DIFFMIMIC: EFFICIENT MOTION MIMICKING WITH DIFFERENTIABLE PHYSICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Motion mimicking is a foundational task in physics-based character animation. However, most existing motion mimicking methods are built upon reinforcement learning (RL) and suffer from heavy reward engineering, high variance, and slow convergence with hard explorations. Specifically, they usually take tens of hours or even days of training to mimic a simple motion sequence, resulting in poor scalability. In this work, we leverage differentiable physics simulators (DPS) and propose an efficient motion mimicking method dubbed **DiffMimic**. Our key insight is that DPS casts a complex policy learning task to a much simpler state matching problem. In particular, DPS learns a stable policy by analytical gradients with ground-truth physical priors hence leading to significantly faster and stabler convergence than RL-based methods. Moreover, to escape from local optima, we utilize an *Demonstration Replay* mechanism to enable stable gradient backpropagation in a long horizon. Extensive experiments on standard benchmarks show that DiffMimic has a better sample efficiency and time efficiency than existing methods (*e.g.*, DeepMimic). Notably, DiffMimic allows a physically simulated character to learn back-flip after 10 minutes of training and be able to cycle it after 3 hours of training, while DeepMimic requires about a day of training to cycle back-flip. More importantly, we hope DiffMimic can benefit more differentiable animation systems with techniques like differentiable clothes simulation in future research. [1] [2]

1 INTRODUCTION

Motion mimicking aims to find a policy to generate control signals for recovering demonstrated motion trajectories, which plays a fundamental role in physics-based character animation, and also serves as a prerequisite for many applications such as control stylization and skill composition. Although tremendous progress in motion mimicking has been witnessed in recent years, existing methods (Peng et al., 2018a; 2021) mostly adopt reinforcement learning (RL) schemes, which require alternatively learning a reward function and a control policy. Consequently, RL-based methods often take tens of hours or even days to imitate one single motion sequence, making their scalability notoriously challenging. In addition, RL-based motion mimicking highly relies on the quality of its designed (Peng et al., 2018a) or learned (Peng et al., 2021) reward functions, which further burdens its generalization for complex real-world applications.

Recently, differential physics simulator (DPS) has achieved impressive results in many research fields, such as robot control (Xu et al., 2022) and graphics (Li et al., 2022). Specifically, DPS treats physics operators as differentiable computational graphs, and therefore gradients from objectives (*i.e.*, rewards) can be directly propagated through the environment dynamics to control policy functions. Instead of alternatively learning between reward functions and control policies, the control policy learning tasks can be resolved in a straightforward and efficient optimization manner with the help of DPS. However, despite their analytical environment gradients, optimization with DPS could easily get into local optima, particularly in contact-rich physical systems that often yield stiff and discontinuous gradients (Freeman et al., 2021; Suh et al., 2022; Zhong et al., 2022). Besides, numerical gradients could also vanish/explode along the backward path for long trajectories.

[1] Our code is available at https://github.com/diffmimic/diffmimic.

[2] Qualitative results can be viewed at https://diffmimic-demo-main-g7h0i8.streamlitapp.com/.

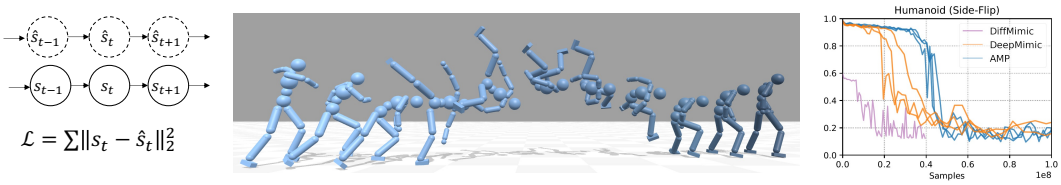


Figure 1: Overview of our method. **Left:** DiffMimic formulates motion mimicking as a straightforward state matching problem and uses analytical gradients to optimize it with off-the-shelf differentiable physics simulators. The formulation results in a simple optimization objective compared to heavy reward engineering in RL-based methods. **Middle:** DiffMimic is able to mimic highly dynamic skills, *e.g.*, Side-Flip. **Right:** DiffMimic has a significantly better sample efficiency and time efficiency than state-of-the-art motion mimicking methods. Our approach usually achieves high-quality motion (pose error < 0.15 meter) using less than 2×10^7 samples.

In this work, we propose DiffMimic, a fast and stable motion mimicking method with the help of DPS. Different from RL-based methods that require heave reward engineering and poor sample efficiency, DiffMimic reformulates motion mimicking as a state matching problem, which could directly minimize the distance between a rollout trajectory generated by the current learning policy and the demonstrated trajectory. Thanks to the differentiable DPS dynamics, gradients of the trajectory distance can be directly propagated to optimize the control policy. As a result, DiffMimic could significantly improve the sample efficiency with the first-order gradients.

However, simply utilizing DPS could not guarantee global optimal solutions. In particular, the rollout trajectory tends to gradually deviate from the expert demonstration and could produce a large accumulative error for long motion sequences, due to the distributional shift between the learning policy and expert policy. To address these problems, we introduce the *Demonstration Replay* training strategy, which randomly inserts reference states into the rollout trajectory as anchor states to guide the exploration of the policy. Empirically, Demonstration Replay gives a smoother gradient estimation, which significantly stabilizes the policy learning of DiffMimic.

To the best of our knowledge, DiffMimic is the first to utilize DPS for motion mimicking. We show that DiffMimic outperforms several commonly used RL-based methods for motion mimicking on a variety of tasks with high accuracy, stability, and efficiency. In particular, DiffMimic allows learning a challenging *back-flip* motion in only 10 minutes on a single V100 GPU. In addition, we release the DiffMimic simulator as a standard benchmark to encourage future research for motion mimicking.

2 Related Work

Motion Mimicking. To produce realistic animation, motion mimicking is employed to learn skills from motion captures (Hong et al., 2019; Peng et al., 2018a;b; Lee et al., 2019) for physics-based characters. Motion mimicking enables various downstream tasks in physics-based animation. Learned primitive actions can be re-combined to generate various motions (Peng et al., 2019; Luo et al., 2020), used to achieve a goal (Bergamin et al., 2019; Park et al., 2019; Peng et al., 2021), or leveraged as a pre-train model for general-purpose motor skills (Merel et al., 2018; Hasenclever et al., 2020; Peng et al., 2022; Won et al., 2022). While these downstream tasks look exciting and promising, the scalability can be largely limited by motion mimicking. In our work, we show the problem can be surprisingly easy with differentiable dynamics.

Speeding Up Motion Mimicking Most motion mimicking works are based on a DRL framework (Peng et al., 2018a; Bergamin et al., 2019), whose optimization is expensive. Several recent works speed up the DRL process by hyper-parameter searching (Yang & Yin, 2021) and constraint relaxation (Ma et al., 2021). Another line of work learns world models to achieve end-to-end gradient optimization (Won et al., 2022; Fussell et al., 2021). However, learning world models requires extra training and introduces the risk of error accumulation. We are the first to use the off-the-shelf DPS for motion mimicking, which achieves better sample efficiency than optimized DRL frameworks and requires no additional world models.

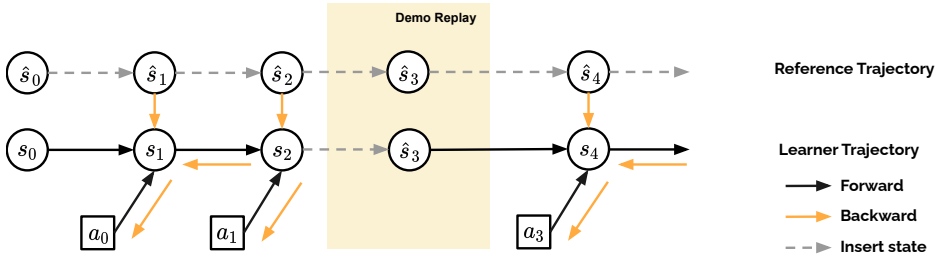


Figure 2: Computation graph of DiffMimic. An example of demonstration replay during the third step is demonstrated. The demonstration state in the third step is used to replace the simulated state.

Differentiable Physics Simulators. The emerging differentiable physics simulators (DPS) (Hu et al., 2019; 2020; Huang et al., 2021; Qiao et al., 2021) have brought success in many domains and provide a distinctly new approach to control tasks. Various applications stem from DPS like system identification (Jatavallabhula et al., 2021), continuous controls (Lin et al., 2022; Xu et al., 2022) in the past few years and demonstrate promising results. Compared to black-box counterparts which learn the dynamics with neural networks (Fussell et al., 2021; Li et al., 2020), differentiable simulations utilize physical models to provide more reliable gradients with better interpretability. On the other hand, the analytical gradient from DPS suffers from noise or can sometimes be wrong (Zhong et al., 2022; Suh et al., 2022) in a contact-rich environment. Non-smoothness or discontinuity in the contact event requires specific techniques to compute the gradient. However, they often introduce noise into the gradient (Zhong et al., 2022). Thus, in this work, we empirically show how to handle the noise in the gradient in a contact-rich environment for motion mimicking.

Policy Learning with Differentiable Physics Simulators. The recent development of differentiable simulators enables the optimization of control policies via analytical gradient (Mozer, 1989; Mora et al., 2021; Xu et al., 2022). Concurrent to this work, ILD (Chen et al., 2022) proposes a simple yet effective Imitation Learning framework with DPS for robot control. While the first-order gradient largely enhances the sample efficiency for policy learning, challenges lie ahead. The noisy gradient and exploding/vanishing gradients are two crucial issues that hinder the wide deployment of analytical gradient-based reinforcement learning (Degrave et al., 2017). Due to the local nature of the gradient, the policy could be stuck at sub-optimal points. In particular, we observe that ILD-like optimization fails to let characters explore highly dynamic motions. We introduce a simple replay mechanism to alleviate these issues.

3 Methodology

3.1 A Motion Mimicking Environment in Differentiable Physics Engine.

Environment. We build our environment in Brax (Freeman et al., 2021). We design our simulated character following DeepMimic (Peng et al., 2018a). The humanoid character has 13 links and 34 degrees of freedom. The character has a weight of 45 kg and a height of 1.62m. Contact is applied to all links with the floor. The environment is accelerated by GPU parallelization. The physics simulator updates at a rate of 480 FPS. The joint limits of the character are relaxed to allow smoother gradient propagation. We keep the system configurations like friction coefficients consistent with DeepMimic.

State and Action. States include the position p, rotation q, linear velocity $\dot{p}$ and angular velocity $\dot{q}$ of all links in the local coordinate. Additionally, following Peng et al. (2018a), a phase variable ϕ in the range $[0, 1]$ is included in the state to serve as a timestamp. Thus, the state of the environment, s, contains the aforementioned information, $s := \{p, q, \dot{p}, \dot{q}, \phi\}$. We follow the common practice and use PD controllers to drive the character. Given the current joint angle q, angular velocity $\dot{q}$ and a target angle $\tilde{q}$, the torque on the actuator of the joint will be computed as:

$$\tau = k_p(\tilde{q} - q) + k_d(\tilde{\dot{q}} - \dot{q}), \tag{3.1}$$

where $\tilde{\dot{q}} = 0$, k_p and k_d are manually-specified gains of the PD controller. We keep k_p and k_d identical to the PD controller used in DeepMimic. A policy network predicts the target angle for the PD controller at each joint. The control policy operates at 30 FPS.

3.2 Motion Mimicking with Differentiable Physics

Motion Mimicking is about matching the policy rollout with the reference motion. While it has a straightforward objective, considering human action can be rich and highly flexible, designing a reward to incentivize and guide policy learning is not easy. For example, reward functions that are tailored to guide a policy to learn to walk or learn to back-flip can be different and this makes reward engineering difficult. We reveal that the mimicking task can be surprisingly easy when analytical gradients can be obtained through DPS.

Algorithm 1 DiffMimic

Input: Optimization Iteration I, Episode Length T, Reference States $\hat{S} = \{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_t\}$, Error Threshold ϵ.
Output: Optimized policy
Initialize the stochastic policy as π_θ.
for optimization iteration $i = 1 \cdots I$ **do**
  # Roll out trajectories with Expert Replay
  Initialize $s_1 \leftarrow \hat{s}_1$
  for step $t = 1 \cdots T-1$ **do**

$$s_{t+1} = \begin{cases} \mathcal{T}(s_t, a_t),\ a_t \sim \pi_\theta(a|s_t) & \text{if } \|s_t - \hat{s}_t\|_2^2 < \epsilon \\ \mathcal{T}(\hat{s}_t, a_t),\ a_t \sim \pi_\theta(a|\hat{s}_t) & \text{otherwise.} \end{cases}$$

  end for
  Compute loss $\mathcal{L} = \sum_{t=1}^{T} \|s_t - \hat{s}_t\|_2^2$
  Update the policy π_θ with analytical gradient $\bigtriangledown_\theta \mathcal{L}$.
end for
Return the policy π_θ.

The underlying idea of DiffMimic is to allow the gradient to directly flow back from the distance between reference motion and policy rollout to the policy through an off-the-shelf DPS. We show the computation graph of DiffMimic in Fig. 2. Compared with existing RL-based approaches, DiffMimic directly optimizes the policy with supervised state-matching signals and avoids reward engineering in any form. Thanks to the analytical gradient, such an approach has a very high sample efficiency. Prior works (Fussell et al., 2021) explored analytic gradients with a learned differentiable world model which describes the dynamics in latent space. However, the learned world model reveals no underlying physics of the system. Consequently, they undermine the effectiveness, generality, and interpretability of the learning with physics-based analytic gradients from DPS, as in DiffMimic. In addition, online learning and inference with learned world models remain computationally expensive when compared with DPS.

We show in detail how to train the policy with DPS. For each iteration, we initialize the state to the first reference state and then roll out to the maximum episode length in a batch of parallel environments. Then we compute the step-wise L2 distance between the rollout trajectory and the reference trajectory in terms of positions and rotations:

$$\mathcal{L} = \sum_{t=1}^{T} \|s_t - \hat{s}_t\|_2^2, \quad \|s_t - \hat{s}_t\|_2^2 \triangleq \frac{1}{\|J\|} \sum_{j \in J} (p^j - \hat{p}^j)^2 + (q^j - \hat{q}^j)^2, \tag{3.2}$$

where p^j and $\hat{p}^j$ are the global positions of the j-th joint from rollout and reference, q^j and $\hat{q}^j$ are the global rotation of the j-th joint from rollout and reference in 6D rotation representation. Note that we use 6D rotation in loss computation since it is advantageous over quaternions in gradient-based optimization (Zhou et al., 2019). Without loss of generality, we denote the transition function in the dynamic system as $\mathcal{T}$ and the next state of the system can be obtained from the transition function given the current action a_t and character state s_t, $s_{t+1} = \mathcal{T}(s_t, a_t)$. In DiffMimic, DPS serves as a transition function $\mathcal{T}$ that is fully differentiable. Thus, from the loss function, we can directly derive the gradient with respect to both the current action a_t and state s_t

$$\frac{\partial \mathcal{L}}{\partial a_t} = \left(\frac{\partial \mathcal{L}}{\partial \mathcal{T}(s_t, a_t)}\right)\left(\frac{\partial \mathcal{T}(s_t, a_t)}{\partial a_t}\right), \quad \frac{\partial \mathcal{L}}{\partial s_t} = \left(\frac{\partial \mathcal{L}}{\partial \mathcal{T}(s_t, a_t)}\right)\left(\frac{\partial \mathcal{T}(s_t, a_t)}{\partial s_t}\right). \tag{3.3}$$

By doing this recursively, the gradient can be propagated through the whole trajectory.

3.3 Demonstration Replay

Although mimicking with DPS results in a succinct learning framework, there are three well-known challenges in policy learning with DPS: 1) Exploding/vanishing gradients with the long horizon; 2) Local minima may cause gradient-based optimization methods to stall; 3) Noisy or wrong gradients.

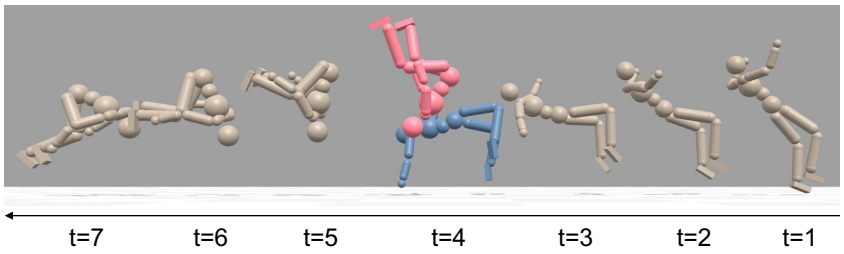


Figure 3: Illustration of Demonstration Replay, which happens when t=4 where the character is about to fall. The rollout state (in blue) is replaced by the reference state (in red) for the next step.

We observe that the high non-convexity in the motion mimicking task poses severe challenges to analytical gradient-based optimization. Due to the local nature of the analytical gradients, the policy can be easily stuck at suboptimal points. As shown Tab. 6, when trying to mimic the *Back-Flip* skill, the character learns to use arms to support itself instead of exploring a more dynamic jumping move. In practical implementations, dividing the whole trajectory into smaller sub-trajectories for truncated Back-Propagation-Through-Time (BPTT) further exacerbates the issue. As shown in Tab. 7 (a)-(b), a 10-step gradient truncation leads the policy to a less favorable local optimal.

Teacher forcing (Williams & Zipser, 1989) is a common remedy to escape the local minimum in gradient-based sequential modeling. Teacher forcing randomly replace the states in rollout by the reference states. Given a teacher forcing ratio γ:

$$s_{t+1} = \begin{cases} \mathcal{T}(s_t, a_t),\ a_t \sim \pi_\theta(a|s_t) & \text{if } b = 0, b \sim \text{Bernoulli}(\gamma) \\ \mathcal{T}(\hat{s}_t, a_t),\ a_t \sim \pi_\theta(a|\hat{s}_t) & \text{otherwise.} \end{cases} \tag{3.4}$$

However, although the mechanism leads to a better global optimal overall, it does not guarantee the character to faithfully mimic the reference motion in each frame. As shown in Tab. 6, although the character's overall trajectory matches the reference, several frames suffer from awkward poses. Fig. 8 shows that a few frames have significantly larger pose errors than other frames.

We propose demonstration-guided exploration to mitigate these challenges in DPS for policy learning to help exploration and encourage faithfully mimicking at the same time. The main idea of demonstration-guided exploration is to replace states in the policy rollout with the corresponding demonstration states when the states are too far away from the reference. For a threshold ϵ:

$$s_{t+1} = \begin{cases} \mathcal{T}(s_t, a_t),\ a_t \sim \pi_\theta(a|s_t) & \text{if } \|s_t - \hat{s}_t\|_2^2 < \epsilon \\ \mathcal{T}(\hat{s}_t, a_t),\ a_t \sim \pi_\theta(a|\hat{s}_t) & \text{otherwise.} \end{cases} \tag{3.5}$$

Since the criterion for choosing which states to replace is based on the performance of the current rollout, the replacing frequency dynamic adjusts itself during the model training.

4 Experiments

4.1 Experiment Setup

For all the experiments, we run the algorithm with one single GPU (NVIDIA Tesla V100) and CPU (Intel Xeon E5-2680). Following Peng et al. (2021), we use the average pose error as the main metric. The average pose error over the whole trajectory of length T with J joints is computed between the pose of the simulated character and the reference motion using the relative positions of each joint with respect to the root joint (in units of meters):

$$e = \frac{1}{T} \sum_{t \in T} \frac{1}{\|J\|} \sum_{j \in J} \|(p_t^j - p_t^{root}) - (\hat{p}_t^j - \hat{p}_t^{root})\|_2, \tag{4.1}$$

where p_t^j and $\hat{p}_t^j$ are the position of j-th joint of the simulated motion and reference motion at timestamp t in the 3D cartesian space and $root$ refers to the root joint. We mainly compare with

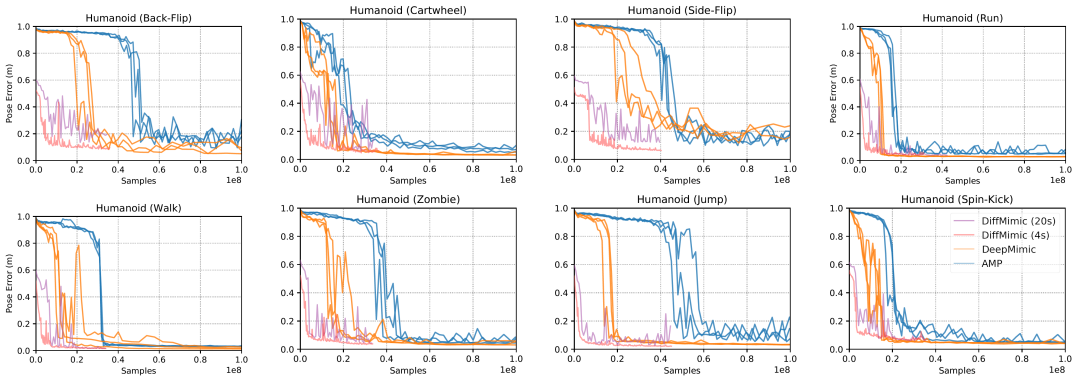


Figure 4: Pose error versus the number of samples. DiffMimic (4s): rollout 4 seconds of Diffmimic for evaluation. DiffMimic (20s): rollout 20 seconds of Diffmimic for evaluation. In general, DiffMimic allows the policy to generate high-quality motions with less than 3.5×10^7 samples. We refer the readers for more results to the appendix.

DeepMimic (Peng et al., 2018a), Spacetime Bound (Ma et al., 2021), and Adversarial Motion Prior (AMP) (Peng et al., 2021). The Dynamic Time Warping (Sakoe & Chiba, 1978) is applied to synchronize the simulated motion and the reference motion following the convention.

Cyclic Motions. Besides mimicking a single motion clip, a popular benchmark for motion mimicking is to imitate cyclic motions like walking and running, where a single cycle of the motion is provided as the reference motion and the character learns to repeat the motion for 20 seconds. Typically, a curriculum is designed to gradually increase the maximum rollout episode to 20 seconds during training (Yang & Yin, 2021; Peng et al., 2018a). In our experiment, we remove all bells and whistles and fix the maximum rollout episode to 4 seconds during the training. DiffMimic can produce a 20-second long cyclic rollout even though it only sees 4-second rollouts in training. The motion clips are directly borrowed from AMP Peng et al. (2021), which are originally collected from a combination of public mocap libraries, custom recorded mocap clips, and artist-authored keyframe animations.

4.2 Comparison on Motion Mimicking

In this section, we aim to understand the 1) efficiency; 2) quality of the learned policy of DiffMimic. Following the conventions of previous works (Peng et al., 2018a; 2021; Ma et al., 2021), we count the number of samples required to train the policy to rollout for 20 seconds without falling. The pose error is calculated over the horizon of 20 seconds.

Analytical gradients enhance the sample efficiency in motion mimicking. We show the comparison between DiffMimic, DeepMimic, and Spacetime Bound on sample efficiency in Table 2. DeepMimic is an RL-based algorithm with a careful reward design. Spacetime Bound performs hyperparameter searching for DeepMimic to further enhance the sample efficiency. Our results show that DiffMimic constantly outperforms DeepMimic in terms of sample efficiency. The analytical gradients provided by the differentiable simulation allow us to compute policy gradient with a small number of samples while the RL-based algorithm requires a large batch to have a decent estimate. Compared with Spacetime Bound, DiffMimic is much more stable and consistent over various tasks. We notice that Spacetime Bound may require more samples than DeepMimic even for simple tasks like *Jump*. We show the learning curve of DiffMimic over eight different tasks in Fig. 4. It shows that DiffMimic generally learns

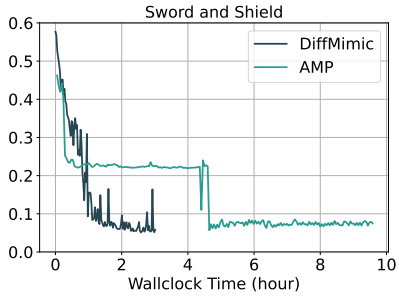


Figure 5: Wallclock training time versus pose error for DiffMimic and AMP. DiffMimic takes half of training time required by AMP to have a comparable result.

Table 1: Pose error comparison in meters. T_{cycle} is the length of the reference motion for a single cycle. The error is averaged on 32 rollout episodes with a maximum length of 20 seconds.

Motion	T_{cycle}(s)	DeepMimic	AMP	Ours
Back-Flip	1.75	0.076 ± 0.021	0.150 ± 0.028	0.097 ± 0.001
Cartwheel	2.72	0.039 ± 0.011	0.067 ± 0.014	0.040 ± 0.007
Crawl	2.93	0.044 ± 0.001	0.049 ± 0.007	0.037 ± 0.001
Dance	1.62	0.038 ± 0.001	0.055 ± 0.015	0.070 ± 0.003
Jog	0.83	0.029 ± 0.001	0.056 ± 0.001	0.031 ± 0.002
Jump	1.77	0.033 ± 0.001	0.083 ± 0.022	0.025 ± 0.000
Roll	2.02	0.072 ± 0.018	0.088 ± 0.008	0.061 ± 0.007
Run	0.80	0.028 ± 0.002	0.075 ± 0.015	0.039 ± 0.000
Side-Flip	2.44	0.191 ± 0.043	0.124 ± 0.012	0.069 ± 0.001
Spin-Kick	1.28	0.042 ± 0.001	0.058 ± 0.012	0.056 ± 0.000
Walk	1.30	0.018 ± 0.005	0.030 ± 0.001	0.017 ± 0.000
Zombie	1.68	0.049 ± 0.013	0.058 ± 0.014	0.037 ± 0.002

Table 2: Number of samples required to roll out 20 seconds without falling in (10^6). Percentage: change in fraction of the DeepMimic samples.

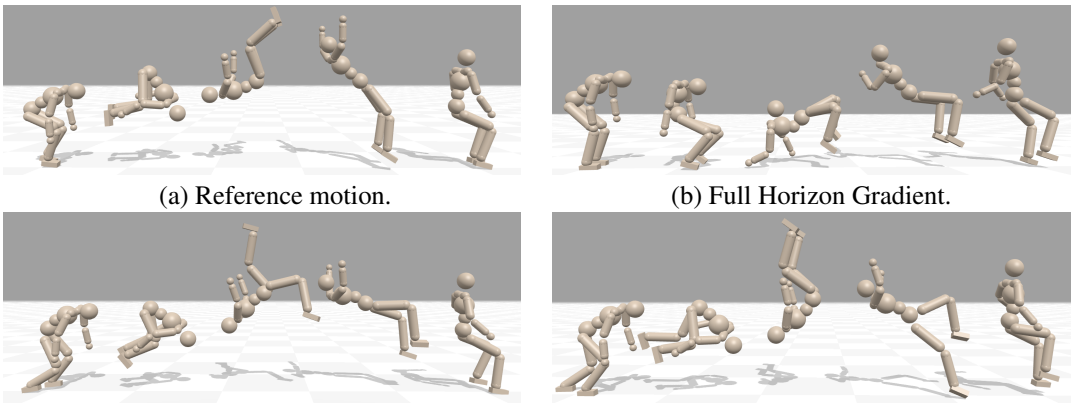
Motion	T_{cycle}(s)	DeepMimic	Spacetime Bound	Ours
Back-Flip	1.75	31.18	41.20 +32.1%	14.88 -52.2%
Cartwheel	2.72	30.45	17.35 -43.0%	13.92 -54.2%
Walk	1.25	23.80	4.08 -79.5%	7.92 -66.7%
Run	0.80	19.31	4.11 -78.7%	8.16 -57.7%
Jump	1.77	25.65	41.63 +77.8%	5.28 -79.4%
Dance	1.62	24.59	10.00 -59.3%	16.56 -32.6%

high-quality motions (pose error < 0.15) with less than 2e7 samples, even for challenging tasks like *Back-flip*. In terms of the wallclock time, DiffMimic learns to perform *Back-flip* in 10 minutes and learns to cycle it in 3 hours (14.88×10^6 samples), which can be found in our qualitative results.

DiffMimic can learn high-quality motions. The average pose error for 12 different motion tasks is shown in Table 1. DiffMimic outperforms AMP consistently and has comparable performance to DeepMimic. Noticeably, DiffMimic only needs to see the demonstrations of 4 seconds to achieve a similar performance of DeepMimic with 20 seconds cyclic rollout, which indicates stable and faithful recovery of the reference motion. This further corroborates the efficacy of DiffMimic. *Analytical gradients accelerate the learning speed in motion mimicking.* As it generally takes more time to compute analytical gradients than the estimated one, it is natural to compare the wall-clock time for each algorithm. Since the implementation of DeepMimic does not utilize GPU for parallelization, we choose to compare DiffMimic with AMP. The implementation of AMP utilizes the highly parallelized environment, IsaacGym (Makoviychuk et al., 2021). Both approaches do not require manual reward design and utilize GPU for acceleration. We compare DiffMimic and AMP on a task where a humanoid character is wielding a sword and a sheild to perform a 3.2-second attack move. The hyperparameter of AMP is from its official release of code. We show the comparison with respect to wallclock time in Fig. 5. The simulated motion is demonstrated in Fig. 9. DiffMimic takes half of training time required by AMP to have a comparable result.

4.3 Ablation on Truncation Length

Training the policy with DPS would suffer from the vanishing/exploding gradients and local optimal. Recent research (Xu et al., 2022) points out that such a problem can be mitigated by a truncated learning window, which splits the entire trajectory into segments with shorter horizons. we carried



(a) Reference motion. (b) Full Horizon Gradient.

(c) Demonstration Replay (Random). (d) Demonstration Replay (Threshold).

Figure 6: Qualitative results of three variants of DiffMimic with respect to demonstration replay. The policy trained with full horizon gradient fails to jump up. The policy trained with Demonstration Replay (Random) fails to recover the reference motion faithfully.

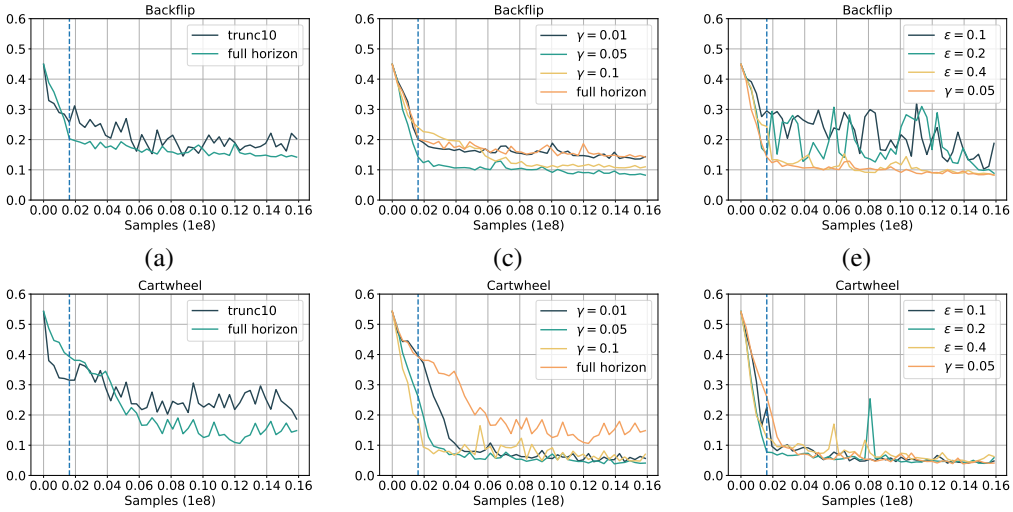


Figure 7: (a)-(b) Comparison between Full Horizon Gradient and truncation length of 10. (c)-(d) Comparison between Demonstration Replay (Random) and Full Horizon Gradient. (e)-(f) Comparison between Demonstration Replay (Random) and Demonstration Replay (Threshold). The blue dotted line denotes 10 minutes in corresponding wallclock time. (Order changed.)

out experiments to validate whether such an idea can be directly applied in motion mimicking with DPS. Truncation length refers to the horizon over which the gradient is calculated. The quantitative results are shown in Fig. 7. Indeed, it is difficult to learn a good policy by propagating the gradient through the whole trajectory that is long. In addition, simply splitting the whole trajectory into segments would worsen the final performance. We hypothesize the naive truncation of the trajectory creates discontinuities in the whole trajectory whereas motions in the trajectory are highly interdependent. For example, how to flip in mid-air closely relates to how the character jumps. We see this as a strong call for a better strategy to handle these two challenges in motion mimicking with DPS.

4.4 Ablation on Demonstration Replay

To understand how demonstration replay affects the policy learning of DiffMimic, we compare three different variants of DiffMimic, namely, Demonstration Replay (Random) and Demonstration Replay (Threshold), and Full Horizon Gradient on *Back-flip* and *Cartwheel*. Demonstration Replay

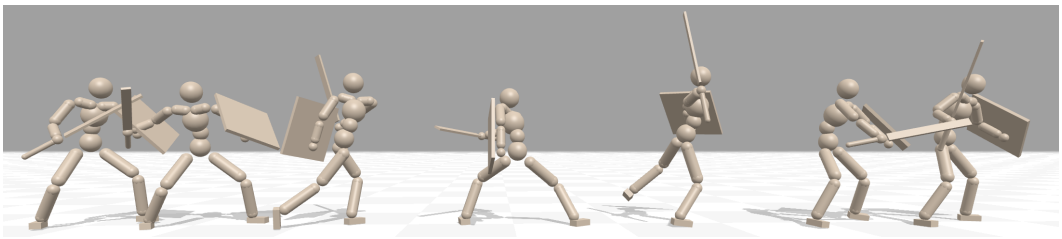


Figure 9: Character performing attack move with a sword and a shield. The reference motion clip is from Reallusion (Reallusion, 2022)

(Random) randomly replaces a state in the rollout of the policy with the demonstration state at the same timestamp similar to the teacher forcing (Williams & Zipser, 1989). Demonstration Replay (Threshold) inserts the demonstration state based on the pose error. If the pose error between the demonstration state and simulated state exceeds a threshold ϵ, the simulated state will be replaced by the demonstration state. The Full Horizon Gradient variant backpropagates the gradient through the full horizon without any additional operation. We show the quantitative result in Fig. 7, and qualitative result in Fig. 6.

Demonstration replay helps stabilize policy learning and leads to better performance. Compared with the Full Horizon Gradient without demonstration replay, the learning curve with demonstration replay is much smoother and finally converges to a lower pose error as shown in Fig. 7 (c)-(d). We show in Fig. 6 (b), this vanilla version of DiffMimic can easily get stuck into a local minimum. Instead of learning to back-flip, the policy learns to bend down and use arms to support itself instead of jumping up. The other two variants, by contrast, both learn to jump and back-flip in mid-air successfully.

Policy-aware demonstration replay leads to a more faithful recovery of demonstration. We compare Demonstration Replay (Threshold) with Demonstration Replay (Random) in Fig. 7 (e)-(f). Quantitatively, both variants yield similar results if the hyperparameter is properly tuned. However, the behavior of policies trained with these two strategies can be significantly different. In Fig. 6 (c) and (d), though both policies learn to back-flip successfully, the policy trained with Demonstration Replay (Random) fails to recover the motion frame by frame. We show in Fig. 8 the per-frame pose error. Even the average error over the whole trajectory for both variant are similar, Demonstration Replay (Threshold) gives a lower maximum per-frame error, which indicates a faithful recovery of the demonstration. This implies that simply minimizing the pose error may not suffice to learn a policy that tracks the demonstration closely. Finer-grained guidance based on the current performance of the policy is required.

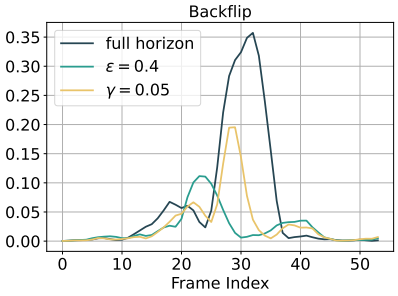


Figure 8: L2 error per frame in the rollout of the final policy. Although Demonstration Replay (Random) reduces the overall pose error compared to full-horizon optimization, the per-frame loss remains large in certain time steps due to the lack of a constraint. Demonstration Replay (Threshold) alleviates the issue.

5 Conclusion & Future works

To conclude, we present DiffMimic that utilize DPS for motion mimicking. We show that DiffMimic outperforms several commonly used RL-based methods for motion mimicking on a variety of tasks with high accuracy, stability, and efficiency. We hope DiffMimic can be considered as a starting point for motion mimicking with differentiable physics simulators and our simulation environments bring additional future research opportunities to the community. There are enormous exciting future directions to be explored, yet, with various challenges. Given differentiable physics largely accelerates motion mimicking, it would be much more valuable for downstream tasks if motion mimicking can be carried out within minutes given any arbitrary demonstration. In this work, we demonstrate that DiffMimic can learn challenging motions in 10 minutes. Yet, the horizon of the task is relatively short and does not require any interaction with other objects. The dynamic system would get more complex if multiple objects are involved. We leave them for future work.

ETHICS STATEMENTS

We have read the ICLR Code of Ethics. While DiffMimic carries out experiments on humanoid characters, no human-subject experiment is involved and all experiments are conducted with open-sourced materials with references to them.

REPRODUCIBILITY STATEMENTS

Our code is available at https://github.com/diffmimic/diffmimic. We promise to clean and release the code upon publication.

REFERENCES

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG)*, 38(6): 1–11, 2019.

Siwei Chen, Xiao Ma, and Zhongwen Xu. Imitation learning via differentiable physics. *arXiv preprint arXiv:2206.04873*, 2022.

Jonas Degrave, Michiel Hermans, Joni Dambre, and Francis Wyffels. A differentiable physics engine for deep learning in robotics. *Frontiers in Neurorobotics*, 13, 2017.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Levi Fussell, Kevin Bergamin, and Daniel Holden. Supertrack: Motion tracking for physically simulated characters using supervised learning. *ACM Transactions on Graphics (TOG)*, 40(6): 1–13, 2021.

Leonard Hasenclever, Fabio Pardo, Raia Hadsell, Nicolas Heess, and Josh Merel. Comic: Complementary task learning & mimicry for reusable skills. In *International Conference on Machine Learning*, pp. 4105–4115. PMLR, 2020.

Seokpyo Hong, Daseong Han, Kyungmin Cho, Joseph S Shin, and Junyong Noh. Physics-based full-body soccer motion control for dribbling and shooting. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B. Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=xCcdBRQEDW`.

Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. *International Conference on Learning Representations (ICLR)*, 2021. URL `https://openreview.net/forum?id=c_E8kFWfhp0`.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)*, 38(4):1–13, 2019.

Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Trans. Graph.*, mar 2022. ISSN 0730-0301. doi: 10.1145/3527660. URL https://doi.org/10.1145/3527660. Just Accepted.

Yunzhu Li, Toru Lin, Kexin Yi, Daniel Bear, Daniel L.K. Yamins, Jiajun Wu, Joshua B. Tenenbaum, and Antonio Torralba. Visual grounding of learned physical models. In *International Conference on Machine Learning*, 2020.

Xingyu Lin, Zhiao Huang, Yunzhu Li, Joshua B. Tenenbaum, David Held, and Chuang Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. 2022.

Ying-Sheng Luo, Jonathan Hans Soeseno, Trista Pei-Chun Chen, and Wei-Chao Chen. Carl: Controllable agent with reinforcement learning for quadruped locomotion. *ACM Transactions on Graphics (TOG)*, 39(4):38–1, 2020.

Li-Ke Ma, Zeshi Yang, Xin Tong, Baining Guo, and KangKang Yin. Learning and exploring motor skills with spacetime bounds. In *Computer Graphics Forum*, volume 40, pp. 251–263. Wiley Online Library, 2021.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=fgFBtYgJQX_.

Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. Hierarchical visuomotor control of humanoids. *arXiv preprint arXiv:1811.09656*, 2018.

Miguel Angel Zamora Mora, Momchil Peychev, Sehoon Ha, Martin Vechev, and Stelian Coros. Pods: Policy optimization via differentiable simulation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7805–7817. PMLR, 18–24 Jul 2021.

Michael C. Mozer. A focused backpropagation algorithm for temporal pattern recognition. *Complex Syst.*, 3, 1989.

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)*, 38(6):1–11, 2019.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018a.

Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions On Graphics (TOG)*, 37(6): 1–14, 2018b.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems*, 32, 2019.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):1–20, 2021.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *arXiv preprint arXiv:2205.01906*, 2022.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming Lin. Differentiable simulation of soft multi-body systems. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=j3fpZLKcXF.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Reallusion. 3d animation and 2d cartoons made simple, 2022.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

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

Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20668–20696. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/suh22b.html.

Nolan Wagener, Andrey Kolobov, Felipe Vieira Frujeri, Ricky Loynd, Ching-An Cheng, and Matthew Hausknecht. Mocapact: A multi-task dataset for simulated humanoid control. *arXiv preprint arXiv:2208.07363*, 2022.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 39(4): 33–1, 2020.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Physics-based character controllers using conditional vaes. *ACM Transactions on Graphics (TOG)*, 41(4):1–12, 2022.

Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. *arXiv preprint arXiv:2204.07137*, 2022.

Zeshi Yang and Zhiqi Yin. Efficient hyperparameter optimization for physics-based character animation. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(1):1–19, 2021.

Yaofeng Desmond Zhong, Jiequn Han, and Georgia Olympia Brikis. Differentiable physics simulations with contacts: Do they have correct gradients w.r.t. position, velocity and control? In *ICML 2022 2nd AI for Science Workshop*, 2022. URL https://openreview.net/forum?id=KIl0LZ9tJex.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019.

CONTENTS

1 Introduction **1**

2 Related Work **2**

3 Methodology **3**

3.1 A Motion Mimicking Environment in Differentiable Physics Engine. 3

3.2 Motion Mimicking with Differentiable Physics 4

3.3 Demonstration Replay 4

4 Experiments **5**

4.1 Experiment Setup 5

4.2 Comparison on Motion Mimicking 6

4.3 Ablation on Truncation length 7

4.4 Ablation on Demonstration Replay 8

5 Conclusion & Future works **9**

A Qualitative Results **14**

B Policy Network **14**

C Hyper-parameters **14**

D State and Action Space **14**

E Learning Curves **15**

F New Character **16**

G New Skill **16**

H New Training Strategy **16**

I Analysis on Robustness **17**

J Analysis on Sensitivity **18**

K Analysis on Motion Quality **18**

L More Discussions on DeepMimic Reward **18**

M More Discussions on Related Works **19**

A QUALITATIVE RESULTS

We present our qualitative results in the following website: https://diffmimic-demo-main-g7h0i8.streamlitapp.com/.

It contains the following qualitative results:

- Back-Flip 20-second rollout.
- Cartwheel 20-second rollout.
- Crawl 20-second rollout.
- Dance 20-second rollout.
- Jog 20-second rollout.
- Jump 20-second rollout.
- Roll 20-second rollout.
- Side-Flip 20-second rollout.
- Spin-Kick 20-second rollout.
- Walk 20-second rollout.
- Zombie 20-second rollout.
- Back-Flip 20-second rollout, trained for 2.5 hours.
- Back-Flip single cycle rollout, trained for 10 minutes.

B POLICY NETWORK

We use an MLP for the policy network. The network has two hidden layers with sizes 512 and 256. We use Swish (Ramachandran et al., 2017) for the activation function.

C HYPER-PARAMETERS

We use Adam optimizer (Kingma & Ba, 2014) in training, with a learning rate of 3e-4 that linearly decreases with training iterations. We apply gradient clipping and set the maximum gradient norm to 0.3. We set the batch size to the same as the number of parallel environments.

Cyclic motions. We set the maximum iterations to 5000. The number of environments is set to 200.

Acyclic motions. We set the maximum iterations to 1000. The number of environments is set to 300.

Expert Replay. We run experiments with two error thresholds, 0.2 and 0.4, and report the better of the two.

D STATE AND ACTION SPACE

Humanoid. We use 6d rotation representation (Zhou et al., 2019) in the state feature. The humanoid has an action size of 28 and a state feature size of 193.

Humanoid with Sword and Shield. We set up the Sword and Shield character following ASE (Peng et al., 2022). Compared to the default humanoid, the new character has 3 additional DOF on the right hand. A sword is attached to its right hand and a shield is attached to its left lower arm. The character has an action size of 31. The state feature size is 208.

E LEARNING CURVES

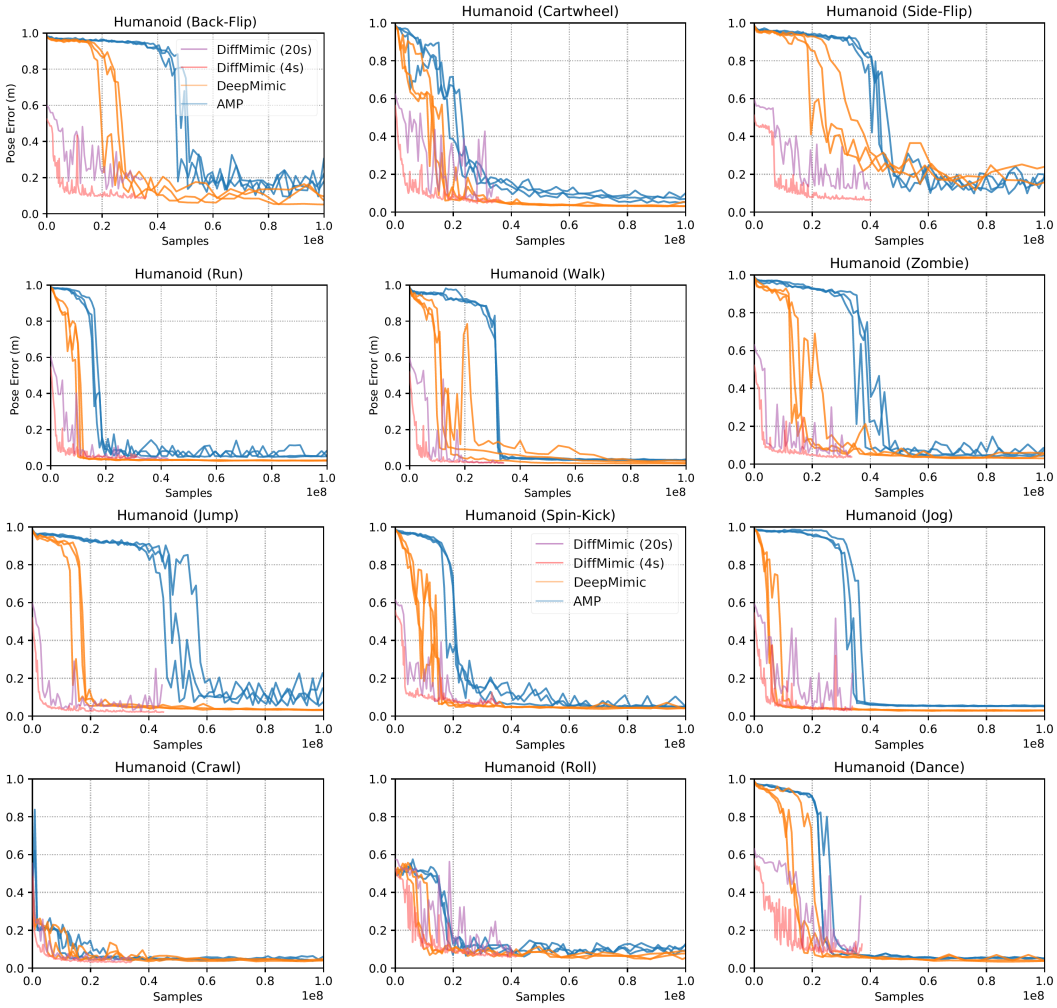


Figure 10: Pose error versus the number of samples. DiffMimic (4s): rollout 4 seconds of Diffmimic for evaluation. DiffMimic (20s): rollout 20 seconds of Diffmimic for evaluation.

F NEW CHARACTER

We include Ant as a new character. We use the trajectory in ILD (Chen et al., 2022) as the reference motion, which is obtained from a PPO (Schulman et al., 2017) agent that learns to move forward as fast as possible. We achieve a 0.059 meter final pose error. We show the training curve in Figure 11. and the qualitative result on the demo website under the tab "New Character".

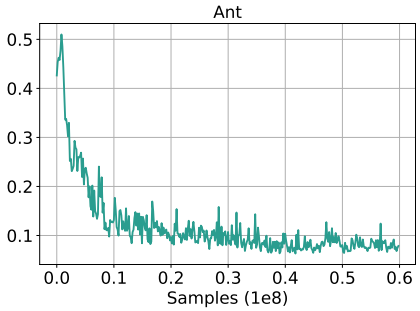


Figure 11: Pose error versus the number of samples for the new character, Ant.

G NEW SKILL

We run DiffMimic on a new skill, *360-degree Jump*, from the CMU Mocap Dataset (motion sequence CMU_075_09). The skill has been reported to be challenging for a RL-based method CoMic (Hasenclever et al., 2020) in a recent dataset paper (Wagener et al., 2022). DiffMimic is able to spin in the air and achieve a 0.020 meter final pose error. We show the training curve in Figure 12 and the qualitative result on the demo website under the tab "New Skill".

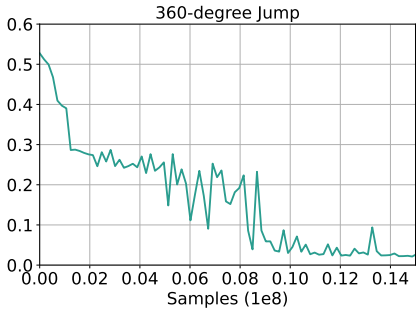


Figure 12: Pose error versus the number of samples for the new skill, *360-degree Jump*.

H NEW TRAINING STRATEGY

We additionally include the Reference State Initialization (RSI) technique (Peng et al., 2018a) into the DiffMimic training. RSI starts the training rollout from a random reference frame instead of the first reference frame. We observe a substantial empirical improvement after employing the new training strategy. The samples required are consistently around 90% less than DeepMimic (Peng et al., 2018a) as shown in Tab. 3. The final pose error has also been slightly improved as shown in Tab. 4. We show the qualitative results on the demo website under the tab "New Training". Notably, using RSI alone or RSI+Early Termination (ET) (Peng et al., 2018a) does not converge as shown in Figure 13.

Table 3: Number of samples required to roll out 20 seconds without falling in (10^6) with Reference State Initialization (RSI) (Peng et al., 2018a). Percentage: change in fraction of the DeepMimic samples.

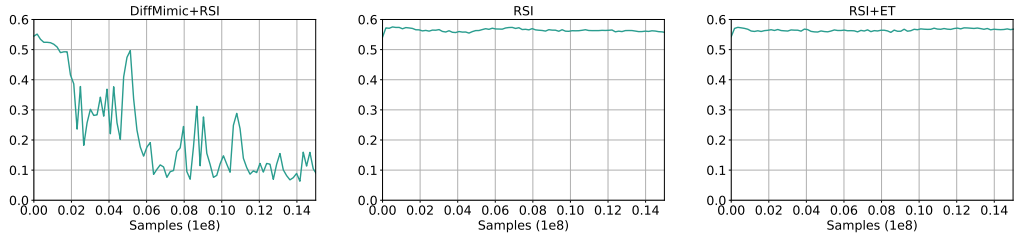
Motion	T_{cycle}(s)	DeepMimic	Spacetime Bound	Ours	Ours+RSI
Back-Flip	1.75	31.18	41.20 +32.1%	14.88 -52.2%	3.82 -87.7%
Cartwheel	2.72	30.45	17.35 -43.0%	13.92 -54.2%	4.72 -84.5%
Walk	1.25	23.80	4.08 -79.5%	7.92 -66.7%	1.55 -93.5%
Run	0.80	19.31	4.11 -78.7%	8.16 -57.7%	1.41 -92.7%
Jump	1.77	25.65	41.63 +77.8%	5.28 -79.4%	2.12 -91.7%
Dance	1.62	24.59	10.00 -59.3%	16.56 -32.6%	2.19 -91.1%

Table 4: Pose error comparison in meters. T_{cycle} is the length of the reference motion for a single cycle. The error is averaged on 32 rollout episodes with a maximum length of 20 seconds. The result of Ours+RSI is obtained without DTW.

Motion	T_{cycle}(s)	DeepMimic	AMP	Ours	Ours w/o DTW	Ours+RSI
Back-Flip	1.75	0.076 ± 0.021	0.150 ± 0.028	0.097 ± 0.001	0.105 ± 0.022	0.058 ± 0.015
Cartwheel	2.72	0.039 ± 0.011	0.067 ± 0.014	0.040 ± 0.007	0.040 ± 0.007	0.027 ± 0.002
Crawl	2.93	0.044 ± 0.001	0.049 ± 0.007	0.037 ± 0.001	0.037 ± 0.001	0.035 ± 0.003
Dance	1.62	0.038 ± 0.001	0.055 ± 0.015	0.070 ± 0.003	0.072 ± 0.014	0.042 ± 0.001
Jog	0.83	0.029 ± 0.001	0.056 ± 0.001	0.031 ± 0.002	0.031 ± 0.002	0.012 ± 0.000
Jump	1.77	0.033 ± 0.001	0.083 ± 0.022	0.025 ± 0.000	0.031 ± 0.002	0.015 ± 0.000
Roll	2.02	0.072 ± 0.018	0.088 ± 0.008	0.061 ± 0.007	0.083 ± 0.001	0.046 ± 0.005
Run	0.80	0.028 ± 0.002	0.075 ± 0.015	0.039 ± 0.000	0.046 ± 0.000	0.019 ± 0.000
Side-Flip	2.44	0.191 ± 0.043	0.124 ± 0.012	0.069 ± 0.001	0.121 ± 0.009	0.035 ± 0.001
Spin-Kick	1.28	0.042 ± 0.001	0.058 ± 0.012	0.056 ± 0.000	0.056 ± 0.000	0.036 ± 0.000
Walk	1.30	0.018 ± 0.005	0.030 ± 0.001	0.017 ± 0.000	0.017 ± 0.000	0.012 ± 0.000
Zombie	1.68	0.049 ± 0.013	0.058 ± 0.014	0.037 ± 0.002	0.037 ± 0.002	0.033 ± 0.009

I ANALYSIS ON ROBUSTNESS

We further evaluate DiffMimic's robustness to external forces. A force is applied to the pelvis of the character halfway through a motion cycle for 0.2 seconds, and we measure the maximum forwards and sideways push each policy can tolerate before falling. DiffMimic achieves comparable robustness to DeepMimic. We show the results in Table 5 and the qualitative results on the demo website under the tab "Robustness".



(a) (b) (c)

Figure 13: Pose error versus the number of samples on *Back-Flip* using (a) DiffMimic+RSI (b) standalone RSI (c) RSI+ET.

Table 5: Maximum forwards and sideways push each policy can tolerate before falling. Each push is applied halfway through a motion cycle to the character's pelvis for 0.2s. F: Forwards push. S: Sideways push. The unit is in Newton.

Motion	DeepMimic (F)	Ours (F)	DeepMimic (S)	Ours (S)
Back-Flip	440	**750**	100	**310**
Cartwheel	200	**350**	470	**690**
Run	**720**	500	**300**	270
Spin-Kick	**690**	630	600	**730**
Walk	240	**250**	**300**	220

J ANALYSIS ON SENSITIVITY

DiffMimic with inaccurate estimates of physical parameters. Briefly speaking, DiffMimic is able to learn the control policy well even though the friction coefficient deviates from the original parameters. We show the results in Table 6 and the qualitative results on the demo website under the tab "Sensitivity".

Table 6: Sensitivity to the friction coefficient μ, a system parameter. The evaluation is conducted on Zombie Walk, a motion heavily impacted by the friction coefficient. $\mu = 1.0$ is the standard setting, we additionally evaluate when $\mu = 0.8$ and $\mu = 1.2$.

Friction	Pose Error (m)
μ=0.8	0.047± 0.009
μ=1.0	0.032± 0.009
μ=1.2	0.028± 0.007

K ANALYSIS ON MOTION QUALITY

Motion quality can be severely affected by a small number of large-error poses, which can not be reflected by the existing average pose error metric. We propose a Pose Absurdity metric to quantify large-error poses. L2@0.01, 0.05, and 0.1 measure the average L2 pose error on the worst 1%, 5%, and 10% frames respectively. Full Horizon Gradient, Demonstration Replay (Random) ratio λ, and Demonstration Replay (Threshold) threshold γ are studied. Demonstration Replay (Threshold) outperforms other baselines by a large margin. We show the results in Table 7

L MORE DISCUSSIONS ON DEEPMIMIC REWARD

DeepMimic (Peng et al., 2018a) defines the reward r_t^I for each frame $time$ as:

$$r_t^I = w^p r_t^p + w^v r_t^v + w^e r_t^e + w^c r_t^c, \tag{L.1}$$

$$w^p = 0.65, w^v = 0.1, w^e = 0.15, w^c = 0.1, \tag{L.2}$$

where w are weighting hyper-parameters, $r_t^p, r_t^v, r_t^e, r_t^c$ are reward functions for joint orientation, joint velocity, end-effector position, and center-of-mass position respectively. Concretely, they are

Table 7: Ablation experiment on Full Horizon Gradient, Demonstration Replay (Random) ratio λ, and Demonstration Replay (Threshold) threshold γ with the new Pose Absurdity metric on *Back-Flip*. L2@0.01, 0.05 and 0.1 measure the average L2 pose error on the worst 1%, 5% and 10% frames respectively. Demonstration Replay (Threshold) significantly improves on the Pose Absurdity metric.

Method	L2@0.01	L2@0.05	L2@0.1
full horizon	0.357	0.343	0.323
$\lambda = 0.01$	0.379	0.341	0.318
$\lambda = 0.05$	0.195	0.177	0.135
$\lambda = 0.10$	0.359	0.332	0.262
$\gamma = 0.1$	0.111	0.107	0.095
$\gamma = 0.2$	0.113	0.107	**0.093**
$\gamma = 0.4$	**0.110**	**0.103**	0.095

defined as follows:

$$r_t^p = \exp\left[-2\left(\sum_j ||\hat{q}_t^j \ominus q_t^j||^2\right)\right], \tag{L.3}$$

$$r_t^v = \exp\left[-0.1\left(\sum_j ||\hat{\dot{q}}_t^j - \dot{q}_t^j||^2\right)\right], \tag{L.4}$$

$$r_t^e = \exp\left[-40\left(\sum_e ||\hat{p}_t^e - p_t^e||^2\right)\right], \tag{L.5}$$

$$r_t^c = \exp\left[-10\left(||\hat{p}_t^c - p_t^c||^2\right)\right], \tag{L.6}$$

$$\tag{L.7}$$

where $-2, -0.1, -40, -10$ are four more weighting hyper-parameters.

Note that there are 8 hyper-parameters in the reward function in total. Moreover, a selection on joints (*e.g.*, only apply position reward to end-effectors and center-of-mass) is also required.

In comparison, DiffMimic uses a direct L2 distance between states without weighting as defined in Eq. 3.2, which is much simpler.

M More Discussions on Related Works

SHAC (Xu et al., 2022) and PODS (Mora et al., 2021) are two differentiable physics-based reinforcement learning algorithms. SHAC addresses the gradient exploding and diminishing problem by manually truncating trajectories into small segments. It requires careful design of gradient truncation throughout the learning process. In contrast, DiffMimic automatically resets the state to the demonstrated state, preventing the state from deviating too much and achieving a similar effect to the trajectory truncation in SHAC. In addition, SHAC seeks to maximize the reward, which can often lead to a local optimum when the reward function is ill-defined or difficult to design. DiffMimic, on the other hand, benefits from the expert demonstration, which provides explicit guidance to escape from local optimal.

Using second-order gradients, PODS claims a monotonic policy improvement and a significant speedup in convergence over first-order methods. Despite its success, PODS makes strong assumptions about the second-order derivatives of a differentiable simulator and can be sensitive to the accuracy of the second-order derivatives. Moreover, second-order derivatives incur more computational demands, which can be potentially costly in tasks with complex dynamics. Compared to SHAC and DiffMimic, PODS can suffer from gradient exploding and diminishing problems without

a proper truncation scheme. This problem can even be exaggerated with the use of second-order gradients in longer trajectory rollouts.

Supertrack (Fussell et al., 2021) adopts the idea of model-based RL and proposes to learn a world model to predict the next simulated character state given the previous simulated character state and the provided PD targets. It learns the world model and the control policy in an iterative way: 1) the policy rollouts in a physics simulator, 2) the trajectories are collected to train the world model, 3) the policy is then trained through the differentiable world model. However, one major disadvantage of the approach is that its policy horizon, i.e., rollout steps, is small, e.g., 32 steps, during training due to the error accumulation in the world model. Training can get unstable for long rollouts, e.g., 64 steps. Therefore, as pointed out by the authors, SuperTrack can be unsuitable when the policy needs a long-term apprehension to mimic some challenging motions. DiffMimic uses DPS in replace of the learned world model to avoid error accumulation.

ScaDiver (Won et al., 2020) presents a scalable approach to learning controllers for a large motion database, where motion mimicking is a key part of the pipeline. It proposes to 1) divide the diverse motion database into clusters of similar motions, 2) use DeepMimic to learn a control policy for each motion cluster, and 3) learn a mixture-of-expert controllers to dynamically ensemble the control policies. We view ScaDiver as an important downstream application of DiffMimic. DiffMimic can serve as a better alternative to DeepMimic in step 2, and contribute to a higher efficiency when learning from large-scale motion databases.