

# Learning to Get Up

Tianxin Tao

University of British Columbia  
Canada  
taotianx@cs.ubc.ca

Ruiyu Gou

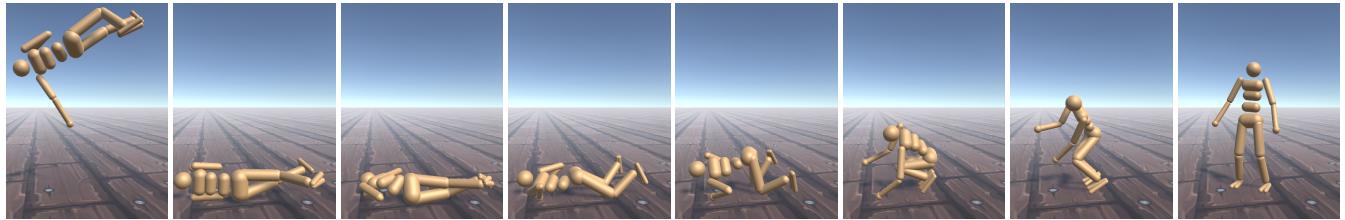
University of British Columbia  
Canada  
reyget42@gmail.com

Matthew Wilson

University of British Columbia  
Canada  
mattwilsonmbw@gmail.com

Michiel van de Panne

University of British Columbia  
Canada  
van@cs.ubc.ca



**Figure 1: We develop a three-stage framework for generating natural and diverse human get-up behaviors using deep reinforcement learning. The learned control policies enable characters to get up from any initial fallen pose, and either slowly or quickly.**

## ABSTRACT

Getting up from an arbitrary fallen state is a basic human skill. Existing methods for learning this skill often generate highly dynamic and erratic get-up motions, which do not resemble human get-up strategies, or are based on tracking recorded human get-up motions. In this paper, we present a staged approach using reinforcement learning, without recourse to motion capture data. The method first takes advantage of a strong character model, which facilitates the discovery of solution modes. A second stage then learns to adapt the control policy to work with progressively weaker versions of the character. Finally, a third stage learns control policies that can reproduce the weaker get-up motions at much slower speeds. We show that across multiple runs, the method can discover a diverse variety of get-up strategies, and execute them at a variety of speeds. The results usually produce policies that use a final stand-up strategy that is common to the recovery motions seen from all initial states. However, we also find policies for which different strategies are seen for prone and supine initial fallen states. The learned get-up control strategies often have significant static stability, i.e., they can be paused at a variety of points during the get-up motion. We further test our method on novel constrained scenarios, such as having a leg and an arm in a cast.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*SIGGRAPH '22 Conference Proceedings, August 7–11, 2022, Vancouver, BC, Canada*  
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-1-4503-9337-9/22/08...\$15.00  
<https://doi.org/10.1145/3528233.3530697>

## CCS CONCEPTS

- Computing methodologies → Animation; Reinforcement learning.

## KEYWORDS

Physics-based character animation, learning curriculum

### ACM Reference Format:

Tianxin Tao, Matthew Wilson, Ruiyu Gou, and Michiel van de Panne. 2022. Learning to Get Up. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings (SIGGRAPH '22 Conference Proceedings)*, August 7–11, 2022, Vancouver, BC, Canada. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3528233.3530697>

## 1 INTRODUCTION

Getting up from the ground to a standing posture is a natural and effortless skill for most humans. Simulated characters will similarly need ways of recovering from the arbitrary fallen states if they are to see broad adoption in simulated worlds. A common current approach is to learn a policy that simply imitates a relevant motion capture clip, which thereby bypasses the difficult problem of needing to discover the best get-up strategy. However, humans get up using a wide variety of styles and speeds, and can rapidly improvise when faced with new circumstances, e.g., having a leg in a cast. This quickly makes it impractical to capture all possible get-up motions in advance.

The get-up problem can also be resolved without recourse to motion capture data, and is known as a particularly challenging problem to solve. The learned policies, however, often exhibit erratic behaviors that are unlike those commonly observed in humans [Pinneri et al. 2020; Tassa et al. 2020]. We speculate that the strong actuation limits made available in the simulated characters

allow for solution modes to be found, while also leading to these overly dynamic and often-unnatural solutions.

Our work develops a learning framework that achieves more natural get-up motions, based on the assumption that human get-up motions are typically weak and slow. We do this using a three-stage strategy: (i) learn a get-up control policy for a strong character, which significantly eases the discovery of solutions modes; (ii) a curriculum is used to adapt the control policy to a progressively weaker character, as implemented via decreasing torque limits; (iii) a control policy is learned which imitates the outcome of the previous stage at speeds up to 5× slower than the original speed.

Our method generates a diverse range of get-up styles across multiple runs. The discovered motions often have significant static stability, i.e., they can be fully paused at many points in time. Learned control policies often exhibit a dominant get-up mode, e.g., always first reverting to a prone position before getting up, but can also exhibit multiple modes, e.g., using different strategies for prone and supine initial states. We visualize the resulting motion trajectories using t-distributed stochastic neighbor embedding (t-SNE) plots to better understand their structure and diversity. Lastly, ablations show the necessity of the various components of our approach. For example, we find that directly introducing regularization terms for control effort and motion speed leads to a failure to learn.

Our principal contributions are as follows: (1) We introduce a method to learn get-up control policies, specifically targeting the generation of natural get-up motions without recourse to motion capture data. At the core of our framework is the idea to first discover successful get-up modes, and then to learn weak-and-slow versions of these modes. (2) We visualize and analyze the behavior of multiple learned control policies across multiple initial states. This reveals a diversity of strategies, as seen across the controllers arising from multiple runs, as well as for a given controller in response to different initial states. We release the code at [https://github.com/tianxintao/get\\_up\\_control](https://github.com/tianxintao/get_up_control).

## 2 RELATED WORK

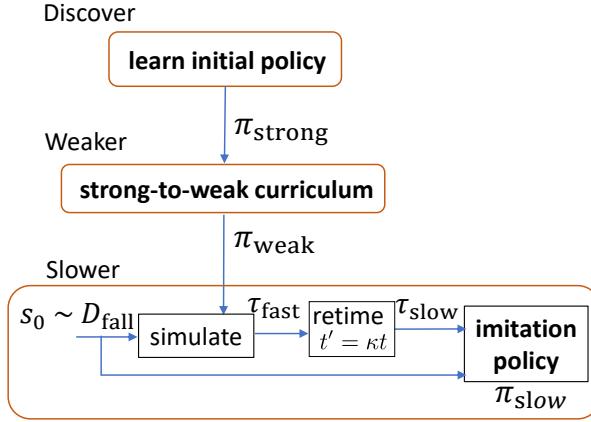
Developing physics-based controllers for character animation is a long-standing research problem. For a thorough history on this topic, we refer the reader to a survey paper [Geijtenbeek and Pronost 2012]. Early work demonstrates significant success on locomotion tasks and often relies on compact manually-designed feedback rules and finite state machines (FSM), e.g., [Coros et al. 2010; Hodgins et al. 1995; Wang et al. 2009, 2010; Yin et al. 2007]. Trajectory optimization has also been used with considerable success to generate locomotion control for both human and non-human characters [Kim et al. 2021; Mordatch et al. 2013; Wampler et al. 2014]. Various motion tracking controllers have been proposed to imitate available motion capture data using model-based approaches [Lee et al. 2010, 2014; Ye and Liu 2010; Yin et al. 2007] or sampling-based methods [Hämäläinen et al. 2014, 2015; Liu et al. 2016, 2015, 2010].

With the rapid progress of deep learning machinery, deep reinforcement learning (DRL) has become a promising method to learn physics-based controllers. Heess et al. [Heess et al. 2017] proposed a framework built on policy-gradient methods to learn a wide range of locomotion skills although the resulting motion suffers from a lack of realism. Many techniques have been proposed to enhance

the motion quality. Symmetry constraints have been applied as an inductive bias to produce realistic motions [Abdolhosseini et al. 2019; Yu et al. 2018]. Yin et al. [Yin et al. 2021] proposed the use of pose variational autoencoders in support of learning natural athletic motions. Several works show that designing a curriculum on the task parameters can assist in discovering complicated motions, including dressing and traversing stepping stones [Clegg et al. 2020; Xie et al. 2020]. Alternative actuation models with muscle activation have also been proposed to learn more human-like motions [Jiang et al. 2019; Lee et al. 2019; Peng and van de Panne 2017]. Additionally, DRL is widely studied to learn tracking controllers for reference motions. Peng et al. [Peng et al. 2018] proposed a DRL-based framework with random state initialization (RSI) and early termination to learn controllers capable of imitating highly diverse motions. DReCon [Bergamin et al. 2019] combines the motion matching system and imitation controller to track the motion data. Building on this, the idea has been extended to address the problem of unlabeled motion data by training a recurrent neural network to predict the reference pose in the next frame [Park et al. 2019]. A hybrid action space of torque and proportional-derivative (PD) control is proposed to accelerate the training process [Chentanez et al. 2018]. The methods have also recently been made much more scalable, e.g., [Won et al. 2020].

Aside from physics-based approaches, kinematics-driven animation methods have also seen significant improvements along with novel deep learning architectures in recent years. Generative animation models are commonly built upon Variational Autoencoders (VAE) [Ling et al. 2020; Rempe et al. 2021], Long Short-Term Memory (LSTM) [Harvey and Pal 2018; Harvey et al. 2020; Martinez et al. 2017] and mixture of experts [Starke et al. 2019, 2020; Zhang et al. 2018]. We refer readers to a survey paper for a more detailed overview on this topic [Hoyet et al. 2021].

Learning a get-up controller has been of interest to computer animation and robotics. Pioneering work by Morimoto et al. [Morimoto and Doya 1998] proposed a hierarchical reinforcement learning framework to master get-up motions on a simplified 2D walker model. Kanehiro et al. [Kanehiro et al. 2007, 2003] developed a get-up strategy with a manually designed contact graph and careful mechanical calibration to the robot. Bilateral symmetry constraints were proposed to master natural stand-up behavior for humanoid robots [Jeong and Lee 2016]. A wide range of sampling techniques have been used to successfully discover get-up motions, e.g., [Hämäläinen et al. 2014; Pinneri et al. 2020]. Despite their success, these sampling methods commonly suffer from limited motion quality and are less suited to online use from arbitrary initial states than direct control inference via a DRL control policy. If relevant motion capture data is available, motion tracking methods with DRL are also capable of generating get-up motions, e.g., [Chentanez et al. 2018; Merel et al. 2017]. Online trajectory optimization methods, i.e., model predictive control, are also capable of generating get-up motions for humanoids [Tassa et al. 2012], albeit with limited motion quality and, to the best of our knowledge, restricted to dynamic versions of the motion. Multiple trajectory optimizations can be structured in a tree-like fashion in order to support reuse of the optimization results, for use from a variety of initial states [Borno et al. 2017]. This uses large torque limits (300 Nm) and relies on local PD-control feedback for stability, rather than closed-loop full



**Figure 2: System overview.** Our system explores an initial policy with a strong character, then refines the motion from a strong character to a weak character. Finally, we train an imitation policy to track the retimed trajectory produced by the weak policy.

state feedback. In contrast to prior work, we propose a framework which does not need a reference motion, achieves fast runtime performance from arbitrary fallen states on flat ground, and that can produce slow-and-weak motions that are more representative of most human get-up motions.

Human get-up motions are rich and varied in nature, with documented demonstrations of at least 52 ways to get up [YouTube 2013], including a variety of ways to get up without the intermediate use of hands [YouTube 2019]. A variety of the methods described require a degree of athleticism and flexibility. At the other end of the spectrum is a slow and low-effort strategy described in support of recovery from falls in the elderly, e.g., [Adams and Tyson 2000]. Our work aims to demonstrate how current DRL algorithms can learn controllers that can discover get-up strategies that can be at the slower-and-weaker end of the spectrum of possible strategies.

### 3 PRELIMINARIES

We formulate the DRL problem as a standard Markov Decision Process (MDP). MDP can be defined by states  $s_t \in \mathcal{S}$ , actions  $a_t \in \mathcal{A}$ , a dynamics function  $p(s_{t+1}|s_t, a_t)$  denoting the probability of reaching state  $s_{t+1}$  with state-action combination  $(s_t, a_t)$ , a discount factor  $\gamma \in [0, 1]$  and a reward function  $R(s_t, a_t)$ . The product of DRL is a policy  $\pi_\theta(s_t)$  parameterized by  $\theta$  interacting with an environment. At each control timestep, the policy selects an action  $a_t$  given the state  $s_t$ . Then, the agent executes the action  $a_t$  and the current state  $s_t$  is transformed into the next state  $s_{t+1}$  according to the dynamics function. A scalar feedback  $R(s_t, a_t)$  is returned as the reward function. The training objective of DRL is to maximize the expected return as:

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[ \sum_{t=0}^T \gamma^t R(s_t, a_t) \right] \quad (1)$$

where  $p_\theta(\tau)$  represents the probability of experiencing trajectory  $\tau = (s_0, a_0, s_1, a_1, \dots, a_{T-1}, s_T)$  following policy  $\pi_\theta$  and  $T$  is a finite integer denoting the episode length.

We choose Soft Actor-Critic (SAC) [Haarnoja et al. 2018] as the DRL algorithm to train all the tasks in this work. SAC is widely considered as a state of the art of model-free DRL method with excellent sampling efficiency. Besides maximizing the expected return  $J_\theta$ , SAC adopts the idea of entropy regularization to achieve a balance between exploration and exploitation. SAC learns a policy network  $\pi_\theta(s_t)$  as actor and an action-value function  $Q_\phi^\pi(s_t, a_t)$  as critic, which are commonly represented by multilayer perceptrons (MLP). In locomotion tasks, the action usually represents the torques of the joint motors, which are commonly normalized to  $[-1, 1]$ . To match the action bounds, SAC usually applies a tanh function on the output as the squashing function.

### 4 SYSTEM OVERVIEW

We illustrate our learning pipeline in Fig. 2. We split the overall training process into three sequential stages: (1) initial policy exploration with strong characters; (2) low-energy motion discovery through a strong-to-weak curriculum; and (3) slow motion refinement via motion imitation. Discovering get-up motions from scratch using DRL is particularly challenging in that the exploration process can readily become trapped in local minima, which results in variants of a kneeling motion. To avoid the issue of local minima, we learn an initial get-up controller  $\pi_{\text{strong}}$  with a character with high torque limits because such a character can explore a larger portion of state and action space. By exploring diverse states and actions, the DRL algorithm is more likely to encounter high reward region to discover a get-up solution mode quickly. However, although a high-strength character is beneficial for exploration, low-strength motions are usually more natural. To enhance motion quality, we therefore introduce a second stage to progressively learn a policy  $\pi_{\text{weak}}$  suitable for much weaker versions of the character. In practice, we combine the training of the initial policy  $\pi_{\text{strong}}$  and its adaption to low-strength actions into one training process. Once the test reward of  $\pi_{\text{strong}}$  reaches a threshold  $\omega$ , the strong-to-weak curriculum is automatically activated.

After the strong-to-weak curriculum, we obtain a state-indexed physics-based controller  $\pi_{\text{weak}}$  generating get-up motions with low energy cost. To generate slower movements, we introduce a motion tracking objective for a third controller  $\pi_{\text{slow}}$  to imitate the retimed trajectories  $\tau_{\text{slow}}$ . Given an initial state  $s_0$ , we first generate a fast get-up trajectory  $\tau_{\text{fast}}$  using policy  $\pi_{\text{weak}}$ . This is then retimed by a factor of  $\kappa$ ,  $\kappa \in [0, 1]$ . The newly trained controller,  $\pi_{\text{slow}}$ , can produce motions up to  $5\times$  slower. At run time, users can specify the value of  $\kappa$  to adjust the speed of get-up motions. Moreover, we also train the controller  $\pi_{\text{slow}}$  to maintain balance while standing by tracking a manually designed standing pose.

Our learning pipeline can discover different get-up strategies from prone and supine positions by simply initializing the training with different seeds. We demonstrate and analyze the diverse get-up behaviors using t-SNE plots. We also provide pseudocode for the three stages in the supplementary material.

## 5 ‘DISCOVER’ AND ‘WEAKER’ STAGES

In common physics-based simulators, the characters are modelled with torque limits to define the strength of joint motors. Well-designed torque limits play an essential role in the quality of motion. Poorly designed torque limits can lead to unnatural motion [Merel et al. 2018; Peng et al. 2018] and degraded learning performance [Abdolhosseini 2019]. Therefore, we design the default torque limits  $\mathcal{T}$  according to documented values for humans [Grimmer 2015], and scale them with respect to the weights of body parts.

We start the training with the designed torque limits  $\mathcal{T}$  of the humanoid character to explore an initial policy  $\pi_{\text{strong}}$ . Then, the initial policy  $\pi_{\text{strong}}$  is refined through a strong-to-weak curriculum, where we keep track and update the current torque limits of the character. Once the accumulated minimum test reward of the policy over multiple episodes reaches a specified threshold  $\omega$ , we advance the torque limit curriculum by setting the torque limits to be  $\beta^i \times \mathcal{T}$  at the  $i^{\text{th}}$  stage of the curriculum, where  $\beta \in [0, 1]$  is a hyperparameter. To enforce the torque limits on the humanoid character, we design our policy at the  $i^{\text{th}}$  stage of the curriculum  $\pi_{\text{weak}}$  to output actions bounded by  $[-\beta^i, \beta^i]$  by modifying the squashing function to  $\beta^i \times \tanh(\cdot)$ . Alternatively, the character configuration can be changed along with the curriculum in the simulation, and the action space is always kept to  $[-1, 1]$ . However, the unchanged action space will confuse the off-policy DRL algorithm because actions collected at different stages of the curriculum then have different meanings.

At the  $i^{\text{th}}$  stage of the strong-to-weak curriculum, we sample the torque limit multiplier  $\beta^i$  from a Gaussian distribution  $\mathcal{N}(\beta^i, \epsilon)$ ,  $\epsilon = 0.04$  at the beginning of each episode. The torque limit multiplier is treated as a sampled value rather than a constant because this helps smooth the otherwise discrete nature of the curriculum.

Previous work commonly employs curriculum on the task objective such as jump heights and stepping stone positions [Xie et al. 2020; Yin et al. 2021], which specifies the ultimate task objective with prior knowledge. In our case, the lowest feasible torque limits for diverse get-up strategies are unknown. Thus, we trigger the end of the curriculum based on the number of simulation steps taken at the current stage of the curriculum. Intuitively, the curriculum ends when the current stage requires more gradient steps than a threshold  $M$ , which indicates that the current task is too difficult under the torque limit constraints. Rather than a constant, the threshold  $M$  grows as curriculum advances since discovering very low-energy get-up motions becomes more challenging with lower torque limits. We associate the threshold with the number of steps taken at the last stage of the curriculum  $N_{i-1}$ , and define the threshold at stage  $i$ ,  $M_i$ , according to  $M_i = \text{clip}(1.5 \times N_{i-1}, N_{\min}, N_{\max})$ , where  $N_{\min}$  and  $N_{\max}$  are hyperparameters defining minimum and maximum steps for all the curriculum stages.

## 6 ‘SLOWER’ STAGE

To produce slow human-like get-up motions, we introduce a third stage that performs imitation-learning of the linearly retimed version of the trajectories produced by the get-up policy  $\pi_{\text{weak}}$ . Specifically, on every episode reset to an initial state  $s_0$ , we iteratively query the get-up policy  $\pi_{\text{weak}}$  to interact with the environment and generate a state trajectory  $\tau_{\text{fast}} = (s_0, s_1, \dots, s_T)$  of length  $T$ . During

training, we sample a constant  $\kappa$  uniformly between  $\kappa_{\text{low}}$  and  $\kappa_{\text{high}}$  ( $0 < \kappa_{\text{low}} \leq \kappa_{\text{high}} < 1$ ) as the retiming coefficient for slow get-up trajectories  $\tau_{\text{slow}}$ . For retiming, we use linear interpolation on the state trajectory over  $[0, T]$ .

To accelerate training, random state initialization (RSI) [Peng et al. 2018] can be used to initialize an episode from a randomly chosen state on the reference trajectory. We adapt RSI to our acyclic motions using a variant we call  $\epsilon$ -RSI.  $\epsilon$  is a scalar between  $[0, 1]$ . With probability  $1-\epsilon$ , RSI is adopted; otherwise, the episode is started from the beginning. At training time,  $\epsilon$ -RSI increases the probability of encountering states started from the beginning such that the controller will focus more on achieving the get-up task from end to end. We compare the performance of  $\epsilon$ -RSI, RSI and without RSI in the supplementary material. In addition, we apply early termination to the episode if the current state diverges too much from the reference motion.

Given a retimed reference trajectory, the policy  $\pi_{\text{slow}}$  aims to imitate it. We choose the PD-controller as the actuation model to compute the joint motor torque. To compute the target orientations  $q$ , the policy  $\pi_{\text{slow}}$  outputs the residual value  $q_r$  added to the reference orientation  $q'$  supplied by the reference trajectory  $\tau_{\text{slow}}$ :  $q = q_r + q'$ . The user can adjust the speed of the get-up motion by controlling the value of the retiming coefficient  $\kappa$ .

After the get-up motion, the most common and natural succeeding movement is that of quiescent stance. In support of this, we also train the controller  $\pi_{\text{slow}}$  to maintain balance in a natural pose. The training objective is switched to simply track a generic static standing pose after  $\frac{T}{\kappa}$  steps into the episode.

## 7 EXPERIMENT SETUP AND TASK SPECIFICATION

We test the proposed method and explore its performance on a regular humanoid model, as well as a modified humanoid model with a leg and an arm in a cast and a humanoid character with a missing arm. This involves the design of the rewards for the Discover and Weaker stages, as well as the imitation rewards specific to Slower stage. We adopt the reward function design proposed in [Tassa et al. 2020] for nearly all the training tasks in this work. In the interest of space, we refer the reader to the supplemental material for the low-level details of these generic types of rewards as well as the implementation details.

The get-up task starts from a rag-doll fall at 1.5m above the ground with a randomized pose. During the rag-doll fall, actions are randomly sampled according to  $a \sim \mathcal{N}(0, 0.1)$ , to model additional stochasticity in the initial states. The rag-doll fall stage lasts for a fixed duration of 80 control steps when the humanoid collides with the ground and remains in a lying pose on the ground afterwards. Then, the controller begins to provide the joint motor torques at each control timestep to accomplish the get-up objective.

**Exploring Initial Policies and its Low-energy Variants.** The character mainly focuses on finding a coarse get-up solution by maximizing the head height. Each training episode ends when it exceeds 250 steps without any early termination criteria. The state variable  $s_{\text{weak}}$  contains the following attributes: (1) joint angles and velocities in the local coordinate, (2) head height, center of mass velocity in the world coordinate, (3) end-effector positions in

the egocentric coordinate, (4) projection of the torso orientation vector to the z axis of the world coordinate,  $o_{\text{torso}}$ , (5) the character strength parameter at  $i^{\text{th}}$  stage of the curriculum,  $\beta^i$ ,  $0 \leq \beta \leq 1$ . The state variable is designed to be agnostic to the facing direction of the humanoid. We further explain the details of the state variables in the supplementary material. The action space is normalized to  $[-\beta^i, \beta^i]$ , which is scaled by the default torque limits  $\mathcal{T}$  in the simulation to compute the joint motor torques.

The reward function consist of the following terms: (1)  $r_h$ : maximize the head height, (2)  $r_{\text{straight}}$ : keep the torso vertically straight, (3)  $r_{v_{\text{com}}^{\text{xy}}}$ : prevent the character from walking or running, (4)  $r_{\text{feet}}$ : constrain the distance between two feet. The detailed explanation for each reward term can be found in App. B. The final reward for the get-up task can be summarized as:

$$R_{\text{weak}} = r_h \cdot r_{\text{straight}} \cdot r_{v_{\text{com}}^{\text{xy}}} \cdot r_{\text{feet}}. \quad (2)$$

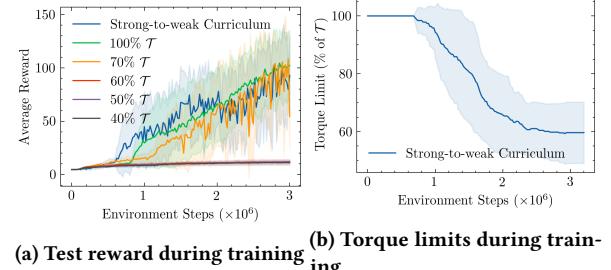
**Slow Get-up Motions.** We aim to train a get-up controller receiving different speed commands by specifying the retiming coefficient  $\kappa$ . We train a motion tracking controller  $\pi_{\text{slow}}$  with DRL by imitating the retimed trajectory  $\tau_{\text{slow}}$ . To keep track of the retimed trajectories, we maintain two simulation environments in parallel: one environment that provides the fast reference trajectory for retiming by iteratively querying the previous weak policy  $\pi_{\text{weak}}$ , another environment that tracks the retimed slow trajectories  $\tau_{\text{slow}}$ . At the beginning of each episode, we obtain a fast get-up trajectory  $\tau_{\text{fast}}$  until the head height  $h_{\text{head}}$  is above  $1.2m$ , which is later retimed to a slower trajectory  $\tau_{\text{slow}}$  through linear interpolation. The training objective is to minimize the distance over a few key attributes between the controlled character motion and the retimed kinematic motion. The maximum length of each episode is determined by the fast trajectory length  $T$  and the retiming coefficient  $\kappa$  as  $\frac{T}{\kappa}$ . In addition, the episode will be terminated when the center of mass height deviates from the reference by  $0.5m$ .

We design the imitation reward  $R_{\text{slow}}$  consisting of the following elements: (1)  $r_{\text{com}}$ : track the center of mass height, (2)  $r_{\text{ori}}$ : track the torso orientation vectors projected to the vertical axis, (3)  $r_{\text{hip}}$ : track the hip joint velocity to avoid oscillatory behavior. The precise definition of each term can be found in App. B. Thus, the final reward for the imitation phase can be expressed as:

$$R_{\text{slow}} = r_{\text{com}} \cdot r_{\text{ori}} \cdot \frac{r_{\text{hip}} + 2}{3} \quad (3)$$

Furthermore, we train the standing task to maintain balance concurrently with the same policy  $\pi_{\text{slow}}$  by switching to a new reward function  $R_{\text{balance}}$  after  $\frac{T}{\kappa}$  steps into the episode. The controller is trained to maintain balance for another 100 timesteps unless early termination is triggered. The early termination will be met if the center of mass height is below  $0.5m$ . We manually designed a single standing pose  $\hat{q}$  for the imitation policy  $\pi_{\text{slow}}$  to mimic. The designed standing pose  $\hat{q}$  is concatenated to the end of reference trajectory  $\tau_{\text{slow}}$ . The reward function for the balancing task mainly reuses reward terms from previous tasks, including  $r_{v_{\text{com}}^{\text{xy}}}$ ,  $r_{\text{straight}}$  and  $r_{\text{com}}$ . We implement a new reward term  $r_{\text{pose}}$  to track the joint rotations of the designed standing pose. Details of the standing reward design are provided in App. B. The final reward for the rotation task can be expressed as:

$$R_{\text{balance}} = r_{v_{\text{com}}^{\text{xy}}} \cdot r_{\text{straight}} \cdot r_{\text{com}} \cdot r_{\text{pose}} \quad (4)$$



(a) Test reward during training  
(b) Torque limits during training

**Figure 3: a) Average test reward curve for the strong-to-weak curriculum and different values of the fixed torque limits. b) Value of the torque limits in training with a curriculum. The results are averaged over 10 runs.**

We also include additional variables in the state space to facilitate the training. In addition to  $s_{\text{weak}}$ , we select several attributes at two future steps from the reference trajectory  $\tau_{\text{slow}}$  to augment the state space. The selected attributes are the local joint rotations  $q$ , the center of mass height  $h_{\text{com}}$  and the vertical projection of the torso orientation vector  $o_{\text{torso}}$ . The concatenation of those attributes forms a vector  $\hat{s}'_t = [q_t, h_{\text{com}}, o_{\text{torso}}]$ . The state space at timestep  $t$  can be eventually expressed as:  $s_{\text{slow}} = [s_{\text{fast}}, \hat{s}'_{t+1}, \hat{s}'_{t+5}]$ . During the get-up phase, the future pose is provided by the retimed slow trajectory  $\tau_{\text{slow}}$  at the specified timestep, which is later replaced with the standing pose  $\hat{q}$  for the balancing stage. This setup informs the policy of the short-term and long-term goals at the same time.

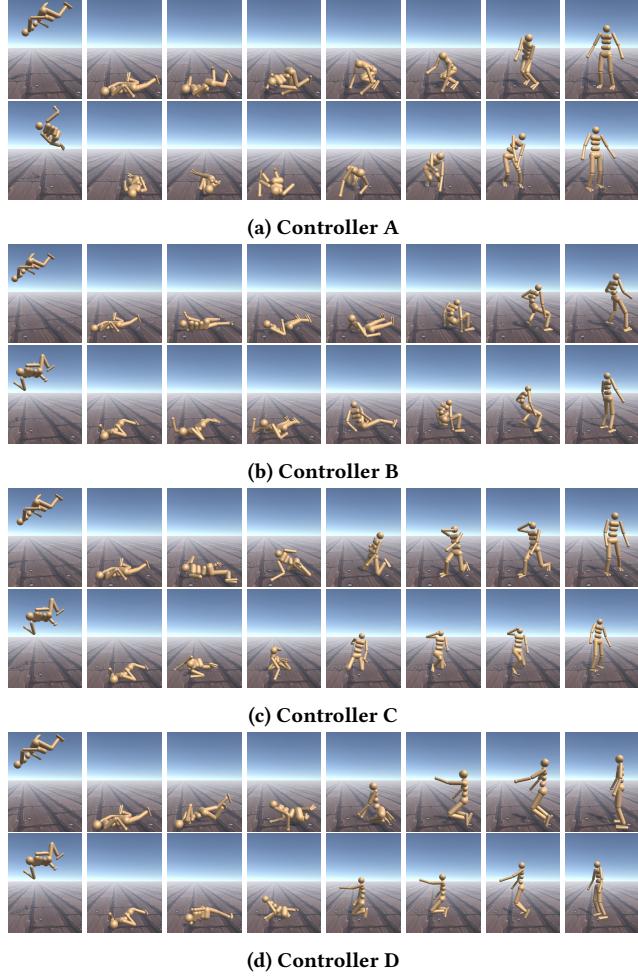
**Get-up Variants.** Besides the standard humanoid character, we also explore and study the generalization ability of our framework on some variants of the humanoid character. We learn get-up motions for a character with a leg and an arm in a cast and a character with a missing arm. To simulate the character with a leg and an arm in a cast, we lock the left elbow joint and the right knee joint to keep those limbs straight throughout the motion. We make no further modifications to the environment and algorithm design except for removing corresponding joint information from the state space and control signal from the action space.

## 8 RESULTS

We first verify the hypothesis that large torque limits are essential to exploration. Low torque limits can prevent the discovery of get-up solutions due to limited exploration. We demonstrate that gradually reducing the torque limits with a curriculum can learn a low-energy solution mode. Then, we show that learning slow get-up motions can further improve the naturalness of the motion. Also, we exploit the future pose conditioned policy  $\pi_{\text{slow}}$  to pause the get-up motions in selected statically-stable poses. Finally, we provide visualization tools to analyse the behavior of each controller and the diversity of the learnt solution modes. We refer readers to the supplementary videos for a clear demonstration of the resulting motions.

### 8.1 Strong-to-weak Curriculum

We demonstrate the learning curves for both fixed torque limits and strong-to-weak curriculum in Fig. 3a and the value of torque



**Figure 4: Get-up motions of four controllers.** Each 8 figures in a row show the get-up motion in one episode from the rag-doll fall to the standing phase. Two runs starting from supine and prone positions are shown for each controller.



**Figure 5: One get-up controller adopting different strategies starting from the supine and prone positions.**

limits with the curriculum in Fig. 3b. Fig. 3a verifies the importance of proper torque limits for exploring a coarse solution mode. When the torque limit is fixed at 60%, 50% and 40% of the default torque limit  $\mathcal{T}$  respectively, the controller is likely to fail to discover any get-up solution mode. By employing the strong-to-weak curriculum, the agent first learns a solution mode and then refines the motion

while adapting to the decreasing torque limits. As shown in Fig. 3b, the final torque limit with curriculum drops to below 60% of the default torque limit  $\mathcal{T}$  on average. The strong-to-weak curriculum achieves comparable learning speed as the full strength model but produces a more natural get-up motion.

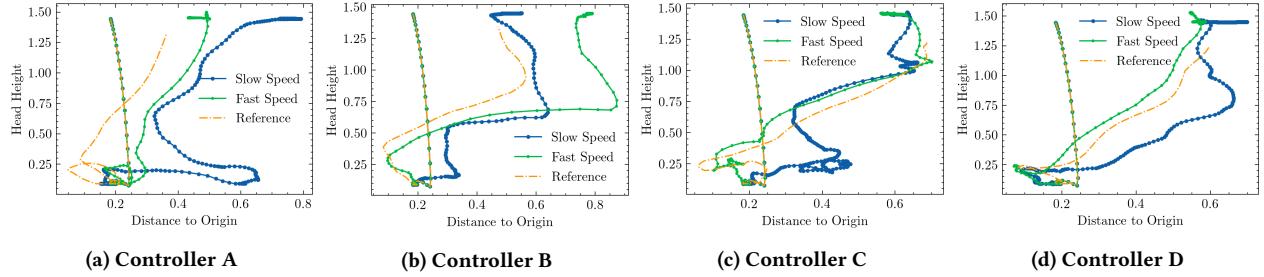
The strong-to-weak curriculum is terminated according to the rule described in Sec. 5. We find that the DRL policy tends to first adopt a fixed solution mode first and then refines it. Launching experiments with different seeding functions usually yields diverse get-up styles. Therefore, those get-up motions will end up with different final torque limits. As a subjective observation, we find the final torque limits to be correlated to the naturalness of the get-up motions. In our experiments, get-up strategies ending with torque limits ranging from 40% to 60% are usually perceived as more natural than those terminating with high torque limits (above 70%). We include the get-up motions of several  $\pi_{\text{weak}}$  in the supplementary video.

## 8.2 Slow Get-up Motion

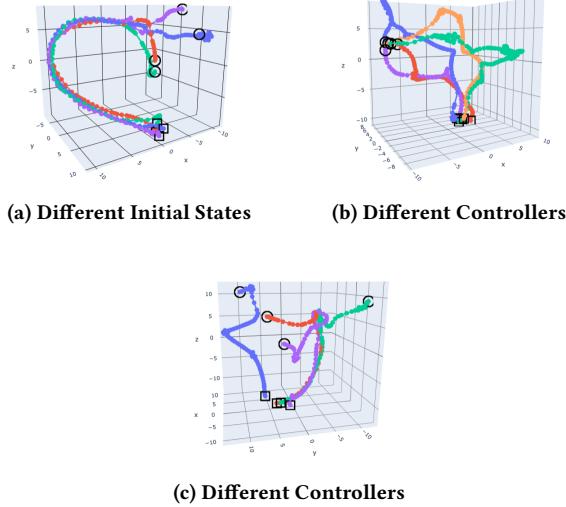
We next show results at medium speed ( $\kappa = 0.5$ ), from the initial rag-doll fall to getting up from the ground and finally remaining to standing. Fig. 4 shows four different controllers labelled as A, B, C and D starting from two initial states. Each controller either prefers to get up from a supine position or the prone position. If a controller prefers to get up in a supine position but starts from a prone position, the character commonly first rolls over, then gets up, and vice versa. However, exceptions also exist that attempt to get up from supine and prone positions using different strategies, as shown in Fig. 5.

Our controller can produce get-up motions with different speeds by adjusting the retiming coefficient  $\kappa$ . To analyse the behavior of controllers running at different speeds, we plot the trajectory of the head in the lateral space. Fig. 6 plots the head height versus the distance to the origin projected to the  $xy$  plane for each controller we showed in Fig. 4. The slow and fast trajectories share a similar path with the reference trajectory but make their adaptations to accomplish the tasks, which indicates the necessity to learn a physics-based controller  $\pi_{\text{slow}}$  rather than merely being an identical-but-slower copy of the reference motion  $\tau_{\text{fast}}$ .

To better understand the structure and diversity of the learned get-up strategies, we project features of the state to a 3D space by t-SNE, and generate plots of the trajectory in 3D space. More details regarding the t-SNE implementation are included in App. G. Fig. 7a shows the get-up trajectories for a given controller for four different initial states. These begin at different points in the embedded space and then merge to a single trunk because the given controller tends to adopt the same strategy to get up, and eventually arrive at the region representing the standing pose. Fig. 7b reveals the differences across multiple controllers starting from an identical initial state. The projected trajectories begin with the same point after the rag-doll stage, then diverge to different paths to get up from the ground, and finally merge to the standing pose. Fig. 7c shows the t-SNE trajectory plot for the controller adopting different strategies in supine and prone positions. Trajectories starting from supine and prone positions take different paths to the standing region in the embedded space.



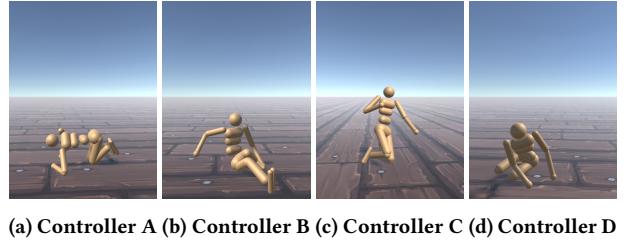
**Figure 6:** Head trajectories in the lateral space of the character. Each plot includes the head trajectory in the lateral view for slow get-up, fast get-up and reference motion. There are noticeable differences between the three trajectories. We choose  $\kappa = 0.25, 0.75$  for the slow and fast trajectories respectively. Each marker in the path represents one control timestep.



**Figure 7:** t-SNE plots of trajectories. Fig. a) shows four trajectories from the same controller starting from four different initial states. Fig. b) shows five trajectories from different controllers starting from the same initial state. Fig. c) shows four trajectories from the controller using different strategies when starting from supine and prone positions. The starting states are circled while the final states are squared.

### 8.3 Paused Get-up Motion

As the slow get-up policy  $\pi_{\text{slow}}$  is conditioned on two future poses, we can manipulate the reference trajectory  $\tau_{\text{fast}}$  in various ways other than uniform retiming. One idea is to repeat one specific state  $\hat{s}'$  in the reference trajectory multiple times such that the policy aims to reach the repeated pose  $\hat{q}$  first, then maintains the pose for a while, and finally continues the rest of the get-up motion. This setting creates a get-up motion paused at the repeated state. Without a future pose conditioned policy, such paused motion is nearly impossible to achieve with a purely state-indexed policy. As a result, our motion can be paused and continued in many statically stable states as shown in Fig. 8, although the character loses balance



**Figure 8:** Paused get-up states. Each picture shows one state that the get-up motion can be paused at.

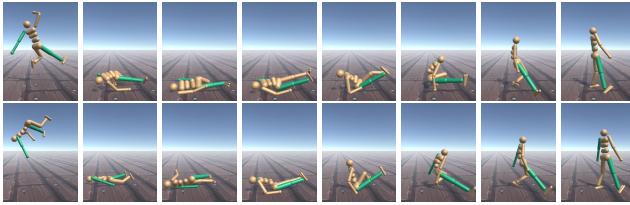
when asked to pause in more dynamical states. We include the get-up motions with pauses in the supplementary video. In general, we find that the generated get-up motions are usually more statically stable at the beginning and become dynamic and less stable near the end of the get-up.

### 8.4 Get-up Motion for Humanoid Variants

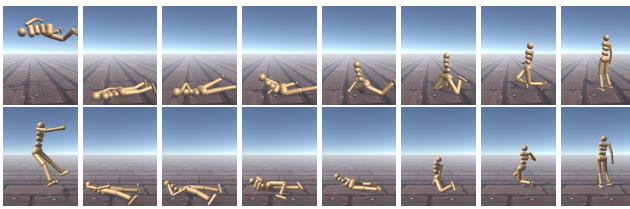
Following the same pipeline, we can generate get-up motions adapted to a humanoid character with a leg and an arm in a cast and a humanoid character with a missing arm. Fig. 9 and Fig. 10 illustrate the discovered get-up motions for those special characters respectively. The resulting motion is also demonstrated in the supplementary video. We show that these irregular humanoid models can still get up at various commanded speeds. Since two joints are removed for the character with limbs in casts, it adopts a get-up strategy that relies on the remaining limbs to gain momentum, while using the limbs in casts for balance at certain stages. The policy developed with the character with a missing arm attempts to get up by pushing against the ground using one arm only. These results show that our pipeline is not restricted to a specific model but can be applied to situations where motion capture data is hard to obtain.

### 9 ABLATION STUDIES

We conduct multiple ablation studies to investigate the role of several components in our system, and explore alternative methods to learn weak and slow get-up motions via reward engineering. We experiment with removing the strong-to-weak curriculum and slow get-up imitation respectively. As a result, we generally observe that the resulting motions are of lower quality. We note that careful



**Figure 9: Get-up motions for humanoid with a leg and an arm in a cast.** The left elbow joint and right knee joint are locked throughout the motion. The corresponding limbs are rendered in green. Each row of images shows the get-up motion from either the supine position or the prone position.



**Figure 10: Get-up motions for humanoid with missing left arm.** Each row of images shows the get-up motion from either the supine position or the prone position.

reward design itself is not sufficient to learn a weak and slow get-up motion. Relevant videos are included in the supplementary material.

**High Strength Get-up Motion.** We first illustrate that training with high-strength characters tends to find an unnatural and highly-dynamic get-up motion. We train the initial policy  $\pi_{\text{strong}}$  without any modification on the torque limit. The results are best observed in the supplemental video. The strong-to-weak curriculum eliminates excessively aggressive and abrupt motions.

**Weak Motions Without a Curriculum.** As discussed earlier (§8.1), starting the training with fixed low torque limits typically traps the policy in local minima and fails to find any suitable solution mode.

**Slow Get-up Imitation Without Curriculum.** We also evaluate an ablation where we skip the strong-to-weak curriculum and proceed directly to imitating retimed versions of a strong policy  $\pi_{\text{strong}}$ . We find that  $\pi_{\text{strong}}$  provides excessively-dynamic get-up motions for  $\pi_{\text{slow}}$  to imitate. As a result, the character fails to get up at low speeds. Although  $\pi_{\text{slow}}$  sometimes succeeds in fast get-up tasks, the motion remains overly dynamic and awkward.

**Weak and Slow Get-up Motion Alternatives.** Instead of an explicit imitation objective, motion constraints can often be embedded in the reward function design. Adding an energy cost term has been proposed in [Fu et al. 2021; Ma et al. 2021; Xie et al. 2020] to improve motion quality. However, we find that adding an energy cost without the strong-to-weak curriculum has minimal effect on the get-up motion. In addition, we experiment with adding a reward term to penalize high joint velocities for learning slow get-up motions. However, such a reward design either has negligible effects on the get-up speeds with fewer weights on this term or leads to training instability with more weights on it. We further test the option of annealing the weights of the energy cost term and the

velocity regularization term linearly. Nevertheless, the resulting motions are not clearly more statically stable and slowed down. We include the corresponding motions in the supplementary video.

## 10 CONCLUSION

We have presented a framework based on deep reinforcement learning to produce natural human get-up from the ground motions without recourse to motion capture data. The final learned policy can yield realistic get-up motions at different speeds and from arbitrary initial states. We first exploit the benefits of a high-strength character to discover a particular get-up strategy. The initial policy is then refined with a progressively weaker character to enhance motion quality. Lastly, our method learns an imitation controller to get up at much slower speeds, including pausing in intermediate statically-stable states. We visualize the diversity across different controllers and the behavior from different initial states.

Our method has a variety of remaining limitations, pointing to directions for future work. Currently our method still requires a separate simulation using  $\pi_{\text{weak}}$  in order to generate the reference trajectory that is used to condition  $\pi_{\text{slow}}$ . It should be possible to learn a single policy that is directly conditioned on the current state and  $\kappa$ , mainly via a distillation. This would eliminate the need to store and use  $\pi_{\text{weak}}$ . We leave this as future work.

Our learning framework can discover, in a tabula rasa fashion, diverse get-up motions, across different runs with different randomized policy initializations. However, there is currently no means to provide user control. We wish to explore various possible methods for adding control over the choice of get-up strategy, and more general control over the style. One interesting strategy would be to learn a set of  $N$  controllers, and then have a user specify their preference for the desired get-up strategy employed from different initial states, and to then reintegrate this into a single controller.

Humans need to get up from chairs, sofas, bathtubs, car seats, variable terrain, and a variety of other constrained situations, e.g., getting up while wearing skates or skis. Humans are extremely adept at finding good solutions to these problems. An exciting direction for future work will be to produce controllers that can generalize well to this broad range of circumstances.

## ACKNOWLEDGMENTS

We thank Sheldon Andrews, Anthony Frezzato, and Arsh Tangri for many useful discussions about the getup problem.

## REFERENCES

- Farzad Abdolhosseini. 2019. *Learning locomotion: symmetry and torque limit considerations*. Ph.D. Dissertation. University of British Columbia. <https://doi.org/10.14288/1.0383251>
- Farzad Abdolhosseini, Hung Yu Ling, Zhaoming Xie, Xue Bin Peng, and Michiel van de Panne. 2019. On learning symmetric locomotion. In *Motion, Interaction and Games*. 1–10.
- Jacqueline MG Adams and Sarah Tyson. 2000. The Effectiveness of Physiotherapy to Enable an Elderly Person to Get up from the Floor: A single case study. *Physiotherapy* 86, 4 (2000), 185–189.
- Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG)* 38, 6 (2019), 1–11.
- Mazen Al Borno, Michiel Van De Panne, and Eugene Fiume. 2017. Domain of attraction expansion for physics-based character control. *ACM Transactions on Graphics (TOG)* 36, 2 (2017), 1–11.

- Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th annual international conference on motion, interaction, and games*. 1–10.
- Alexander Clegg, Zackory Erickson, Patrick Grady, Greg Turk, Charles C Kemp, and C Karen Liu. 2020. Learning to collaborate from simulation for robot-assisted dressing. *IEEE Robotics and Automation Letters* 5, 2 (2020), 2746–2753.
- Stelian Coros, Philippe Beaudoin, and Michiel Van de Panne. 2010. Generalized biped walking control. *ACM Transactions On Graphics (TOG)* 29, 4 (2010), 1–9.
- Zipeng Fu, Ashish Kumar, Jitendra Malik, and Deepak Pathak. 2021. Minimizing energy consumption leads to the emergence of gaits in legged robots. *arXiv preprint arXiv:2111.01674* (2021).
- Thomas Geijtenbeek and Nicolas Pronost. 2012. Interactive character animation using simulated physics: A state-of-the-art review. In *Computer graphics forum*, Vol. 31. Wiley Online Library, 2492–2515.
- Martin Grimmer. 2015. *Powered Lower Limb Prostheses*. Ph.D. Dissertation. <https://doi.org/10.13140/RG.2.2.15449.47200>
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.
- Perttu Hämäläinen, Sebastian Eriksson, Esa Tanskanen, Ville Kyrki, and Jaakko Lehtinen. 2014. Online motion synthesis using sequential monte carlo. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–12.
- Perttu Hämäläinen, Joose Rajamäki, and C Karen Liu. 2015. Online control of simulated humanoids using particle belief propagation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–13.
- Félix G Harvey and Christopher Pal. 2018. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*. 1–4.
- Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 60–1.
- Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Esfandiari, et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017).
- Jessica K Hodgins, Wayne L Wooten, David C Brogan, and James F O'Brien. 1995. Animating human athletics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 71–78.
- Ludovic Hoyet, Lucas Mourou, François Le Clerc, François Schnitzler, and Pierre Hellier. 2021. A Survey on Deep Learning for Skeleton-Based Human Animation. In *Computer Graphics Forum*.
- Heejin Jeong and Daniel D Lee. 2016. Efficient learning of stand-up motion for humanoid robots with bilateral symmetry. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1544–1549.
- Yifeng Jiang, Tom Van Wouwe, Friedl De Groot, and C Karen Liu. 2019. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions On Graphics (TOG)* 38, 4 (2019), 1–12.
- Fumio Kanehiro, Kiyoshi Fujiwara, Hirohisa Hirukawa, Shin'ichiro Nakaoka, and Mitsuharu Morisawa. 2007. Getting up motion planning using mahalanobis distance. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2540–2545.
- Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Shuuji Kajita, Kazuhito Yokoi, Hirohisa Hirukawa, Kazuhiko Akachi, and Takakatsu Itozumi. 2003. The first humanoid robot that has the same size as a human and that can lie down and get up. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, Vol. 2. IEEE, 1633–1639.
- Nam Hee Kim, Hung Yu Ling, Zhaoming Xie, and Michiel van de Panne. 2021. Flexible Motion Optimization with Modulated Assistive Forces. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4, 3 (2021), 1–25.
- Seunghwan Lee, Moonseok Park, Kyounghyun Lee, and Jehee Lee. 2019. Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)* 38, 4 (2019), 1–13.
- Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. In *ACM SIGGRAPH 2010 papers*. 1–8.
- Yoonsang Lee, Moon Seok Park, Taesoo Kwon, and Jehee Lee. 2014. Locomotion control for many-muscle humanoids. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–11.
- Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 40–1.
- Libin Liu, Michiel Van De Panne, and KangKang Yin. 2016. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)* 35, 3 (2016), 1–14.
- Libin Liu, KangKang Yin, and Baining Guo. 2015. Improving sampling-based motion control. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 415–423.
- Libin Liu, KangKang Yin, Michiel Van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. In *ACM SIGGRAPH 2010 papers*. 1–10.
- Li-Ke Ma, Zeshi Yang, Xin Tong, Baining Guo, and KangKang Yin. 2021. Learning and Exploring Motor Skills with Spacetime Bounds. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 251–263.
- Julieta Martinez, Michael J Black, and Javier Romero. 2017. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2891–2900.
- Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. 2018. Hierarchical visuomotor control of humanoids. *arXiv preprint arXiv:1811.09656* (2018).
- Josh Merel, Yuval Tassa, Dhruva TB, Srinivasan Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. 2017. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201* (2017).
- Igor Mordatch, Jack M Wang, Emanuel Todorov, and Vladlen Koltun. 2013. Animating human lower limbs using contact-invariant optimization. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–8.
- Jun Morimoto and Kenji Dooya. 1998. Reinforcement learning of dynamic motor sequence: Learning to stand up. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, Vol. 3. IEEE, 1721–1726.
- Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning Predict-and-Simulate Policies From Unorganized Human Motion Data. *ACM Trans. Graph.* 38, 6, Article 205 (2019).
- Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.
- Xue Bin Peng and Michiel van de Panne. 2017. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 1–13.
- Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolínek, and Georg Martius. 2020. Sample-efficient Cross-Entropy Method for Real-time Planning. *arXiv preprint arXiv:2008.06389* (2020).
- Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Sripathi Sridhar, and Leonidas J Guibas. 2021. HuMoR: 3D Human Motion Model for Robust Pose Estimation. *arXiv preprint arXiv:2105.04668* (2021).
- Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Trans. Graph.* 38, 6 (2019), 209–1.
- Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 54–1.
- Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4906–4913.
- Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. 2020. dm\_control: Software and Tasks for Continuous Control. *arXiv:2006.12983 [cs.RO]*
- Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033.
- Kevin Wampler, Zoran Popović, and Jovan Popović. 2014. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.
- Jack M Wang, David J Fleet, and Aaron Hertzmann. 2009. Optimizing walking controllers. In *ACM SIGGRAPH Asia 2009 papers*. 1–8.
- Jack M Wang, David J Fleet, and Aaron Hertzmann. 2010. Optimizing walking controllers for uncertain inputs and environments. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 1–8.
- Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 33–1.
- Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. 2020. ALL-STEPS: Curriculum-driven Learning of Stepping Stone Skills. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 213–224.
- Yuting Ye and C Karen Liu. 2010. Optimal feedback control for character animation using an abstract model. In *ACM SIGGRAPH 2010 papers*. 1–9.
- KangKang Yin, Kevin Loken, and Michiel Van de Panne. 2007. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 105–es.
- Zhiqi Yin, Zeshi Yang, Michiel Van De Panne, and KangKang Yin. 2021. Discovering diverse athletic jumping strategies. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–17.
- YouTube. 2013. *The Stand Up Challenge: 52 Ways to Get Up*. <https://www.youtube.com/watch?v=1Af1DtnTLS8>
- YouTube. 2019. *No Hands Get Ups – 7 Get up Variations*. <https://www.youtube.com/watch?v=h376pQ6uFt4>
- Wenhao Yu, Greg Turk, and C Karen Liu. 2018. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.
- He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.

## Supplemental Material

### A PSEUDOCODE

We provide the pseudocode of the Discover and the Weaker stages in Algorithm 1, and that of the Slower stage in Algorithm 2.

**ALGORITHM 1:** Discover and Weaker Stages

---

```

Input: Torque limit multiplier  $\beta$ , Minimum steps  $N_{min}$ , Maximum
steps  $N_{max}$ , Policy  $\pi$ , Threshold  $\omega$ 
// Initialize parameters;
i  $\leftarrow 0$ ;
N  $\leftarrow 0$ ;
Nc  $\leftarrow 0$ ;
 $\hat{\beta} \leftarrow 1.0$ ;
Initialize initial state to s;
for step = 1, ..., max_steps do
    a  $\sim \pi(s, \hat{\beta})$ ;
    (s', R, done) = ForwardSimulation(s, a);
    Save (s, a, R, s', done) in the replay buffer D;
    s  $\leftarrow s'$ ;
    Perform SAC update using data from D;
    Nc = Nc + 1;
    if step % test_frequency == 0 then
        Rtest = TestPolicy( $\pi$ );
        // Advance curriculum;
        if Rtest >  $\omega$  then
            i = i + 1;
            // Update termination criteria;
            N = Clip(Nc, N_min, N_max);
            Nc = 0;
        else
            // Terminate curriculum;
            if Nc > N then
                End Training;
            end
        end
    if done then
        // Sample the torque limit multiplier;
         $\hat{\beta} \sim N(\beta^i, \epsilon)$ ;
        s = ResetEpisode();
    end
end

```

---

### B DETAILED REWARD DESIGN

The overall reward function  $R(s_t, a_t)$  is the product of multiple reward terms  $r$  between 0 and 1. As shown in Figure. 11, each reward term  $r$  is calculated by a function of input value  $i$  defined by three parameters bounds  $b$ , margin  $m$  and value  $v$  as  $f(i, b, m, v)$ . Bounds  $b = [b_l, b_u]$  defines the region where the reward term is 1 if the input value  $i$  is inside. The reward value will drop smoothly outside the bounds following a Gaussian curve until reaching value  $v$  at a distance of margin  $m$  [Tassa et al. 2020].

---

### ALGORITHM 2: Slower Stages

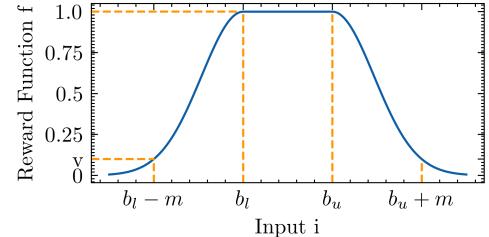
---

```

Input: Weak Policy  $\pi_{weak}$ , Low speed limit  $k_{low}$ , High speed limit
 $k_{high}$ 
done  $\leftarrow True$ ;
for step = 1, ..., max_steps do
    if done then
        s = ResetEpisode();
         $\tau_{fast} = \text{GenerateTrajectory}(s, \pi_{weak})$ ;
         $k \sim \mathcal{U}(k_{low}, k_{high})$ ;
         $\tau_{slow} = \text{Retime}(\tau_{fast}, k)$ ;
    end
    a  $\sim \pi_{slow}(s, \tau_{slow})$ ;
    (s', R, done) = ForwardSimulation(s, a);
    Save (s, a, R, s', done) in the replay buffer D;
    s  $\leftarrow s'$ ;
    Perform SAC update to  $\pi_{slow}$  using data from D;
end

```

---



**Figure 11: Reward Function  $f(i, b, n, v)$ .** The general reward function is bounded in  $[0, 1]$  and specified by three parameters, bounds  $b = [b_l, b_u]$ , margin  $m$  and ending value  $v$ .

### B.1 Exploring Initial Policies and its Low-energy Variants.

One direct indication of the get-up behavior is the head height. Therefore, the agent will be rewarded 1 once the head height is above a threshold. As the character gets up from the ground, the head height reward grows up to 1 until reaching the threshold. We set the reward function as follows:

$$r_h = f(i = h_{\text{head}}, b = [1.55, \text{inf}], m = 0.37, v = 0.1). \quad (5)$$

Additionally, the character is also encouraged to keep the torso vertically straight when getting up. We add a reward based on the vertical projection of the unit torso up vector, i.e.  $z_{\text{torso}}^{\text{up}}$ . The reward is only applied when the center of mass height  $h_{\text{com}}$  is above 0.5m as:

$$r_{\text{straight}} = \begin{cases} f(i = z_{\text{torso}}^{\text{up}}, b = [0.9, \text{inf}], m = 1.9, v = 0.0), & \text{if } h_{\text{com}} > 0.5 \\ 1.0, & \text{otherwise} \end{cases}. \quad (6)$$

The velocity is one way to distinguish a walking or running motion from a get-up motion. To prevent the agent from being rewarded for walking or running, we add a reward term constraining on the center of mass velocity projected on the horizontal

plane  $v_{\text{com}}^{\text{xy}} = [v_{\text{com}}^x, v_{\text{com}}^y]$ :

$$r_{v_{\text{com}}^{\text{xy}}} = \frac{1}{2} \sum_{v' \in v_{\text{com}}^{\text{xy}}} f(i = v', b = [-0.3, 0.3], m = 1.2, v = 0.1), \quad (7)$$

where we set the bounds parameter  $b$  small to allow slow roll over motion.

We also observe that get-up motion with further split feet is perceived unnatural for humans. Thus, we constrain the distance between the feet by adding a penalty term. The penalty term encourages the feet to keep the euclidean feet distance projected to the  $xy$  plane  $d_{\text{feet}}$  no more than roughly twice the shoulder width:

$$r_{\text{feet}} = f(i = d_{\text{feet}}, b = [0, 0.9], m = 0.38, v = 0). \quad (8)$$

## B.2 Slow Get-up Motions

For clarity and compactness, we denote the attributes in the reference trajectory  $\tau_{\text{slow}}$  with an apostrophe as  $(\cdot)'$ .

We first compare the center of mass height between the character and reference trajectory  $\tau_{\text{slow}}$  at the current timestep by computing the distance  $\Delta h_{\text{com}} = h_{\text{com}} - h'_{\text{com}}$ . The reward term can be formally expressed as:

$$r_{\text{com}} = f(i = \Delta h_{\text{com}}, b = [0, 0], m = 0.5, v = 0.1). \quad (9)$$

Similarly, we also enforce the torso orientation vectors projected to the vertical axis between the character and reference trajectory to match. Therefore, we include a reward term on the vertical axis projection of the torso orientation vector  $o_{\text{torso}} = [x_{\text{torso}}^{\text{up}}, y_{\text{torso}}^{\text{up}}, z_{\text{torso}}^{\text{up}}]$  to the Cartesian coordinate:

$$r_{\text{ori}} = \prod_{o \in \Delta o} f(i = o, b = [-0.03, 0.03], m = 0.6, v = 0.3), \quad (10)$$

where  $\Delta o$  represents the difference between the orientation vectors, i.e.,  $\Delta o = o - o'$ .

In addition, we notice that the tracking controller might exhibit unnatural swift stepping motion to gain balance during the get-up motion. This behavior is mainly caused by fast manipulating the hip joint motor along the  $y$  axis. To mitigate this issue, we add another regularization term on the angular joint velocities on both hip joints  $v_{\text{hip}} = [v_{\text{hip}}^1, v_{\text{hip}}^2]$  to match the corresponding value in the reference trajectory:

$$r_{\text{hip}} = \frac{1}{2} \sum_{\Delta v \in v_{\text{hip}} - v'_{\text{hip}}} f(i = \Delta v, b = [-0.5, 0.5], m = 1.3, b = 0.1) \quad (11)$$

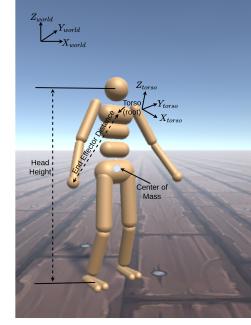
## B.3 Standing

We slightly modify the center of mass velocity term  $r_{v_{\text{com}}^{\text{xy}}}$  by setting the upper and lower bounds to 0 as  $b = [0.0, 0.0]$  to penalize any movement. Besides, we add a pose tracking reward  $r_{\text{pose}}$  on the local joint rotations to minimize the distance between the current pose and the designed standing pose  $\hat{q}$ :

$$r_{\text{pose}} = \exp \left[ -\frac{1}{4} \sum_{j=0}^J \|q_j - \hat{q}_j\|^2 \right], \quad (12)$$

where  $J$  is the number of free joints, and  $q_j$  represents the rotation angles in radians.

## C DETAILED EXPLANATION OF STATE VARIABLES



**Figure 12: Diagram indicating the variables in the state definition.**

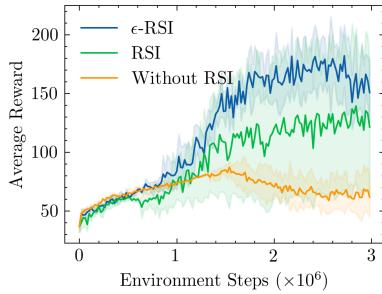
As illustrated in Fig. 12, the head height is measured from the head link to the ground along the vertical direction. The end-effector positions are computed as the translational offset between the end-effectors and the root link in egocentric coordinates. The state also involves a variable indicating the straightness of the torso  $[x_{\text{torso}}^{\text{up}}, y_{\text{torso}}^{\text{up}}, z_{\text{torso}}^{\text{up}}]$  as the z-component of the torso orientation vectors  $[X_{\text{torso}}, Y_{\text{torso}}, Z_{\text{torso}}]$ .

## D IMPLEMENTATION DETAILS

All the training tasks are simulated using Mujoco engine [Todorov et al. 2012] running at 800 Hz while all the control policies run at 40 Hz. Our character is roughly 1.5 m tall and weighs 38.3 kg. The character has a total of 19 body parts and 21 DoFs in total. The default joint angle limits and torque limits  $\mathcal{T}$  are listed in Tab. 2 in App. F.

To train the controllers, we apply soft actor critic (SAC) [Haarnoja et al. 2018] with fully connected neural networks and ReLU activation function. Controller  $\pi_{\text{weak}}$  adopts torque as the action space while we use a low-level PD controller running at 800 Hz to compute the torque for controller  $\pi_{\text{slow}}$ . PD controller has been proved to outperform torque as actuation model for imitation tasks [Peng and van de Panne 2017]. For the weak and slow get-up policies, they are trained on a 32-core CPU desktop with Nvidia Geforce RTX 2070 GPU for roughly 24 hours each. During training, policies are evaluated across ten random initial states. The ten randomly sampled initial states are fixed throughout the training.

For initial policy learning and strong-to-weak curriculum, the trained policy is evaluated every 20000 simulation steps over 10 episodes. The policy is updated through SAC every simulation step after the first 10000 simulation steps. We choose  $\beta = 0.95$  and  $\omega = 60$  throughout all the experiments. We set the clipping boundary for  $M_i$  as  $N_{\min} = 3 \times 10^5$  and  $N_{\max} = 8 \times 10^5$ . For learning slower get-up motions, we choose the interpolation coefficient to be randomly sampled from  $[0.2, 0.8]$  ( $\kappa_{\text{low}} = 0.2$ ,  $\kappa_{\text{high}} = 0.8$ ) range for all the motions. In terms of the PD-controller, we set the gain to be the default joint limits  $k_p = \mathcal{T}$ , and the damping coefficient as  $k_d = \frac{1}{10} k_p$ . Our SAC implementation adopts the



**Figure 13: Training curve comparing different initialization strategies. We compare three strategies:  $\epsilon$ -RSI, RSI and without RSI.**

temperature annealing technique to adapt to different scales of reward functions. Both the actor and critic models are two-layer fully connected neural networks with 1024 hidden units. Tab. 1 show the hyperparameters of the SAC algorithm that we used for all the experiments.

**Table 1: Hyperparameters used for training the SAC algorithm.**

Hyperparameter	Value
Critic Learning Rate	$10^{-4}$
Actor Learning Rate	$10^{-5}$
Initial Temperature $\alpha$	0.1
$\alpha$ Learning Rate	$10^{-4}$
Optimizer	Adam
Log of Policy Standard Deviation Min	-5
Log of Policy Standard Deviation Max	2
Target Update Rate ( $\tau$ )	$5 \cdot 10^{-3}$
Batch Size	1024
Iterations per time step	1
Discount Factor	0.97 for $\pi_{\text{weak}}$ 0.95 for $\pi_{\text{slow}}$
Reward Scaling	1.0
Gradient Clipping	False

## E EPISODE INITIALIZATION STRATEGY COMPARISON

In Fig. 13, we illustrate the performance of three initialization strategies:  $\epsilon$ -RSI, RSI and without RSI. Here, without RSI refers to always starting the episode from the beginning of the reference motion. Our experiment suggests that  $\epsilon$ -RSI outperforms other options by a large margin. Without RSI, initializing from the beginning of the reference motion can lead to instability in training. Regular RSI can successfully imitate the reference motion, but still does not perform as well as  $\epsilon$ -RSI.

## F HUMANOID CHARACTER CONFIGURATION

In Table 2, we list the joint torque limits and joint angle limits of the humanoid character in this work.

**Table 2: Joint Configuration of the Humanoid Model**

Joint Name	Torque Limit $\mathcal{T}$ (Nm)	Angle Limit (rad)	Rotation Axis
<i>abdomen<sub>z</sub></i>	40	[−0.79, 0.79]	[0, 0, 1]
<i>abdomen<sub>y</sub></i>	40	[−1.31, 0.52]	[0, 1, 0]
<i>abdomen<sub>x</sub></i>	40	[−0.61, 0.61]	[1, 0, 0]
<i>right hip<sub>x</sub></i>	40	[−0.44, 0.09]	[1, 0, 0]
<i>right hip<sub>z</sub></i>	40	[−0.52, 0.44]	[0, 0, 1]
<i>right hip<sub>y</sub></i>	120	[−1.92, 0.35]	[0, 1, 0]
<i>right knee</i>	80	[−2.79, 0.03]	[0, −1, 0]
<i>right ankle<sub>y</sub></i>	20	[−0.35, 0.79]	[0, 1, 0]
<i>right ankle<sub>x</sub></i>	20	[−0.87, 0.87]	[1, 0, 0.5]
<i>left hip<sub>x</sub></i>	40	[−0.44, 0.09]	[−1, 0, 0]
<i>left hip<sub>z</sub></i>	40	[−0.52, 0.44]	[0, 0, −1]
<i>left hip<sub>y</sub></i>	120	[−1.92, 0.35]	[0, 1, 0]
<i>left knee</i>	80	[−2.79, 0.04]	[0, −1, 0]
<i>left ankle<sub>y</sub></i>	20	[−0.35, 0.79]	[0, 1, 0]
<i>left ankle<sub>x</sub></i>	20	[−0.87, 0.87]	[1, 0, 0.5]
<i>right shoulder1</i>	20	[−1.48, 1.05]	[2, 1, 1]
<i>right shoulder2</i>	20	[−1.48, 1.05]	[0, −1, 1]
<i>right elbow</i>	40	[−1.57, 0.87]	[0, −1, 1]
<i>left shoulder1</i>	20	[−1.05, 1.48]	[2, −1, 1]
<i>left shoulder2</i>	20	[−1.05, 1.48]	[0, 1, 1]
<i>left elbow</i>	40	[−1.57, 0.87]	[0, −1, −1]

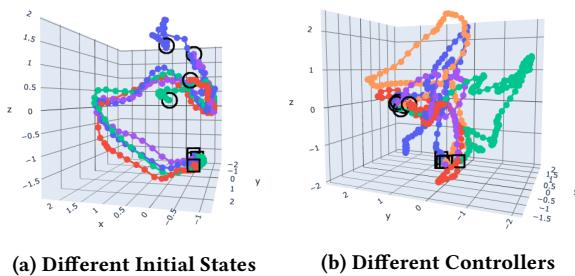
Our humanoid character is largely simplified if compared with the human body. Some important components are missed in the current design, including fingers and toes. Fingers and toes are vital in the get-up motion because they contact the ground directly to provide momentum to get up. In addition, the current character is modeled with simple cylinders and capsules, which is far from the shape of the real human body. We believe that careful modeling of the humanoid character can lead to more realistic motions.

## G DETAILS ON TSNE PLOTS OF TRAJECTORIES

Since the state variable has high dimensionality, we select the more representative features from the state variable to perform the t-SNE analysis. The selected features are ankle rotations, knee rotations, hip rotations, head height, and the vertical projection of the torso up vector  $z_{\text{torso}}^{\text{up}}$ . For each trajectory, we remove the states representing the rag-doll fall and most of the standing part. The rag-doll fall states are identical given the same initial states, which can mess up the nearest neighbour calculation for t-SNE. The same reason applies to the removal of most of the standing states. We choose the *perplexity* parameter of t-SNE to be 10 for Fig. 7a and Fig. 7b, and 20 for Fig. 7c.

## H ALTERNATIVE TRAJECTORY VISUALIZATION

We choose t-SNE algorithm to compress the high dimension state to a 3D vector for visualization purpose. Alternatively, Principal Component Analysis (PCA) serves as another popular dimensionality reduction algorithm to encode the original state variables through a linear transformation. As shown in Fig. 14, we experiment the choice of PCA on the same data used in the visualization demonstrated in Fig. 7. The plots generated by PCA still provide reasonable trajectories. However, the converging behavior of the



**Figure 14: PCA plots of trajectories.** These two diagrams correspond to the same data but different dimensionality reduction algorithm of Fig. 7a and Fig. 7b.

same controller is not as clearly illustrated as with t-SNE. Additionally, different trajectories produced by different controllers spread out, and are more difficult to interpret.