Leveraging Motion Imitation in Reinforcement Learning for Biped Character

Xiyi Chen*
ETH Zurich
xiychen@ethz.ch

Anthony Eid*
ETH Zurich
anteid@ethz.ch

Kexin Shi*
ETH Zurich
kexshi@ethz.ch

Jin Jin*
ETH Zurich
jinjin@ethz.ch

Fatemeh Zargarbashi
ETH Zurich
fatemeh.zargarbashi@inf.ethz.ch

Stelian Coros
ETH Zurich
scoros@inf.ethz.ch

Siyu Tang
ETH Zurich
siyu.tang@inf.ethz.ch

Abstract

This paper delves into the significant realm of teaching highly dynamic skills to robots and virtual characters through imitation of motion capture clips, a problem that bridges the divide between human expertise and robotic capabilities. It first scrutinizes the strengths and weaknesses of current approaches, drawing attention to their struggles with sophisticated, agile movements and the flexibility to adapt to diverse scenarios. Based on recent advances in deep learning, reinforcement learning (RL), and imitation learning, we build upon DeepMimic to combine these techniques to optimize control policies and facilitate more realistic, diverse, and adaptable dynamic skills. We demonstrate our approach using the DeepMimic motion clip data, successfully deploying it on the bob biped robot for executing various movements, such as walking, running, and jumping. Additionally, a curriculum training strategy is proposed to extend our algorithm's applicability to various biped robots with different shapes, masses, and dynamics models, thereby driving the innovation in robotics and virtual reality applications. Our code and demo are publicly available at https://github.com/xiyichen/dh-project.

1. Introduction

Learning highly dynamic skills by imitating reference motion capture clips is a compelling research area within robotics and animation. The ability to teach robots and virtual characters to perform complex and agile movements has significant implications across various domains, including entertainment, robotics, virtual reality, and human-robot interaction. By leveraging the rich information captured in motion capture data, these methods aim to bridge the gap between human expertise in dynamic motions and the capabilities of robotic systems. In this paper, we will explore the importance of this problem, the limitations of existing methods, and our proposed method.

The problem of teaching highly dynamic skills through motion imitation is crucial because it enables robots and virtual characters to perform tasks that involve agility, acrobatics, and precise control in dynamic environments. Examples include performing aerial maneuvers, acrobatic flips, parkour-style movements, and other physically demanding actions. Such skills find applications in entertainment industries for creating lifelike and engaging characters, in robotics for enabling agile robotic systems capable of navigating complex environments, and in virtual reality for immersing users in dynamic and realistic virtual experiences.

Existing methods have made significant progress in motion imitation, but they often fall short when it comes to learning highly dynamic skills. Traditional methods based on kinematic controllers and handcrafted rules struggle to capture the intricacies and nuances of agile movements. They rely on manual engineering and lack the flexibility to adapt to diverse scenarios and achieve natural and realistic dynamic motions.

To address these limitations, recent research has proposed novel approaches that combine deep learning, reinforcement learning, imitation learning and optimization techniques to learn highly dynamic skills by imitating reference motion capture clips. These methods aim to leverage the power of deep neural networks to learn complex representations of motion data and optimize control policies that reproduce the desired dynamic behaviors. The combination of data-driven learning, physics-based simulation, and optimization methods can now capture the subtle details and nuances of agile motions, enabling robots and virtual characters to perform dynamic skills with a higher level of realism, diversity, and adaptability expanding the possibilities for robotics, entertainment, and virtual reality applications.

One notable paper in this area is DeepMimic [22] by

*These authors share the first authorship.

Peng *et al.* The authors introduce a deep reinforcement learning framework that combines motion capture data with physics-based simulation to teach virtual characters highly dynamic skills. Their approach enables characters to learn a wide range of dynamic movements, such as acrobatic flips, aerial dives, and athletic jumps, surpassing the capabilities of traditional kinematic controllers.

Another significant contribution is OPT-Mimic [8] by Jiang *et al.* The paper presents a framework that integrates trajectory optimization and motion imitation techniques to enable robots to learn highly dynamic and agile motor skills. By iteratively refining the reference motion capture trajectories through optimal control and incorporating them into the learning process, the approach produces robots capable of performing complex dynamic tasks with enhanced stability and efficiency.

Furthermore, recent advancements in learning adversarial motion priors (AMP) [23] by Peng *et al.* have shown promise in improving the quality and diversity of imitated motions. The authors introduced an adversarial training framework that combines deep RL with generative adversarial networks (GANs). The adversarial discriminator network is trained to distinguish between imitated motions and real motion capture data. By leveraging adversarial training, the imitating system can generate motions that exhibit enhanced realism, fluidity, and diversity. This capability opens avenues for creating lifelike virtual characters, realistic simulations, and agile robotic systems.

In this paper, we propose a reinforcement learning-based method to imitate agile behaviors from motion clips data provided in Deepmimic [22], and deploy it on the bob biped robot with successfully performing walk, run, and jump motion. We also attempt to train a policy that enables bob to walk in different heading directions with different velocities as well as single policy for multiple motions via skill-selector. We propose finally a curriculum training strategy to generalize our algorithm on different biped robots with different shape, mass and dynamics models.

The rest of this paper is structured in the following order: Section 2 introduces the development in motion learning, Section 3 details the components of the proposed approach, Section 4 demonstrates the performance and the ablation study, Section 5 summarizes this work and throw out thoughts for the future, and Section 6 details the contribution among team members.

2. Related Work

2.1. Kinematic and Physics-based Models

Kinematic and physics-based models have a longstanding history in representing, analyzing, and recreating human motion, serving as crucial cornerstones in computer animation and robotics. Initial work in kinematic modeling used joint angles over time to recreate movement [20]. While these deterministic models provided control, the resulting movements often appeared stiff and robotic, lacking the fluidity of natural human motion.

Recognizing this shortfall, researchers began to incorporate physics-based models that considered forces, torques, and biomechanical constraints into their work [25]. In computer graphics, pioneers like Popovic *et al.* [26] initiated physically-based animation of human figures, leveraging control methods to generate life-like behaviors.

Despite added realism, physics-based models were inherently complex due to numerous factors, including the need for maintaining balance and coordination. As a solution, optimization-based methods were introduced [18]. This approach, as demonstrated by Liu *et al.* [19], optimized over multiple frames to generate stable and smooth movements, thus marking significant improvements. However, these methods often required substantial computational resources, limiting their applicability in real-time scenarios.

To overcome these limitations, researchers pivoted towards data-driven approaches that used motion capture (MoCap) data as a reference for generating animations [3, 27]. While these methods improved the generation of complex motions, challenges remained when applying highly dynamic skills due to difficulties in temporal and spatial alignment.

2.2. Motion Imitation

Motion imitation, particularly of MoCap data, is a research area that has received significant attention. Early works like that of Kovar *et al.* [16] used motion graphs to blend pre-recorded clips into novel sequences. However, these methods often failed to produce plausible transitions in a continuous, unsegmented stream of motion.

Later, Holden *et al.* [14] addressed these issues by learning a neural network policy directly from MoCap data, which allowed characters to reproduce complex human locomotion. Liu *et al.* [2] extended this approach to a broader set of skills through the use of hierarchical policies. However, challenges still persist in capturing highly dynamic skills due to complexities in temporal and spatial alignment, as well as the nuances in motion dynamics. Aytar *et al.* [4] demonstrated the potential of RL in learning from visual demonstrations. However, the applicability to highly dynamic skills remains a challenge.

Recent work by Peng *et al.* [22] proposed DeepMimic, an approach that used RL to train policies to imitate complex skills from MoCap data. This approach demonstrated the robust control of simulated characters and the ability to learn a variety of skills from reference MoCap data. However, this method also required significant computational resources and had limitations in generalizing to unseen tasks [21].

2.3. Reinforcement Learning

Reinforcement learning has been a significant subject of research for learning complex motion skills, mainly due to its ability to learn from trial-and-error experience. Sutton and Barto's work [29] is a pioneering piece in this field, providing an understanding of the principles of RL. Early works of RL in motion skills focused on low-dimensional problems with few state-action pairs [31]. The successful application of RL to high-dimensional, continuous control tasks were achieved by Todorov *et al.* [30] with their MuJoCo physics engine, a significant tool for RL in continuous action spaces.

In terms of motion skills, several works have proposed RL methods for locomotion. The works of Peng *et al.* [22] and Heess *et al.* [12] use RL for learning locomotion in complex, dynamic environments. Notably, Heess *et al.* demonstrated locomotion skills transferable to novel tasks, showing RL's generalizability [12].

RL has also been successfully applied to manipulation tasks. Levine *et al.*'s work on end-to-end training of deep visuomotor policies [17] pioneered RL's use for manipulation, enabling a robot to perform object manipulation based solely on raw pixel input. In addition, Popov *et al.* [24] proposed data-efficient manipulation skills learning using a method called Data Aggregation, achieving superior performance on several benchmarks.

The combination of RL with imitation learning [13] and inverse RL [1] have also shown promising results in motion skill acquisition, enabling the agent to learn from expert demonstrations, thereby reducing the time and samples required for effective learning.

In spite of the significant improvement brought by RL, it often suffers from sample inefficiency, requiring many interactions with the environment to learn effectively [6]. Hafner *et al.* [11] and Kaiser *et al.* [11] proposed Dreamer and DreamerV2, respectively, to learn behaviors purely from predictions, significantly improving sample efficiency. Off-policy RL algorithms, such as SAC [10], TD3 [9], and D4PG [5], are noteworthy as they can learn from previous experiences, contributing to the solution of sample inefficiency. For instance, SAC has been used to learn diverse robotic skills from scratch [10].

Besides, some methods propose to adopt parallel learning in curriculum to improve the data efficiency. By generating massive environments with increasing difficulties, it helps the agent to learn more samples effectively. For example, in Florensa *et al.* [7], a reverse curriculum strategy was used where the agent begins learning in states near the goal and progressively learns from states further away. Justesen *et al.* [15] applied a curriculum learning approach to train a deep RL agent for playing complex video games, demonstrating the potential of curriculum strategies in RL for real-world applications. Rudin *et al.* [28] leverage a massively paralleled curriculum system to achieve rapid learning of complex locomotion skills, enabling the robot to learn to walk in just a few minutes of real-time training.

In this paper, we build upon DeepMimic [22] to enable robot Bob perform dynamic motion skills. To transfer the algorithm to a new robot, we incorporate the pipeline in DeepMimic [22] with the manually defined mass curriculum so that the new robot can gradually adapt to the motions from simple to difficult. We have proven that such a combination can help generalize the original algorithm to different robot models.

3. Method

In this paper, we use the biped robot Bob in Assignment 2 of the Digital Human course [1] to reproduce the imitation task in Deepmimic [22]. The main workflow of our algorithm is shown in Figure 1.

3.1. Joint Mapping

In order to use the MoCap data that is provided in Deepmimic [22], we need to adapt it to the kinematic structure of the biped bob. The data specifies the position and rotation of the pelvis along with the 3D rotation of eight spherical joints (as quaternions) and four revolute joints. On the other hand, bob has 44 revolute joints, which makes the matching process quite intricate. We first matched the joints based on their position. Each of the spherical joints was matched with three revolute joints connected in series. The axis of rotation of those joints along with their order dictates the Euler angle base that should be used in converting the quaternion to 3 angles that can be assigned to each joint. Finally, the relative rotation of the joint frame relative to the root frame should also be considered: the frame of each joint in the data should be rotated to align it with the first joint of the triplet that matches it. Most of the joints had the axis aligned with one or 2 axes pointing in the opposite direction of the corresponding one. A quick change of sign to the corresponding Euler angle is enough to align them. The ankle joints were the only special case, since they are spherical joints in the Deepmimic [22] MoCap data, but are only represented by two revolute joints in bob's kinematic setup. To remedy this, the quaternions were converted to Euler angles based on the order of the two available axis. The third angle can then be easily disregarded. Table 1 shows the joint correspondences along with the axes order used in the Euler angle conversion and the axes that were flipped to ensure proper alignment.

Bob's 18 joints that were not matched with the reference motions are set based on their default angles. Those joints

[1] https://github.com/Digital-Humans-23/a2

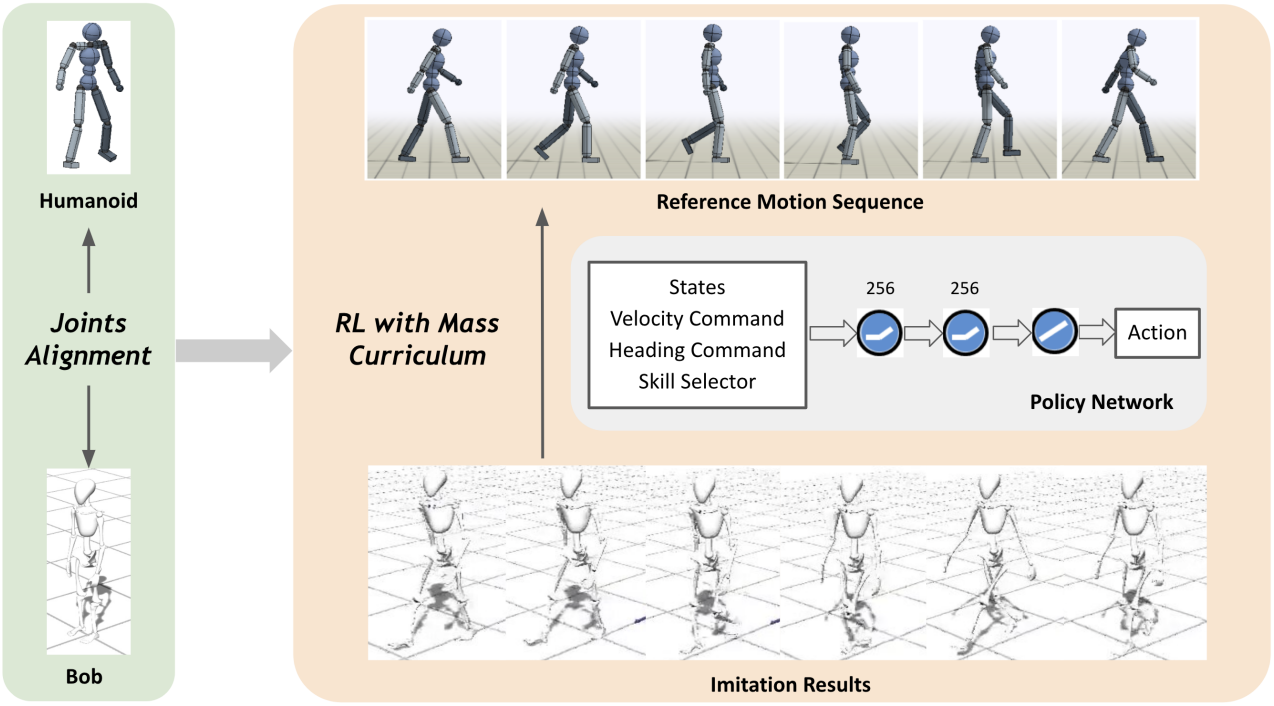

Figure 1. Main Workflow. First we align body joints between Humanoid robot and Bob robot. Then we adopt model-free Reinforcement Learning with mass curriculum training strategy to learn reference motions from MoCap data provided in Deepmimic [22]. The inputs of policy network are states, velocity and heading commands, and skill selector when training one policy for multiple motions. The network architecture is two-layer MLP and the output is action.

Deepmimic	Bob	Order	Flipped Axes
pelvis	pelvis	yzx	z
chest	upperback	zyx	z
neck	lowerneck	zyx	z
l/r hip	l/r hip	zxy	y, z
l/r knee (1D)	l/r knee	N/A	N/A
l/r ankle	l/r ankle (2D)	yxz	N/A
l/r shoulder	l/r shoulder	zxy	y, z
l/r elbow (1D)	l/r elbow	N/A	N/A

Table 1. Joint mappings from humanoid in Deepmimic [22] to Bob. "Order" denotes the order for euler angle rotations we used when converting the quaternions in Deepmimic data to euler angles for Bob's observation space. For some joints, we put a minus sign on the flipped axes of the converted euler rotation axes.

should be fixed so they don't interfere with the trained motion. Although they are removed from the action space, their rotations could still change in the physical simulation. Therefore, we tried increasing their stiffness parameters to infinity to make them as rigid as possible ensuring they have minimal changes.

To visualize the matching, we made sure to run a training with one step per episode and each time we reset bob's joints to the converted angles that we got from the MoCap data. Each time we reset the environment we increase the frame counter so we can check out the complete motion.

3.2. State and Action Spaces

Our state space includes the 3D position and 3D rotation of the pelvis point and 44-dimensional joint rotations for other joints of the humanoid. In addition, there's also a 50-dimensional velocity vector, including 3D velocity of the pelvis position, 3D velocity of the pelvis rotation, and 44-dimensional velocities for the 44 other joints. The total dimension of the state space is therefore 100. Our continuous action space includes the joint rotations to execute for the 26 joints out of the 44 joints that are mapped with deepmimic's humanoid. Note that the other 18 joints that are not mapped were originally in the action space, but we remove them and always set them to the default values before applying them in the environment, as explained in section 3.1.

3.3. Random State Initialization

In order to determine the target frame in the reference motion, we follow Deepmimic [22] to introduce a phase

variable ϕ and add it into the observation space as an additional input to the policy network. When starting an episode, we randomly sample the initial phase variable $\phi_{init} \in [0, 1]$ for loop motions or $\phi_{init} \in [0, 0.5]$ for non-loop motions. For loop motions such as walking and running, since the reference motion is short, we do not stop the training episode when reaching the end of the reference motion. Instead, we offset the pelvis position of the reference motion by the entire distance the humanoid has traveled during the reference motion to start a new loop. The corresponding frame index is $i = \phi_{init} * (N - 1)$, where N is the total number of frames in the reference motion. To initialize the observation space, we linearly interpolate (LERP) between i_{th} and $(i + 1)_{th}$ frames to get the initial joint rotations. As for the velocities we set them based on the first order Euler approximation of the velocities in the i_{th} frame. In the step function, ϕ_t for a current step t in the episode is calculated as:

$$\phi_t = \frac{dt \cdot t}{T} + \phi_{init} \tag{1}$$

where dt is the duration of a step, and T is the entire duration of the reference motion. If $\phi_t > 1$, we terminate the episode for non-loop motions. For loop motions, we take its integer part as the number of loops that the reference motion is currently in to offset the pelvis position of the target humanoid, and only keep the decimal part for ϕ_t. We again calculate $i_t = \phi_t * (N - 1)$ to get the frame index of the current timestep for the corresponding reward functions.

We observe that this randomization technique is significantly helpful for learning stable motions. As suggested in [22], the random state initialization can be interpreted as "an additional channel through which the agent can access information from the reference motion in the form of a more informative initial state distribution". Always initializing the episode to the first frame limits the agent to only learn the motion through reward functions, which converges more slowly and sometimes even leads to failed imitation.

3.4. Reward Functions

We follow [22] to define the global reward functions:

$$r_t^p = exp(-2(\sum_j ||\hat{q}_t^j \ominus q_t^j||^2)) \tag{2}$$

$$r_t^v = exp(-0.1(\sum_j ||\hat{\dot{q}}_t^j - \dot{q}_t^j||^2)) \tag{3}$$

$$r_t^c = exp(-10(||\hat{p}_t^c - p_t^c||^2)) \tag{4}$$

r_t^p is the pose reward function that encourages the observed joint rotation quaternions $\hat{q}_t^j$ to be close to the reference quaternions q_t^j (the target reference is determined based on the current phase of the environment). To calculate the quaternion difference $\ominus$, we convert the observed joint rotation Euler angles to quaternions following the orders and flipping axes in table 1. We then multiply the target quaternion with the inverse of the observed quaternion to get the relative rotation between them. We convert this quaternion to an axis-angle representation and feed the corresponding angle in the reward formula above. r_t^v is the velocity reward function. The target velocities are approximated using first degree Euler between the corresponding nearest frames. r_t^c is the center-of-mass reward. For simplicity, we define the center-of-mass point in our humanoid $\hat{p}_t^c$ to be the pelvis point and directly compare it with pelvis position of reference frames p_t^c. Due to the difference in height and limb lengths between our humanoid and Deepmimic's, we ignore the end-effector reward which is also used by Deepmimic. Similar to random state initialization, we also perform linear interpolation on i_{th} and $(i + 1)_{th}$ frames for quaternions $\hat{q}_t^j$ and euler angles $\hat{\dot{q}}_t^j$ using the frame index i_t of the current timestep calculated in section 3.3.

Our total reward function is:

$$r_t^I = w^p r_t^p + w^v r_t^v + w^c r_t^c \tag{5}$$

where the hyper-parameter weights are set as $w_p = 0.72, w_v = 0.17, w_c = 0.11$.

3.5. Velocity and Heading Commanding

Now that bob is able to walk according to the pace and direction set while generating the MoCap data, the next step is to train policies that can enable walking in different heading angles and at different speeds. To achieve that we need first to augment the observation tensor to include our goal parameters: the speed v and the heading angle θ, indicating the heading on the horizontal plane with $\theta = 0$ aligned with the z axis. θ can be set to any value between 0 and 2π, and v is chosen to be between 0.5 and 2 m/s, since the provided data for walking is collected with a walking speed of 1 m/s whereas the jogging data has a speed of 2.6 m/s. At the beginning of each episode, we sample a random heading and speed from the respective intervals in order to train policy that can handle all different goals that the user might choose. The pelvis position, orientation and velocity used in the initialization of the episode are modified to match the new configuration with the new heading using the equations:

$$r = \sqrt{x_{target}^2 + y_{target}^2} \tag{6}$$

$$x_i = r.(v_{goal}/v_{walking}).sin(\theta_{goal}) \tag{7}$$

$$y_i = r.(v_{goal}/v_{walking}).cos(\theta_{goal}) \tag{8}$$

$$yaw_i = yaw_{target} + \theta_{goal} \tag{9}$$

$$v_{xi} = v_{goal}.sin(\theta_{goal}) \tag{10}$$

$$v_{yi} = v_{goal}.sin(\theta_{goal}) \quad (11)$$

All other parameters are set based on the target value computed by interpolation using the random phase ϕ

To make sure bob follows the speed and heading goals, we replace the center of mass reward and the component of the velocity reward that is computed using the pelvis velocity. We substitute those rewards with a task goal reward which was formulated similarly in Deepmimic

$$r_{goal} = exp(-2.5(max(v_{goal} - v_{proj}, 0))^2) \quad (12)$$

v_{proj} is the projection velocity vector along the heading.

$$v_{proj} = v_{obs}^T.d_{heading} \quad (13)$$

In this formulation, we do not penalize having a speed larger than the set goal.

3.6. Mass Curriculum

Using the random state initialization in section 3.3 and the reward functions in section 3.4 enables us to train a stable walking motion. However, when we extend it to more complex motions such as running and jumping, we observe that the bob always tends to fall down and never successfully imitates to step forward.

We apply a mass curriculum to solve this issue to let our robot gradually adapt to the motions from simple to difficult. The masses for all body parts are initialized to 0.01 at the beginning of an episode. When the maximum episode length an episode ever reached gets to 40, we increase the masses by 25%. When it increases by another 10, we increase the masses by another 25%. We repeat this until 100% of the real masses are reached.

3.7. Training One Policy for Multiple Motions

In addition to training a different policy for each single motion, we also attempt to train a single policy for multiple motions via a skill selector, as proposed by [22]. To be precise, we augment the observation with a goal vector in form of a one-hot vector representing one of the specific motions to train. We train our skill-selector on walk, run, and jump, the 3 motions we successfully trained with single policies, with a mass curriculum. The overall reward function stays the same as equation 5, but r_t^I is computed as r_t^i on the selected motion i. When starting an episode, we randomly sample an integer from [0,3] and select it as the motion to learn for the current episode, and load the target motion accordingly.

4. Results

4.1. Joint Mapping Results

Figure 2, 3, 4 shows 3 example motions mapped to Bob from a Deepmimic reference. To see more examples and their animations, please visit our github repository.

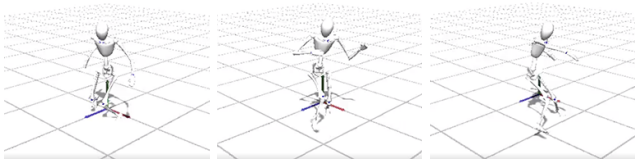

Figure 2. Joint Mapping Results: Punch

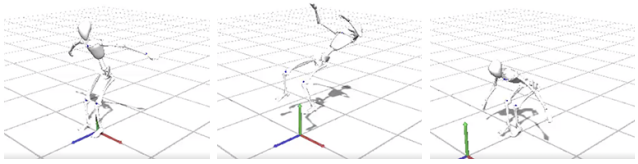

Figure 3. Joint Mapping Results: Backflip

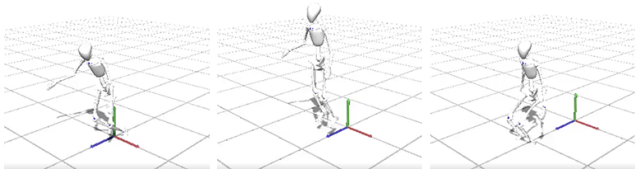

Figure 4. Joint Mapping Results: Backflip

4.2. One Policy for Each Motion

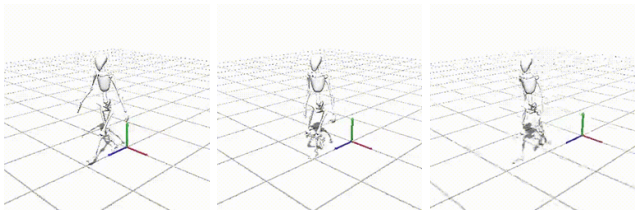

Figure 5. Training Results: Walk

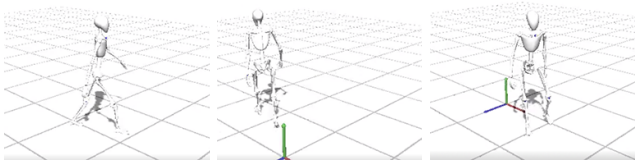

Figure 6. Training Results: Walk, Random Velocity and Heading

As shown in figure 5, we successfully learn walking with the reward functions in section 3.4 without any curriculum. The bob could walk continuously until the episode ends

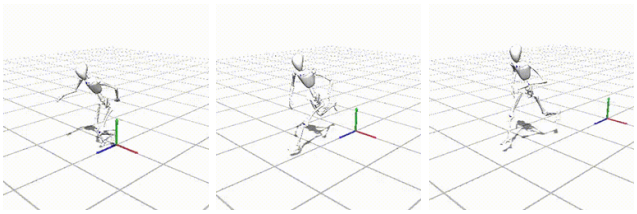

Figure 7. Training Results: Run

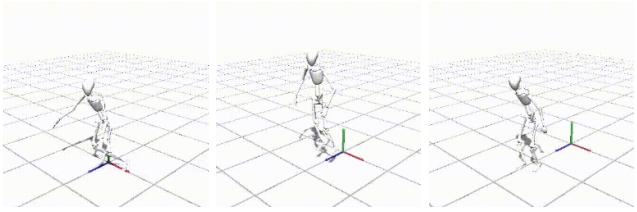

Figure 8. Training results: Jump

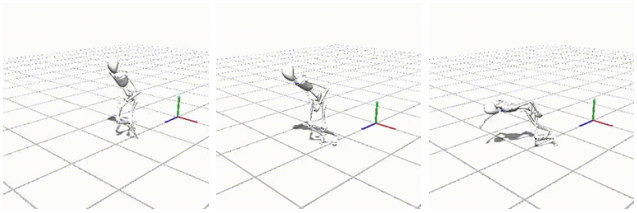

Figure 9. Ablation Results: Run without Mass Curriculum

without falling down. When adding the velocity and heading goal, as shown in figure 6, it successfully learns to walk in random directions with random speed, although the bob could not stabilize for as long as walking in the reference direction. Figure 7 and 8 shows that we successfully learn running and jumping with mass curriculum, although also not as stable as walking. Figure 9 provides a comparison that trains running without mass curriculum. In such case, the bob consistently falls down from the initialized pose and position and never properly learns to step forward.

4.3. Skill Selector

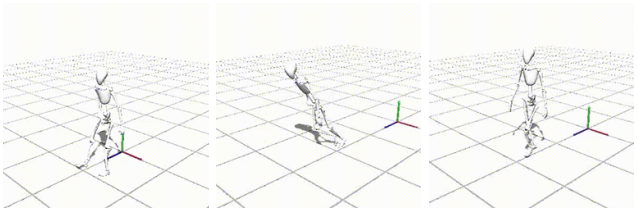

Figure 10. Training Results: Skill Selector. From left to right: jump, run, walk. Only walking could be successfully imitated.

Requiring a policy network to jointly learn multiple motions via one-hot encoding turns out to be challenging. As figure 10 shows, even with the mass curriculum deployed, our skill selector only manages to imitate walking well, while the bob falls in both jumping and running.

5. Summary

5.1. Conclusion

Learning highly dynamic skills by imitating reference motion capture clips represents an exciting frontier in the cross-disciplinary realm of robotics, animation, machine learning, and human-robot interaction. In this research, we delved into this intriguing domain by building upon a reinforcement learning-based method [22], utilizing reference motion clips to facilitate the learning process. The proposed method is deployed on a biped robot, 'Bob', enabling it to perform dynamic tasks such as walking, running, and jumping.

This approach breaks away from the constraints of traditional methods based on kinematic controllers and handcrafted rules, which often struggle with complex, agile movements. By incorporating advanced techniques from deep learning, reinforcement learning, and imitation learning, we significantly enhance the ability of Bob to execute agile behaviors. Furthermore, our training strategy generalizes to other biped robots with different shapes, mass, and dynamic models, thereby amplifying its practical significance.

A distinctive aspect of this methodology is the random state initialization, which we borrowed from DeepMimic [22], adding a phase variable to the observation space. This variable serves as an effective mechanism for the agent to access informative initial state distribution, enabling faster convergence of learning and successful imitation of reference motions. Our customized reward functions bolster the learning process by promoting optimal pose, velocity, and center-of-mass.

The mass curriculum strategy, introduced in our research, facilitates the training of complex motions such as running and jumping. This strategy gradually increases the body mass during the episode, allowing the Bob robot to adapt to challenging motions without losing balance.

Simultaneously, we implemented a method that trains a policy enabling the robot to walk in any heading direction or speed by augmenting our observation space to include our goal parameters and adding a reward encouraging the robot to track the desired goal velocity.

Finally, we tackled the ambitious goal of training a single policy for multiple motions, leveraging a skill-selector. The success in training a skill-selector for walking, running, and jumping motions further demonstrates the potential of the approach.

5.2. Limitations and Future Works

However, there are certain limitations. The complexity and high-dimensionality of motion data pose challenges to the efficiency and robustness of our method. We also assume a perfect physics engine and sensor data, which is often not the case in real-world applications. The current method still cannot stabilize the motions for long and lacks the ability to cope with unexpected disturbances and rapidly changing environments. In addition, learning multiple motions jointly via a skill selector is more challenging than learning one policy for each motion since it takes significantly longer to train to reach a common policy that works for all motion. The Deepmimic framework suggests training a composite policy from the individual policy obtained previously. This method has proven to be more promising especially if the skills that we want to use are very diverse.

Looking forward, several exciting directions exist for extending our work. First, integrating techniques for learning robust policies against external disturbances, *e.g.* adding noise during training, could be crucial for practical applications. Second, exploring approaches that reduce the reliance on perfect sensor data, such as unsupervised and self-supervised learning methods, could increase the method's applicability. Third, our approach can be enriched by investigating multi-task learning and meta-learning strategies that would allow a single policy to learn a broader range of skills and adapt quickly to new tasks. Lastly, regarding learning one policy for multiple motions, a learnable latent code for each motion could be potentially added to make strengthen the expressiveness of one-hot encoding in the skill selector. Meanwhile, the other 2 composite methods proposed in Deepmimic, i.e., multi-clip reward and composite policy can also be attempted. Due to our limited time in this project, we leave them for future works.

6. Contributions of team members

- Joint Mapping/Visualization from Deepmimic to Bob: Anthony, Xiyi
- Implementing the general framework (Random state initialization, reward formulation...): Xiyi, Anthony
- Training individual motions: Anthony, Xiyi, Jin, Kexin
- Implementing of the mass curriculum: Xiyi, Anthony
- Training with mass curriculum: Kexin, Xiyi
- Implementing velocity and heading commanding: Anthony
- Training velocity and heading commanding: Anthony, Xiyi
- Implementing and training skill selector: Xiyi
- Writing report: Xiyi, Jin Kexin, Anthony

7. Acknowledgements

This is a course project for Digital Humans (263-5806-00L) at ETH Zurich, instructed by Prof. Stelian Coros and Prof. Siyu Tang. We appreciate our supervisor Fatemeh Zargarbashi for her help with the project. Our implementation is based on the code of assignment 2 of the course.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004. 3

[2] Kfir Aberman, Rundi Wu, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Learning character-agnostic motion for motion retargeting in 2d. *arXiv preprint arXiv:1905.01680*, 2019. 2

[3] Okan Arikan and David A Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics (TOG)*, 21(3):483–490, 2002. 2

[4] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando De Freitas. Playing hard exploration games by watching youtube. *Advances in neural information processing systems*, 31, 2018. 2

[5] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018. 3

[6] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021. 3

[7] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017. 3

[8] Yuni Fuchioka, Zhaoming Xie, and Michiel van de Panne. Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors. *arXiv preprint arXiv:2210.01247*, 2022. 2

[9] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018. 3

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 3

[11] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International*

conference on machine learning, pages 2555–2565. PMLR, 2019. 3

[12] 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. 3

[13] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016. 3

[14] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017. 2

[15] Niels Justesen, Philip Bontrager, Julian Togelius, and Sebastian Risi. Deep learning for video game playing. *IEEE Transactions on Games*, 12(1):1–20, 2019. 3

[16] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *ACM SIGGRAPH 2008 classes*, pages 1–10. 2008. 2

[17] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018. 3

[18] C Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics (TOG)*, 24(3):1071–1081, 2005. 2

[19] C Karen Liu, Aaron Hertzmann, and Zoran Popović. Composition of complex optimal multi-character motions. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 215–222, 2006. 2

[20] Frank Merat. Introduction to robotics: Mechanics and control. *IEEE Journal on Robotics and Automation*, 3(2):166–166, 1987. 2

[21] Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4):39–1, 2020. 2

[22] 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, 3, 4, 5, 6, 7

[23] 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. 2

[24] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017. 3

[25] Jovan Popović, Steven M Seitz, Michael Erdmann, Zoran Popović, and Andrew Witkin. Interactive manipulation of rigid body simulations. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 209–217, 2000. 2

[26] Zoran Popović and Andrew Witkin. Physically based motion transformation. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20, 1999. 2

[27] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: Texturing and synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 501–508, 2002. 2

[28] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022. 3

[29] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 3

[30] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012. 3

[31] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992. 3