tech.

using a single control policy. For the preprocessing, we carefully prepare all the reference motions by retargeting the existing dog's motion database, which results in 701 motion clips with 15 different motion types. Then we extend the existing motion imitation framework [9] to improve its scalability and robustness. We design a new problem formulation, including a new observation space that includes future and past references and a new reward function that does not incentivize low-level kinematic tracking. In addition, we propose a novel adaptive motion sampling (AMS) scheme to learn all the trajectories without ignoring some outlier motions, such as jumping, and also to avoid catastrophic forgetting about the previously learned motor skills.

We demonstrate that our framework can learn a single versatile motion imitation policy that can track a large variety of reference motions. Our policy can even track new out-of-distribution trajectories, such as a star-shaped path or a motion with multiple jumps. By conducting an ablation study, we show that adaptive trajectory sampling is necessary to learn all the motor skills in the database. We also demonstrate the robustness of the framework, which is achieved by our novel learning formulation. Our key contributions are summarized as follows:

- We propose novel techniques for greatly improving the robustness and scalability of motion imitation.
- We showcase that our framework can learn a single policy to track various challenging trajectories.
- We validate the proposed components by conducting ablation studies.

II. Related works

A. Quadrupedal Locomotion

The control of quadrupedal robots has been thoroughly studied by many robotics researchers. One common approach is to develop a model-based controller that captures the important characteristics of the robot's dynamics using a mathematical model and generates optimal control trajectories [1]–[4]. While demonstrating impressive robustness and agility on hardware, a model-based approach often requires manual engineering to develop the proper dynamics model for the given task. In recent years, researchers have showcased that it is possible to learn robust locomotion policies using deep reinforcement learning (deep RL) [5]–[8]. However, it is also a well-known challenge that deep RL often requires an extensive amount of reward shaping to obtain the best quality policy that can be effectively transferred to the real world. Therefore, the developed reward functions sometimes have many different terms, up to nine or ten, to guarantee symmetric, energy-efficient, cyclic, and effective gaits [14], [15]. Therefore, developing a high-quality motion controller for novel tasks still remains a challenging problem for both model-based and learning-based approaches and requires a lot of human effort.

B. Motion Imitation

Motion imitation is a problem formulation that aims to track the given reference motion. Because the task is implicitly encoded in the reference, this framework allows us to use a unified problem formulation for various tasks, unlike standard task-based problem formulations. The early work of Peng et al. [9] demonstrates that it is possible to train a virtual human character to track a single motion in a physics-based simulation. This research is followed by many other works in computer animation [10]–[12], [16] to track a wide range of motions. The robotic community also adopts the same motion imitation framework to develop quadrupedal robot controllers to achieve natural animal-looking motions [13]. Kim et al. [17] demonstrate a human motion interface to control a quadrupedal robot by combining motion imitation and motion retargeting. Escontrela et al. [18] show that adversarial reward formulation of motion imitation can be a good substitute option for complex reward functions. Our work is also closely related to these state-of-the-art contributions in both the computer animation and robotics communities. We extend the motion imitation framework to support a large dataset for quadrupedal locomotion by proposing a novel adaptive sampling and policy design.

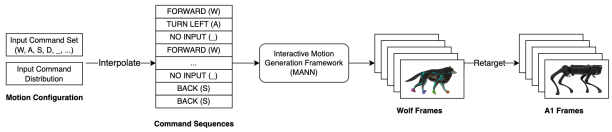

Fig. 2. Motion dataset generation pipeline.

III. Scalable Motion Imitation

In this section, we will describe the proposed scalable motion imitation framework to track more than 700 motion clips as well as out-of-distribution trajectories with a single policy. We first explain our data generation procedure in Section III-A. Then we present our novel problem formulation in Section III-B, which is designed to improve the robustness of the existing motion imitation framework. Finally, we describe our novel adaptive trajectory sampling method in Section III-C, which is necessary to learn a large number of motion trajectories.

A. Data Generation

The motion dataset contains 701 motion clips with 15 different motion types. We generate the dataset using our motion generation pipeline (See Fig. 2). Every motion clip in the dataset lasts for 10 seconds and is of 60 Hz frame rate. For the random keyboard input commands and their distribution, the pipeline interpolates them into a sequence of 600 to simulate user interaction. Then we infer kinematic data, such as joint angles, with the interactive motion generation framework [19], which is trained with a wolf character skeleton.

Therefore, we need to retarget the motion of a wolf into our A1 quadrupedal robot [20]. We apply the algorithm implemented by Peng et al. [13], which pairs corresponding key points from the source subject's body to the target

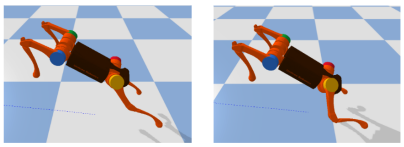

Fig. 3. Jumping with different scaling factors.(**Left**: 1.0, **Right**: 0.825 (ours)).

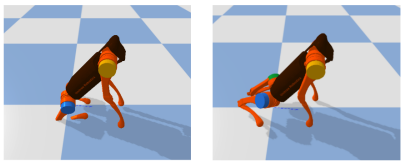

Fig. 4. Different retargeted sit motions. **Left:** IK with hips and feet restriction. **Right (ours)**: Adjusted IK considering knee positions for sit motions.

robot's body, including the positions of the feet and hips, and then performs inverse-kinematics (IK) [21] to fulfill the morphological gap. The pipeline has many robot model-specific hyperparameters that affect the results. For instance, Fig. 3 shows that an improper coordinates scaling may cause physically unreasonable front knee joint angles when jumping. We also need to remove all the ground penetration: therefore, our retargeting algorithm is aware of the motion type to handle such special cases (See Fig. 4). We further remove some artifacts, such as foot skating or jitterness, by applying inverse kinematics and smoothing. The content of the generated database is summarized in Table I.

B. Problem Formulation for Scalability and Robustness

Imitating the given reference motion is a popular approach to developing a versatile physics-based controller. While researchers traditionally have approached this problem using model-based control [22], the recent advances in deep reinforcement learning offer an automated approach to learning a tracking policy for a variety of motions. We mostly follow the formulation of Peng et al. [9], while making a few adjustments in the state, action, and reward function designs. We define our problem as a Markov Decision Process with reward function at time t, r_t, action $\mathbf{a}_t$, observation $\mathbf{o}_t$ and state $\mathbf{s}_t$.

TABLE I

GENERATED MOTION CLIP DATASET SPECIFICATION

Motion Type	Stand	Step	Pace	Trot	Gallop	Jump
Number of Clips	1	1	200	50	1	200
Motion Type	Turn Left		Turn Right		Sit	Lie
Number of Clips	51		51		1	1
Motion Type	Turn Left In-Place			Turn Right In-Place		
Number of Clips	25			25		
Motion Type	Triangle Trace		Star Trace		Random Mixed	
Number of Clips	1		1		92	

1) Observation Space: In our formulation, the observation space consists of three components: the robot proprioceptive data, some privileged data, and the reference motion data to track. For each control time step $t \in \mathbb{R}$ (every $0.02s$), the robot proprioception is composed of the joint positions in radians $\mathbf{q}_t \in \mathbb{R}^{12}$, the joint velocities $\dot{\mathbf{q}}_t \in \mathbb{R}^{12}$ in rad/s which are given by the encoders, the angular velocity $\omega_t \in \mathbb{R}^3$ in rad/s which is given by the robot on-board gyroscope. The policy has also access to some privileged information that is usually not estimated on a robot, or which requires some additional estimation than just pure proprioceptive readings [23]. This privileged information is composed of the Center-Of-Mass (CoM) $\mathbf{x}_t \in \mathbb{R}^3$ of the robot with respect to (w.r.t) an origin frame (the same as the reference data), the robot base orientation w.r.t the same origin frame given as the full rotation matrix $\mathbf{R}_t \in SO(3)$, and the body linear velocity at the CoM $\mathbf{v}_t \in \mathbb{R}^3$, expressed in the inertial frame of the robot. The CoM position could be estimated using a joint or visual odometry, as well as the rotation matrix, and the velocity could also be estimated using the accelerometer data [23]. The robot data is written as: $\mathbf{o}_t^{robot} = (\mathbf{x}_t^T, (\mathbf{R}_t)_{i,j}^{i,j\in[1,3]}, \mathbf{q}_t^T, \mathbf{v}_t^T, \omega_t^T, \dot{\mathbf{q}}_t^T) \in \mathbb{R}^{42}$. $(\mathbf{A})_{i,j}$ refers to the coefficient (i,j) of the matrix $\mathbf{A}$.

In contrast to other works [9], [10], the observation space does not include the state of every joint and link of the robot (i.e. twist information, orientation, and relative body position w.r.t. the root joint) and contains only the base full state and the joint angles. The policy has not access to a key frame identifier or marker such as a normalized phase variable [9] that is used to make the motion learning faster.

The reference motion vector $\bar{\mathbf{m}}_t \in \mathbb{R}^{24\times 8=192}$ is comprised of the target joint positions $\bar{\mathbf{q}}_t \in \mathbb{R}^{12}$, the rotation matrix $\bar{\mathbf{R}}_t \in SO(3)$, and the CoM position w.r.t an origin frame $\bar{\mathbf{x}}_t \in \mathbb{R}^3$. Then our observation $\mathbf{o}_t \in \mathbb{R}^{234}$ is defined as the concatenation of the robot data and the reference motion data for a short time window, $\mathbf{o}_t = (\mathbf{o}_t^{robot^T}, \bar{\mathbf{m}}_{t-1.0}^T, \bar{\mathbf{m}}_{t-0.5}^T, \bar{\mathbf{m}}_{t-0.2}^T, \bar{\mathbf{m}}_{t-0.02}^T, \bar{\mathbf{m}}_{t+0.02}^T, \bar{\mathbf{m}}_{t+0.2}^T, \bar{\mathbf{m}}_{t+0.5}^T, \bar{\mathbf{m}}_{t+1.0}^T)$. Note that we include both past and future reference motions for learning efficiency. We also exclude the current reference frame $\bar{\mathbf{m}}_t$ to avoid the copy-and-paste behavior of the current frame and promote broader exploration, i.e. adaptation of the low-level joint positions of the reference to the robot and environment dynamics.

2) Action Space: The action $\mathbf{a}_t \in \mathbb{R}^{12}$ is defined as the delta to a nominal (i.e independent of the reference and fixed at all time) joint configuration of the robot, which becomes the target position for the proportional-derivative controller at each joint. The generated actions are further smoothed by applying a moving average with a window size of two. The nominal joint configuration is: $\mathbf{a}_m = (-0.01, 0.75, -1.5, 0.01, 0.75, -1.5, -0.01, 0.75, -1.5, 0.01, 0.75, -1.5)$, which corresponds to a standing configuration. Joint positions are bounded. $\theta_{hip} \in [-0.5, 0.5]rad$, $\theta_{thigh} \in [-0.1, 1.5]rad$, and $\theta_{calf} \in [-2.1, -0.5]rad$.

3) Reward Function: We design our reward function as follows:

$$r_t = w_1 exp(-k_1||\bar{\mathbf{x}}_t - \mathbf{x}_t||^2) + w_2 exp(-k_2||\bar{\mathbf{R}}_t - \mathbf{R}_t||^2) + w_3 exp(-k_3||\bar{\mathbf{e}}_t - \mathbf{e}_t||^2), \quad (1)$$

where $\mathbf{x}$, $\mathbf{R}$, and $\mathbf{e}$ are the root position, the base orientation represented as a rotation matrix, and the end-effector positions expressed in the origin frame. The other terms $\bar{\mathbf{x}}$, $\bar{\mathbf{R}}$, and $\bar{\mathbf{e}}$ are the corresponding desired values from the reference motions. Therefore, each term encourages to track the given reference motion. w_1, w_2, and w_3 are the weight vectors to adjust the importance of each term and k_1, k_2, and k_3 are additional decaying parameters to tune the sensitivity of the reward term. We set the parameters as $w_1 = 0.7$, $w_2 = 0.5$, $w_3 = 0.15$, $k_1 = 12.5$, $k_2 = 20.0$, and $k_3 = 40.0$ for all the experiments.

As discussed in III-B.3, we do not have a low-level tracking reward term as [9], [10], [13], in order to prevent overfitting on the kinematics, and as such we provide more joint position information in the observation space to passively enforce the reference joint positions.

4) Early Termination: In contrast to [9], [10], which uses early termination based on the CoM tracking performance [9], or the reward function [10] in order to speed up the motion tracking learning, and avoid poor performing tracking policies. Our formulation uses simple contact-based termination without penalty. The only allowable contacts are the four feet with the ground. Won et al. [10] pointed out that contact-based terminations prevented them from learning motions which included self-body contacts. However, as our final goal is to deploy learned behaviors on a real robot, self-body contacts or inadmissible contacts are not desirable. Our formulation, without penalty and prior in the action space (joint residuals w.r.t. a nominal joint configuration) allows the policy from learning certain motions where the kinematic reference has inadmissible contacts, the policy tries to satisfy the high-level reference state (CoM, End-effector, body orientation) instead of trying to reproduce the low-level reference data (joint positions) at any cost.

C. Adaptive Motion Sampling

Although many researchers have demonstrated successful motion imitation using deep RL, it is still challenging to learn a single policy for various heterogeneous locomotion skills [9], [13], [24], including walking, turning, and jumping. One difficulty is that the computation of gradients is highly affected by a stochastic sampling of simulation rollouts, which is impossible to cover the entire range if there are too many reference motions in the database. Even worse, the stochastic nature of deep RL can lead a policy to forget about previously learned motor skills, which is referred to as catastrophic forgetting.

Our Adaptive Motion Sampling (AMS) allows us to train our policy from an unlabelled and unbalanced dataset. We found that our policy even performed better at tracking the reference motion data and producing natural joint motions when trained directly on all the motion clips at once. No pre-training is thus required. Having a rich set of motions to train on prevents the policy from overfitting on the dynamics and kinematics of certain locomotion skills, serving as data augmentation, and promoting more general locomotion strategies which can track a rich set of motions. As pointed out in [25], complex skills emerge when trained on a rich set of tasks. We propose a novel adaptive sampling scheme to overcome these challenges. Our key idea is to maintain two sets of the reference motions: $\mathcal{U}$ and $\mathcal{S}$, which represent unsuccessful and successful motions, respectively. At the beginning of learning, we initially assign all the motions to the unsuccessful group $\mathcal{U}$ and set the successful group $\mathcal{S} = \emptyset$. Sampling from these sets is done following a uniform distribution, and without re-drawing so that after several episodes the policy has been trained on the entirety of the sets. For every 200 policy iteration, we evaluate the current policy on all the motions and classify them into each group again based on their performance. If the policy is able to track the motion until the end without early termination, we assign the given motion to the successful group $\mathcal{S}$ (resp. $\mathcal{U}$).

Once the motions are classified into two groups, $\mathcal{U}$ and $\mathcal{S}$, we adjust the sampling of the reference motions. We sample 70 % of the reference trajectories from $\mathcal{U}$, while taking 30 % of trajectories from $\mathcal{S}$. This mechanism allows the policy to majorly focus on difficult reference motions that are yet unsuccessfully learned while not forgetting already learned motions.

IV. Results

A. Implementation Details

We develop the proposed framework using RaiSim [26]. We use the integrated implementation of Proximal Policy Optimization [27] for learning. Our neural network policy has two layers of [256, 256] hidden neurons with LeakyReLu activation functions. We select an A1 quadrupedal robot [28] from Unitree as an experimental platform. We conduct all the experiments using a desktop with AMD Ryzen Threadripper 3970X 32 cores CPU, and RTX 3090. Using AMS a policy trained from scratch on 701 motions takes about 40 hours with 100 environments, 30 threads, and episodes of length 10 seconds. For the motor gains, we choose a proportional gain $k_p = 50.0$ and a derivative gain $k_d = 2.0$ in order to support more stable learning and smoother motions. The entropy coefficient is chosen as $\epsilon = 0.0001$, the policy is queried every $0.02s$ and the motion references are played at a frequency of $1kHz$.

B. Generating Diverse Motions

Our framework is able to learn a single capable policy that can track a large number of trajectories with great diversity, including walking, turning, jumping, sitting, and lying. Using AMS, the policy can successfully track all 701 motions, and $\approx 90\%$ of 47 long random mixed motion clips that are used for validation and to test the ability of our policy to generalize to out-of-distribution motions it has never seen.

An episode length is taken as 10 seconds. Most motions last 10 seconds, but for instance, the star motion lasts 40

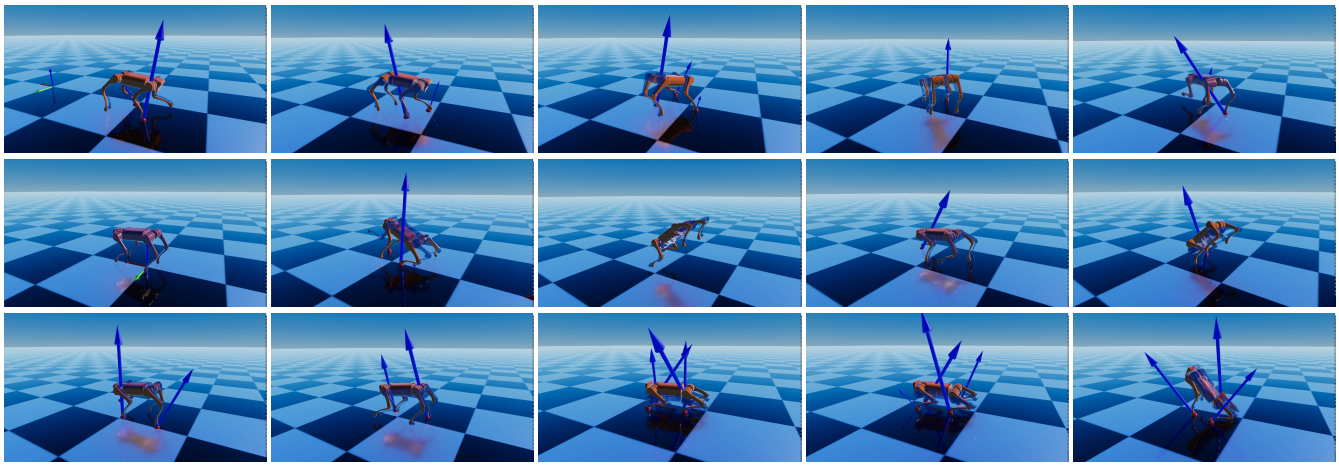

Fig. 5. A variety of generated motions using a single policy. **1st row:** a sharp turn during a star-trajectory tracking task. **2nd row:** multiple jumps in one sequence. **3rd row:** lying down and sit motions. Please refer to the supplemental video for the entire sequences.

seconds. Although the policy is only trained for the first 10 seconds of the motion clip, the policy can successfully track the entire star motion, which supports its generalization capabilities. The policy is able to re-use to some extent the learned locomotion skills to track unseen motion references. This makes the learning faster as training on longer motion clips is not required. Instead, it is possible to train on segments or individual locomotion skills present in a longer motion clip.

The policy can track motions that contain a lot of transitions between skills, and sudden changes in yaw, or speed for instance. Indeed, it is able to track a short clip that involves lying down and sitting to demonstrate generalization over non-locomotion tasks. Finally, we demonstrate that our policy can track a very long sequence that involves many different components, including walking, turning, different gaits, speed changes, and more. Note that it will be very difficult to develop a single control policy to execute all the motor tasks included in our testing sequences. Please refer to Fig. 5 and the supplemental videos for qualitative evaluation. We will also provide more quantitative analysis in the following section.

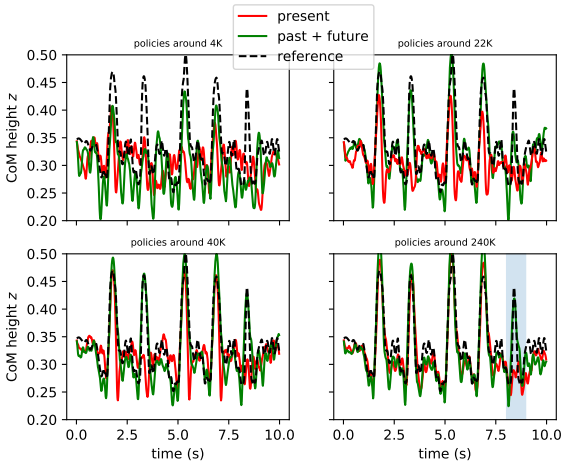

Fig. 6. Influence of the past and future trajectory. Our observation space that includes the past and future enables faster learning: even at the 22K-th policy iteration (top right), our design (green) tracks the reference motion better than the baseline design (red). It also results in a better final policy. At the 240K-th policy iteration (bottom right), our design can track all five jumps while the baseline misses the last jump as highlighted in the blue box centered at around 8.0 seconds.

V. Analysis

A. Influence of Past and Future Target Information in Observations

Inspired by other works [10], [13], the policy is provided with future and past information of the reference to track. The current reference frame is excluded in order to incentivize the agent to interpolate between the different keyframes in order to prevent the policy from overfitting on the low-level kinematic reference data. Fig. 6 shows that this configuration encourages a faster learning and higher quality learned (smoother, more symmetric) joint trajectories. Fig. 6 presents the CoM height tracking for a policy with only the present reference information and a policy with past and future information. First, we find that our design of past and future reference converges faster, as green curves (ours) are closer to the blue-dotted reference motion than red curves (the baseline design with only the current frame) in early policy iterations. Even after a long-enough training with 240k iterations, the policy with the baseline observation design misses the final jump and decides to run through. We hypothesize that past and future trajectory information is critical to plan ahead dynamic jumping motions.

A qualitative comparison is presented in Fig. 7. The baseline observation with only the current frame leads to a reactive behavior that uses its rear legs awkwardly in order to balance. In fact, we observe this behavior consistently over all 200 jumping motions the baseline was trained on. On the other hand, our observation space with past and future reference information finds smooth and natural jumping

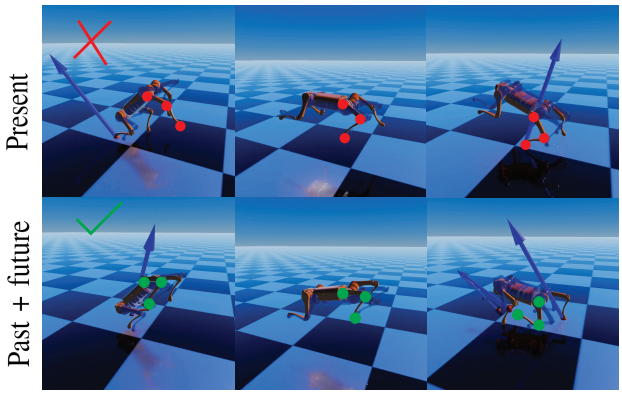

Fig. 7. Selected key frames of a dynamic jump tracking: the fastest jump present in the dataset at a maximum CoM speed of $1.76m/s$. The baseline policy shows reactive and awkward behaviors for balancing (**top**) while our observation design leads to more natural and smooth jumping motions (**bottom**) .

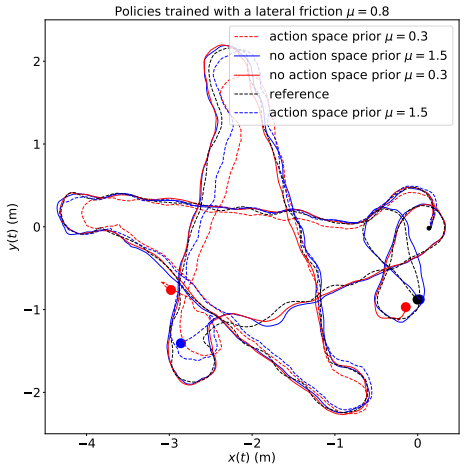

Fig. 8. Generalization over unseen frictions of 0.3 (red) and 1.5 (blue). We examine policies with and without action prior. The policies without action prior (ours, solid lines) show better robustness, while the policy with action prior (baseline, dotted lines) shows larger tracking errors and even cannot complete the sequence. Circles represent the location where the episodes end.

behaviors which can handle even multiple jumps, and even exhibits alterations from the kinematic reference that look as natural as the reference (See Fig. 7).

B. Influence of Action Prior

There exist two common choices of action spaces in the literature of motion imitation. The first is to define it as the delta to the current frame of the reference motion under the expectation that the desirable PD targets are closer to the reference motion (*Action Prior*). The second is to make it independent from the reference motion, such as the delta to the fixed nominal pose (*No Action Prior*, ours). In our experience, a policy without action prior shows much better robustness, particularly when it is combined with our joint-agnostic reward design (Eq. (1)).

Fig. 8 illustrates well the generalization capability of policies over unseen surfaces with low (0.3, red) and high (1.5, blue) lateral friction coefficients (μ), whereas the policy is trained with $\mu = 0.8$. In our experience, learning with *action priors* overfits to track the joint motions and does not generalize well when the robot starts to deviate from the desired trajectory. As a result, the policy exhibits high tracking errors (dotted lines) and even terminates early. On the other hand, our learning formulation without both action prior and joint tracking objective allows the policy to show more robust behaviors to complete complex star-shaped trajectories (solid lines).

C. Adaptive sampling

Finally, this subsection analyzes the importance of our adaptive motion sampling (AMS). In our experience, AMS is critical to cover a large number (~ 700) motions without missing a few outlier motions, such as jumping, lying, and sitting. For instance, we have 200 pace motions while having only 10 of jumping motions. Therefore, naive sampling will likely prioritize pace motions.

We plot (1) the average episodic reward over time and (2) the number of failed motions in Fig. 9. From the perspective of the conventional reward curve (top), it seems that AMS performs suboptimally with slightly lower episodic rewards. However, please note that AMS puts a policy in tougher scenarios by sampling harder tracking problems more often, and we cannot directly compare the reward function. Therefore, we also plot the number of the failed trajectories out of 701 motions at the bottom of Fig. 9, as a more fair comparison criterion. It shows that our AMS fails less over by not ignoring some minority motions.

VI. Conclusion

We present a scalable motion imitation framework to learn a single policy that can track a large variety of motions, including walking, turning, running, jumping, sitting, and lying. Starting from the existing motion imitation framework [9] , we carefully design the observation space, action space, and reward function to improve the effectiveness and robustness of the final policy. In addition, we propose an adaptive motion sampling scheme, which is designed to focus on the learning of more challenging trajectories and to avoid catastrophic forgetting of the previously learned skills. We successfully train a very versatile single policy from a large number of trajectories. We demonstrate that it can also generalize well to novel trajectories to execute a complex, long motion sequence that involves many different motor skills. In addition, we also showcase that the learned policy is robust against the change of environment parameters such as lateral friction. We finally analyze the importance of our problem formulation and adaptive motion sampling by conducting a set of experiments.

There exist several interesting future research directions that we want to explore. For instance, we plan to add

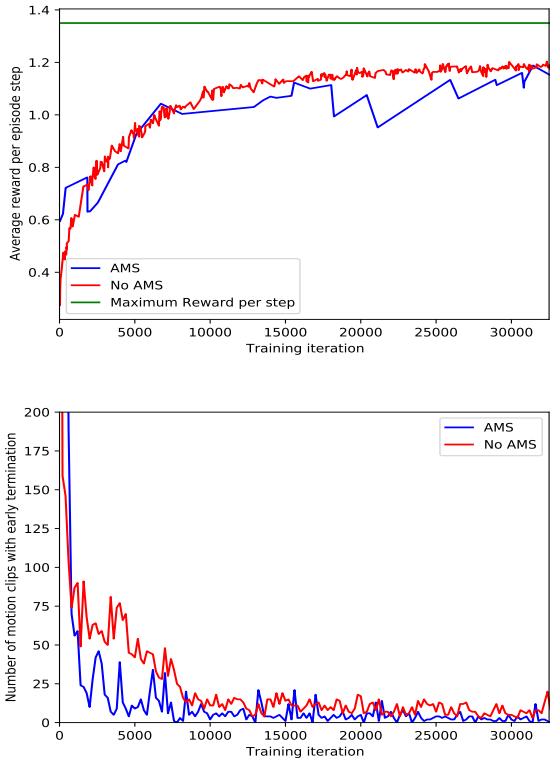

Fig. 9. Adaptive Motion Sampling (AMS): comparison of training results with a policy trained with AMS and one policy without AMS. AMS looks suboptimal in terms of the episodic reward (**top**), but it actually successfully tracks a lot more motions than the baseline without AMS (**bottom**).

more reference motions to the database for non-trivial tasks, such as stair climbing, crawling, and walking over rough terrains. However, the current data generation scheme of motion retargeting will have limitations because it relies on the existing public motion capture data set of a real dog. One possible solution is to add more data using the off-the-shelf trajectory optimization framework [29], which can generate physically valid trajectories for various environments. Once we increase the size of the database, we may need to even further improve the scalability of the current learning framework. It can be approached by adopting more parallelized reinforcement learning algorithms [30], investigating novel policy architecture [31], structuring the dataset [10] or adopting the framework of adversarial learning [11].

Our obvious next step is to deploy the learned policy on the real hardware of the A1 robot. We expect that the learned policy needs to cross a large sim-to-real gap, which can be approached by system identification or domain randomization. However, domain randomization may also increase the difficulty of the problem, and the learning may not converge effectively. In this case, we may want to pretrain a policy without domain randomization and fine-tune the policy in randomized environments.

REFERENCES

[1] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[2] H.-W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the mit cheetah 2: Control design and experiments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017.

[3] 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.

[4] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[5] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

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

[8] 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.

[9] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, jul 2018.

[10] J. Won, D. Gopinath, and J. Hodgins, "A scalable approach to control diverse behaviors for physically simulated characters," *ACM Trans. Graph.*, vol. 39, aug 2020.

[11] 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)*, vol. 40, no. 4, pp. 1–20, 2021.

[12] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions On Graphics (TOG)*, vol. 41, no. 4, pp. 1–17, 2022.

[13] 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.

[14] 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.

[15] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *arXiv preprint arXiv:2205.02824*, 2022.

[16] J. Won, D. Gopinath, and J. Hodgins, "Control strategies for physically simulated characters performing two-player competitive sports," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–11, 2021.

[17] S. Kim, M. Sorokin, J. Lee, and S. Ha, "Human motion control of quadrupedal robots using deep reinforcement learning," *arXiv preprint arXiv:2204.13336*, 2022.

[18] 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.

[19] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–11, 2018.

[20] Unitree, "A1 by unitree robotics." https://www.unitree.com/products/a1.

[21] M. Gleicher, "Retargetting motion to new characters," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 33–42, 1998.

[22] T. Li, J. Won, S. Ha, and A. Rai, "Fastmimic: Model-based motion imitation for agile, diverse and generalizable quadrupedal locomotion," *arXiv preprint arXiv:2109.13362*, 2021.

[23] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4630–4637, apr 2022.

[24] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, vol. 5, dec 2020.

[25] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," *CoRR*, vol. abs/1707.02286, 2017.

[26] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.
[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[28] "Unitree robotics." http://www.unitree.cc/.
[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] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames," *arXiv preprint arXiv:1911.00357*, 2019.
[31] K. N. Kumar, I. Essa, and S. Ha, "Cascaded compositional residual learning for complex interactive behaviors," *arXiv preprint arXiv:2212.08954*, 2022.