Leveraging motion imitation in RL for biped character

Zhichao Sun
ETH Zurich
zhisun@ethz.ch

Junzhe He
ETH Zurich
junzhe@ethz.ch

Xuejing Luo
ETH Zurich
xueluo@ethz.ch

Chen Gao
ETH Zurich
chegao@ethz.ch

Abstract

Generating natural and versatile movements for simulated characters is crucial for enhancing immersion and user experience in interactive applications such as video games, animation films, and virtual reality. Recent research has integrated imitation learning into RL methods, exemplified by DeepMimic, to address challenges in pure RL and example-based methods. However, limitations remain, motivating our work to refine the DeepMimic method. Our work introduces adaptive modifications and enhancements to improve DeepMimic's performance. These include retargeting the motion dataset to the character, training multiple specialized policies, developing a multi-skill policy, and enabling the policy to seamlessly switch between motion clips for various tasks. Extensive evaluations demonstrate the effectiveness of our framework in terms of both task performance and motion quality, thereby enhancing immersion and user experience in interactive applications. Source code is available at https://github.com/zcsun52/Leveraging-motion-imitation-in-RL-for-biped-character.git.

***Index terms* —** Reinforcement learning, Motion imitation, Motion retargeting, Multi-skill policy, Biped character animation.

1. Introduction

Physics-based character locomotion, the task of generating natural and versatile movements for simulated characters, is a cornerstone of several interactive applications, including video games, animation films, and virtual reality. Achieving high-fidelity movements and interactions within a dynamic environment enhances immersion and user experience, pushing the boundaries of realism within the digital world.

Several approaches to creating such character locomotion have emerged over time. These primarily include example-based [4, 11], model-based [7, 12], and pure reinforcement learning (RL) [3, 9] methods. Each approach comes with its inherent strengths and limitations.

Example-based methods can produce highly realistic animations given sufficient training data, but they often struggle to adapt to changes in the environment or task objectives. Model-based approaches offer more adaptability and can handle different environments effectively, but they often require considerable manual tuning and may still fail to reproduce the full range of natural human motion.

Pure RL methods, on the other hand, show promise in their adaptability, enabling the learning agent to adjust to new circumstances and learn from mistakes [8]. However, they often lead to less natural movements due to the challenge of encapsulating the complexity of human motion in the reward function [14].

Recognizing these challenges, recent research [2, 6, 13, 16] has focused on integrating imitation learning into reinforcement learning methods, an approach exemplified by the pioneering work of DeepMimic by Peng et al. [14]. This approach helps circumvent some of the difficulties associated with pure RL and example-based methods.

The integration of RL with imitation learning allows expert demonstrations to guide the learning process. As such, the learning agent can draw upon these demonstrations to achieve the desired behavior, reducing the need for meticulous reward function design. This guidance can significantly enhance the stability of the learning process, accelerating the speed of convergence [14].

Moreover, by utilizing a policy optimization approach, DeepMimic effectively reproduces a diverse range of complex human movements. It demonstrates the capacity to handle high-dimensional, continuous control tasks, leading to high-fidelity motion generation [14]. This approach has opened up new possibilities for creating realistic, dynamic, and fluid character locomotion in a simulated environment.

Nevertheless, despite these substantial advancements, certain limitations and challenges persist, which motivates our work. Notably, DeepMimic was restricted to a simplified humanoid character model. To enhance the realism of the simulation, we adopted a character model named Bob, which more closely resembles human physiology. In response to the remaining challenges, we present adaptive modifications and improvements to the DeepMimic

method. Specifically, we divide the problem into four tasks:

1. **Retargeting Motion Dataset to the Character:** This task should ensure correct retargeting of complex tasks that involve interactions with environments, such as preventing issues like body penetration.

2. **Training Several Specialized Policies:** For this task, we trained several specialized policies, each imitating a specific motion clip while being capable of accomplishing a user-specified task input, such as striking a target and reaching a target position.

3. **Training a Multi-Skill Policy:** For this task, we trained a single policy that learns all motions jointly. With a selection of one-hot vector input, this policy can imitate the motion clip corresponding to the nonzero entry and smoothly transit between motion clips upon change of input.

4. **Training a Multi-Skill Policy for Task-Switching:** Utilizing adversarial imitation learning, we trained a policy that provides locomotion strategies by imitating motion styles from a policy dataset to accomplish a user-specified task. This approach automatically selects which motion to perform generalizing from the policy dataset.

By addressing these tasks, we aim to refine the methods proposed by DeepMimic, striving towards a more robust, adaptable, and efficient system for physics-based character locomotion.

2. Related Work

In the field of animation and reinforcement learning, various projects and studies have laid the foundation for our research. The goal of this project is to further develop these concepts and approaches, and apply them to tasks of retargeting motion dataset to a character, training policies that imitate individual motion clips, training a single policy for all motion clips based on selection input, and training a policy that can switch between motion clips for other tasks.

Physics-Based Methods: Physics-based methods for animation serve as a foundation for our project, a key aspect of which is represented by the DeepMimic approach proposed by Peng et al. [14]. This groundbreaking work introduced a novel methodology to reproduce complex, human-like motion data in physics-based simulations. Their system employs an actor-critic reinforcement learning method that combines sample efficiency from off-policy learning with the robustness of on-policy learning, a combination crucial in achieving highly responsive physics-based character control. DeepMimic has exhibited a level of fluidity and responsiveness in simulated characters that were previously unseen. Our project extends this method to various other tasks, integrating it with other established approaches to expand the possibilities of robust and adaptive motion control policies.

Reinforcement Learning: RL is a vital part of animation techniques, specifically for training motion policies. DeepMimic utilizes a type of RL, the Proximal Policy Optimization (PPO), in its learning strategy. This approach, originally proposed by Schulman et al. [19], aims to provide an alternative to the traditional policy gradient methods, which often struggle with issues of data efficiency and stability. PPO maintains the benefits of policy optimization while adding stability and robustness, allowing for more sophisticated training mechanisms.

An essential contribution to RL in the domain of animation and motion control is the work by Rajeswaran et al. [18]. Their research presents a method of multi-task reinforcement learning where a single policy can generalize to multiple tasks. This work shows how a policy trained on various tasks simultaneously can adapt to a range of tasks better than a policy trained on a single task. In our project, this approach inspires the development of a single policy for all motion clips, dictated by a selection input. This task requires the policy to understand and replicate various motion clips, pushing the boundaries of RL in animation.

Imitation Learning: Imitation Learning techniques, which involve learning from demonstrations, also significantly impact our project. For the training of policies that imitate individual and multiple motion clips, the work of Argall et al. [1] stands out. They presented an approach to robot learning through demonstration, allowing the development of policies for each motion task. This methodology led to efficient learning and is seen as one of the founding works in learning from demonstration.

Another critical piece in this area is the Hierarchical Deep Reinforcement Learning approach by Tessler et al. [21]. This method allows the model to select and switch tasks efficiently, offering much-needed flexibility. Tessler et al. also demonstrate how this approach can incorporate prior knowledge of related tasks to improve learning efficiency. This aspect is particularly relevant for our project, where training a policy to switch between motion clips based on other tasks requires both flexibility and adaptability.

Peng et al [17] introduced a generative adversarial learning scheme for physics-based character animation that enables the imitation of a wide range of behaviors from large and diverse datasets by introducing a style reward with a learned discriminator. This reward indicates the similarity of the motions generated by the policy to the motions sampled from the dataset, with which users can specify high-level task input to control where to go, while the learned discriminator can control what to do, namely the low-level style of a character's motions. However, the discriminator

in this approach is trained on state transitions $D(s, s')$ instead of the original state-action pairs $D(s, a)$. State transitions provide a less direct and informative feedback signal compared to state-action pairs. The discriminator relies on the consistency between generated trajectories and expert trajectories, but it may be less effective in providing precise feedback on specific actions. This limitation can hinder the learning process and result in slower convergence or suboptimal imitation performance. To address this issue, our approach in Task 4 utilizes the specialized policies we trained in Task 2 to train the discriminator directly from a policy dataset instead of a motion clip dataset.

Latent Space Models: The section relates to Latent Space Models, a powerful tool for handling high-dimensional data. The motion retargeting work of Holden et al. [10] plays a pivotal role here. They proposed a deep learning framework using Autoencoders that reduces the complexity of motion data into a low-dimensional representation. This framework can transform and retarget motion in a reduced space, making it a crucial component for our project's task of retargeting motion datasets to a character.

Moreover, the concept of a latent space model for motion prediction was introduced by Fragkiadaki et al. [5]. They proposed an encoder-decoder structure that learned to predict future frames in a video given the past frames, extending

3. Method

In this section, we present the methods used for each task in our work.

3.1. Retargeting Motion Dataset to the Character

Our first task involves supplying the learning-based method with a high-quality motion capture dataset. In the realm of 3D character animation, BioVision Hierarchy (BVH) files have proven instrumental for representing motion capture (mocap) data. These files comprise two essential parts: a joint hierarchy delineating the character's skeletal structure and motion data detailing the temporal progression of joint movements. Thus, BVH files furnish an invaluable source of expert demonstrations to guide the learning process.

Nevertheless, there is a critical distinction between the raw motion data contained in the BVH files and the specific data format that DeepMimic can understand. As such, we must retarget or convert the BVH files to align with the character model utilized in DeepMimic (in our case, Bob) and its specific motion data representation as shown in Figure 1. The key challenge in this step lies in retaining the naturalness and realism of the retargeted movements, despite potential disparities between the human actor's and the digital character's dimensions and proportions.

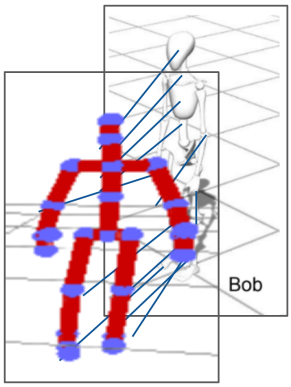


Figure 1. Visualization of retargeting the BVH files to align with the character model Bob.

The retargeting process from BVH files to DeepMimic-compatible input data encompasses several crucial steps:

1. **Parsing BVH Files:** Initially, the Python BVH library is employed to read and parse the BVH file, thereby extracting motion data and hierarchical information.

2. **Mapping Joints:** Subsequently, the joint names from the BVH file correspond to those used in the DeepMimic character model. This mapping disregards any joints present in the BVH file but absent in the DeepMimic model and estimates missing joints.

3. **Converting Coordinate System:** The next step involves translating the motion data from the BVH file's coordinate system into the one used by DeepMimic.

4. **Normalizing the Data:** Then, the joint angles are normalized to match the range expected by the DeepMimic model, which ensures appropriate motion scaling.

5. **Outputting Converted Data:** Ultimately, the normalized and converted motion data is outputted in a format compatible with the DeepMimic model.

Through this retargeting process, we enable the effective training of our physics-based character model within the DeepMimic environment. This approach guarantees that our simulated character's movements closely align with the expert demonstrations embodied in the BVH files.

3.2. Train several specialized policies each of which imitates a motion clip

This task focuses on exploring the efficacy of RL in imitating the motion of biped characters. To achieve this, we leveraged the open-source codebase of DeepMimic [15],

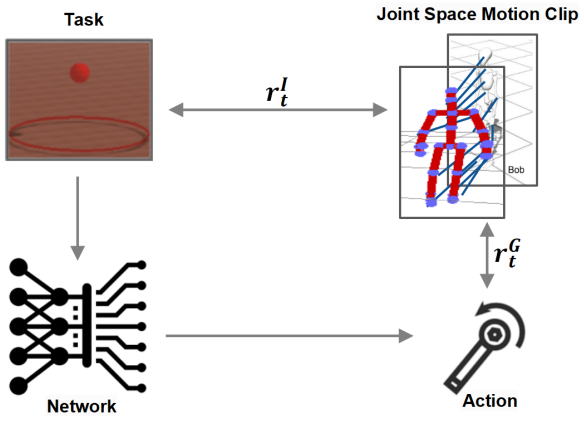


Figure 2. Overview of the integration of task-specific rewards and action rewards into the reinforcement learning network, enabling motion imitation and task execution.

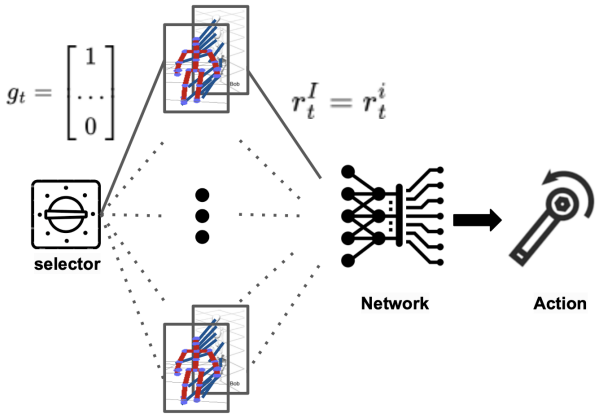


Figure 3. Overview of the pipeline of the Skill Selector, which is able to execute arbitrary sequences of diverse skills on demand.

which presents a novel method for training agents to perform complex tasks with human-like motion as shown in Figure 2.

DeepMimic uses PPO [20], a policy optimization method that is a part of the RL framework. This method relies on stochastic gradient ascent to optimize policy by using a surrogate objective function. DeepMimic introduces a reward function that depends on the difference between the reference and character poses, also taking factors like the impact of falls and the smoothness of the motion into account. Since a reference motion is a sequence of target poses $\{\hat{q}_t\}$, which only provides kinematic information in the form of target poses, the policy is responsible for determining which actions should be applied at each timestep in order to realize the desired trajectory. In addition to simple motion imitation, we also trained the RL models to perform specific tasks. The reward r_t at each step t consists of two terms that encourage the character to match the reference motion while also satisfying additional task objectives:

$$r_t = w^I r_t^I + w^G r_t^G$$

Here, r_t^I and r_t^G represent the imitation and task objectives, with ω^I and ω^G being their respective weights. This allows the model to not only mimic a specific motion, but also use that motion to achieve a certain task goal. r_t^I is further decomposed into terms as follows:

$$r_t^I = w^p r_t^p + w^v r_t^v + w^e r_t^e + w^c r_t^c$$

Here, r_t^p represents the pose reward, r_t^v represents the velocity reward, r_t^e represents the end-effector reward and r_t^c represents the center-of-mass reward. The velocity reward r_t^v encourages the character to match the joint orientations of the reference motion at each step and is computed from the difference of local joint velocities. The velocity reward r_t^v is computed from the difference of local joint velocities. The end-effector reward r_t^e encourages the character's hands and feet to match the positions from the reference motion. Finally, r_t^c penalizes deviations in the character's center-of-mass from that of the reference motion.

3.3. Train a single multi-skill policy that imitates all motion clips based on selection input

In the previous subsection, our discussion focused on the imitation of individual motion clips. However, in order to enable the model to perform more complex tasks, it is necessary to compose and sequence multiple clips. In this subsection, we examine a specific scenario where a single policy has the capability to imitate multiple diverse motion clips. Once trained, the policy can imitate an arbitrary sequence of clips based on user selection. This task is also referred to as the Skill Selector.

To represent the desired imitation goals for the policy, we introduce a one-hot vector:

$$g_t \in \{0,1\}^n : \sum_{i=1}^{n} g_{t,i} = 1$$

where n denotes the total number of motion clips that the policy should be able to imitate. Each entry $g_{t,i}$ in the vector represents the corresponding motion that should be executed at a specific time. The objective of the character is to imitate the non-zero entry in g_t. During training, the character is trained to optimize the imitation objective, denoted as r_t^I, which corresponds to the imitation object of the currently selected motion r_t^i (where $g_{t,i} = 1$). No additional task objective r_t^G is needed. The overall pipeline is illustrated in Figure 3.

To ensure that the policy learns smooth transitions between all the skills within the set of clips, the following crucial steps are followed during training:

1. **Random Goal Assignment:** A goal input indicating which clip the policy should imitate is randomly provided. The reference mocap data is switched to the selected clip, and the reward is calculated accordingly.

2. **Clip Duration:** Each clip is held for a duration of 5-10 seconds before sampling a new goal and transitioning to the new clip.

3. **Root Position and Heading Synchronization:** Whenever the clip changes, the root position and heading of the reference mocap data are synchronized with those of the simulated character.

Following these steps, the policy can effectively learn the transition between different skills within the set of motion clips and ensure smooth execution of the desired imitation tasks.

3.4. Train a multi-skill policy that can switch between motion clips based on user-specified tasks

While shown to be able to learn different motion clips and transitions from task 3, training a single policy has several drawbacks including limited expressiveness, a significant increase in complexity and training Time, and a lack of modularity. Different motion clips often involve a wide range of motion styles, each having intricate details and nuances with different joint angle distributions. Training a single policy under this setup can be more complex and time-consuming compared to training multiple specialized policies. Besides, using a single policy makes it difficult to modify or fine-tune specific behaviors independently. It can be challenging to make targeted adjustments or improvements to individual motion patterns without affecting the entire policy or other imitated motions. To address these issues, Peng et al [14] proposed to use a composite policy to select specialized policies using a Boltzmann distribution. But it has been shown that this approach does not generalize well to large and diverse motion datasets without careful selection of appropriate motion for the character to track given a specific task.

Another multi-skill training method was brought up by Peng et al [17] using generative adversarial learning. By training a discriminator, also denoted as an Adversarial Motion Prior (AMP), this approach can learn from a large and diverse dataset while fulfilling a high-level task. However, as discussed in Sec. 2, state transitions can hinder the learning process and result in local optima due to a lack of expressiveness in the joint space. Our method adapted the Adversarial Motion Prior to an Adversarial Policy Prior (APP), where the discriminator takes the state-action pairs sampled from pre-trained specialized policies as input instead of the state transitions, as shown in Figure 4. We adopted the same

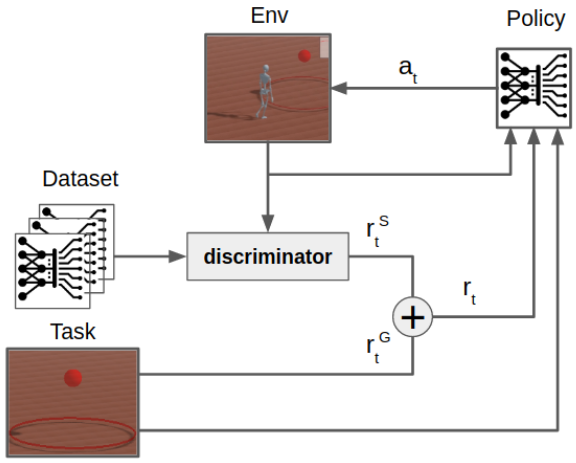


Figure 4. Overview of the system structure. Given a policy dataset containing the desired motion policies for the character, the system trains a policy prior that specifies style-rewards r_t^S for the policy during training. Then the task-rewards r_t^G are combined with the style rewards to train a policy that enables a simulated character to satisfy the user-specified task, while also generating behaviors that imitate the reference motion styles in the dataset

objective function to train the discriminator as proposed in AMP, except that the input to the discriminator is changed to a state-action pair:

$$\arg\max_D \mathbb{E}_{d^{\mathcal{M}}(s,a)}\left[(D(s,a)-1)^2\right]+\mathbb{E}_{d^{\pi}(s,a)}\left[(D(s,a)+1)^2\right] \quad (1)$$

The reward function that trains the policy to discriminate samples from the dataset $\mathcal{M}$ (score of 1) and from the policy (score of -1) is given by:

$$r(s,a) = \max[0, 1 - 0.25(D(s,a) - 1)^2] \quad (2)$$

4. Experiments

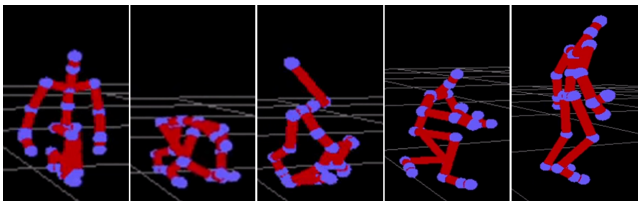
In this section, we provide details of the experiments conducted to evaluate the effectiveness of our framework for each task.

4.1. Retargeting Motion Dataset to the Character

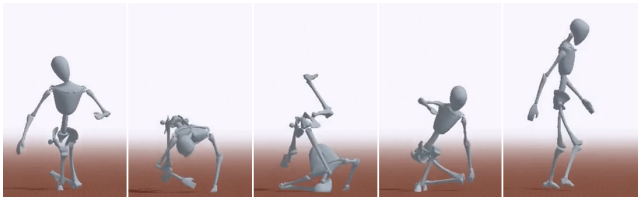
Our first task involves supplying the learning-based method with a high-quality motion capture dataset represented in BVH files obtained from the SFU Motion Capture Database[1].

To visualize the retargeting process, we present Figure 5, which showcases the retargeting of BVH files to align with the character model, Bob. Figure 5a displays the original motion clip, demonstrating a parkour roll on the floor

[1]The data used in this project was obtained from http://mocap.cs.sfu.ca. The database was created with funding from NUS AcRF R-252-000-429-133 and SFU President's Research Start-up Grant.



(a) Original motion clip.



(b) Retargeted motion clip.

Figure 5. Visual comparison of the original BVH motion clip depicting a parkour roll on the floor starting from a kneeling position, and the retargeted motion clip applied to Bob.

starting from a kneeling position. Figure 5b displays the retargeted motion clip using the method described in Sec. 3.1.

The retargeting process ensures that the motions from the BVH files are properly adapted to Bob, while preserving the naturalness and realism of the movements. By comparing the original motion clip with the retargeted version, we can observe the successful alignment of the character's movements with the intended motion demonstrated in the BVH file.

4.2. Train several specialized policies each of which imitates a motion clip

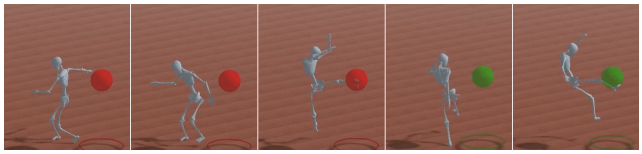
For this task, We trained the RL models described in Sec. 3.2 with our retargeted motion clips: walk, backflip, sideflip and spinkick as shown in Figure 6a. We also trained motion clip spinkick with a target-hitting task as shown in Figure 6b, and both results look promising. This model also acts as a baseline model for training a policy that can switch between motion clips based on other tasks.

4.3. Train a single multi-skill policy that imitates all motion clips based on selection input

By utilizing the one-hot vector representation, we conducted training to enable a policy to perform different styles of walking, specifically resembling a zombie, walking in a normal manner, and adopting a stealthy gait. During the testing phase, we introduced a new random goal to the policy approximately every 3 seconds to simulate random user input. As depicted in Figure 7a, the single trained policy successfully imitates all three types of walking and exhibits smooth transitions between the skills.

Furthermore, we made an effort to train policies capable of transitioning between more intricate skills, such as dancing, flipping, and jumping. However, the outcomes were not satisfactory within a limited training time.

(a) Top-to-bottom: backflip, sideflip and spinkick.



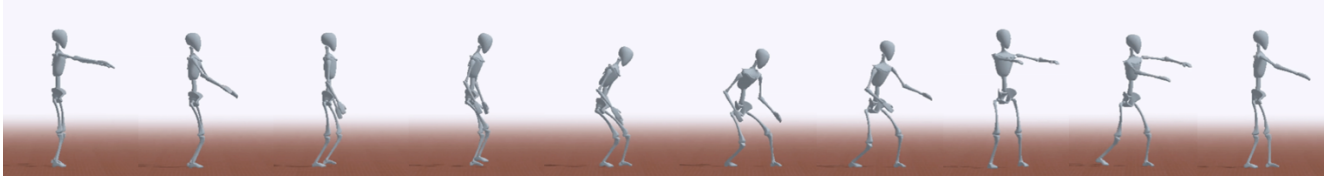
(b) Spinkick policy trained to strike a target with the character's right foot.

Figure 6. Snapshots of motions from the trained policies of task 2.

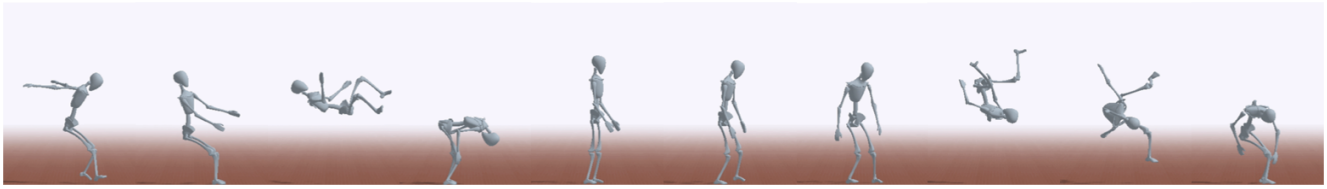
As presented in Figure 7b, we specifically focused on training a policy to perform side-flipping, back-flipping, and jumping. After two days of training, the policy demonstrated consistent and successful execution of back-flipping. However, it encountered difficulties in effectively performing side-flipping and jumping, particularly in maintaining a stationary position after flipping or not performing the selected skill at all. This outcome was expected since the inclusion of numerous diverse motions often hinders the policy from effectively imitating any of the motions. Hence, it is evident that a more advanced approach, as discussed in Sec. 3.4, is necessary to composite a larger and more diverse set of skills.

4.4. Train a multi-skill policy that can switch between motion clips based on user-specified tasks

Using the state-action pairs sampled from the policies trained in task 2 including walking, punching, turning, etc, we trained a multi-skill policy in an adversarial generative manner which successfully accomplished user-defined tasks including a navigation goal (Figure 8a) and a strike target (Figure 8b) with proper locomotion strategies. To benchmark with AMP, we conducted 20 experiments for each AMP policy and APP policy with the same set of motion clips and target/goal distribution. Our method demonstrates a faster convergence and higher success rate in reaching

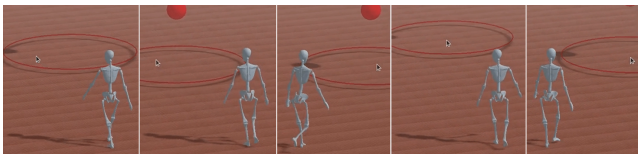


(a) Single policy trained for different styles of walking, including resembling a zombie, walking in a normal manner, and adopting a stealthy gait.

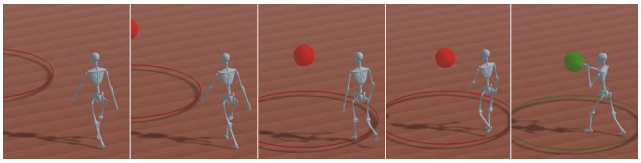


(b) Single policy trained for performing back-flipping, jumping, and side-flipping.

Figure 7. Snapshots of motions from the trained policies of task 3.



(a) Walking policy trained to walk towards a target.



(b) Punching policy trained to walk and strike a target with the character's right hand.

Figure 8. Snapshots of motions from the trained policies of task 4.

Method	AMP	Ours
iter # at convergence	$\approx$ 22500	$\approx$ 20000
success rate far goal($> 25m$)	0.85	0.75
success rate close goal ($< 15m$)	0.9	1.0

Table 1. Benchmarking with AMP.

goals or striking targets distributed within a 15m radius around the robot. For target/goal farther away ($> 25m$, unseen in training), however, APP's performance is not as good as AMP, as shown in Table 1. This indicates that utilizing state-action pairs is indeed more expressive in the joint space, leading to improved sample efficiency, yet at the expense of generalizability to tasks not encountered during training, resulting in a lower success rate for distant targets.

5. Conclusion

In this research, we have presented adaptive modifications and improvements to the DeepMimic method, building upon the work in DeepMimic. Our contributions include retargeting the motion dataset to the new character model, Bob, training specialized policies, developing a multi-skill policy, and enabling task-switching between motion clips. Through evaluation, we have demonstrated the effectiveness of our method in generating natural and versatile movements for biped characters.

However, our research also highlights several challenges and limitations. For example, complex tasks involving interactions with environments may not be adequately retargeted, and the policies may struggle with generalization to new motion clips and exhibit overfitting to specific training clips. Training a single policy that can imitate all motion clips and produce diverse motions remains a challenging task. Furthermore, the robustness of the model requires further investigation, particularly regarding failures induced by significant changes in inputs. Adversarial Policy Prior solves the diversity of dataset but still requires tuning over the transitions between clips, namely we have to manually train some policies for the transition phases.

Future work could focus on addressing these challenges and limitations, such as developing more robust and adaptive methods for motion retargeting, exploring techniques to improve the generalization of policies to new motion clips, and investigating the use of more diverse and complex datasets for training. By tackling these areas, we can advance the field of physics-based character locomotion and enhance the realism and versatility of character movements in interactive applications.

6. Contributions of team members

In this project, each task is assigned to one group member. Task 1 is assigned to Xuejing Luo, task 2 is assigned to Zhichao Sun, task 3 is assigned to Chen Gao, and task 4 is assigned to Junzhe He.

References

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. 2

[2] Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pages 1–10, 2018. 1

[3] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016. 1

[4] Andrew W Feng, Yuyu Xu, and Ari Shapiro. An example-based motion synthesis technique for locomotion and object manipulation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 95–102, 2012. 1

[5] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics, 2015. 3

[6] 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. 1

[7] Félix G Harvey and Christopher Pal. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*, pages 1–4. 2018. 1

[8] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017. 1

[9] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016. 1

[10] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning Motion Manifolds with Convolutional Autoencoders. In *Proceeding SA '15 SIGGRAPH Asia 2015 Technical Briefs*, page 18. ACM, 2015. 3

[11] Rune Skovbo Johansen. Automated semi-procedural animation for character locomotion. *Aarhus Universitet, Institut for Informations Medievidenskab*, 2009. 1

[12] Ian Mason, Sebastian Starke, and Taku Komura. Real-time style modelling of human locomotion via feature-wise transformations and local motion phases. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5(1):1–18, 2022. 1

[13] Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017. 1

[14] 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, 2018. 1, 2, 5

[15] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. https://github.com/xbpeng/DeepMimic, 2018. 3

[16] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020. 1

[17] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP. *ACM Transactions on Graphics*, 40(4):1–20, jul 2021. 2, 5

[18] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations, 2018. 2

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 2

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. 4

[21] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel Mankowitz, and Shie Mannor. A Deep Hierarchical Approach to Lifelong Learning in Minecraft. *Proceedings of the AAAI Conference on Artificial Intelligence*, Feb. 2017. 2