Learning to Get Up with Deep Reinforcement Learning

by

Tianxin Tao

B.A.Sc., University of British Columbia, 2019

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL
STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

April 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Learning to Get Up with Deep Reinforcement Learning

submitted by **Tianxin Tao** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Examining Committee:

Michiel van de Panne, Professor, Department of Computer Science, UBC
Supervisor

Dinesh K. Pai, Professor, Department of Computer Science, UBC
Supervisory Committee Member

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 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.

Lay Summary

We develop a method that enables physics-based human simulations to learn how to get up from an arbitrary fallen state on flat ground. This is accomplished using reinforcement learning, i.e., through iterative trial-and-error improvements. Unlike some approaches, our framework do not require motion capture data. We also provide visualization tools to analyze the behavior of the learnt controllers.

Preface

Chapter 4, 5 and 6 are primarily adapted from a paper currently under review. I developed most of the key ideas, implemented the proposed multi-stage learning pipelines, and conducted experiments to evaluate its effectiveness and the necessity of various components. My supervisor, Dr. Michiel van de Panne, has offered suggestions and guidance on this project and contributed to the paper writing. Matthew Wilson proposed and implemented the visualization tools to better understand the learnt motion controllers. Ruiyu Gou applied and adapted the tools for visualization results in this thesis and the related paper submission.

Table of Contents

Abstract . . . . . **iii**

Lay Summary . . . . . **iv**

Preface . . . . . **v**

Table of Contents . . . . . **vi**

List of Tables . . . . . **viii**

List of Figures . . . . . **ix**

Glossary . . . . . **xi**

Acknowledgments . . . . . **xii**

1 Introduction **1**

2 Related Work **3**

2.1 Character Animation 3

2.2 Learning Get-up Motions 4

2.3 Human Get-up Motions 5

3 Preliminaries **6**

3.1 Soft Actor-Critic 6

4 Methods **8**

4.1 System Overview 8
4.2 'Discover' and 'Weaker' Stages 10
4.3 'Slower' Stage 12

5 Experiment Setup and Task Specification 13
5.1 Exploring Initial Policies and its Low-torque Variants. 16
5.2 Slow Get-up Motions. 18
5.3 Get-up Variants. 20

6 Results 21
6.1 Strong-to-weak Curriculum 21
6.2 Slow Get-up Motion 22
6.3 Paused Get-up Motion 27
6.4 Get-up Motion for Humanoid with a Cast 27

7 Ablation Studies 29

8 Conclusion and Future Work 31
8.1 Policy Distillation 31
8.2 Get-up Style Control 32
8.3 Adaption to Diverse Environments 32
8.4 Optimal Reward Discovery 32

Bibliography 34

List of Tables

Table 5.1 Joint Configuration of the Humanoid Model 14
Table 5.2 Hyperparameters used for training the SAC algorithm. 15

List of Figures

Figure 1.1 Teaser Figure. This shows the get-up motion starting from rag-doll fall to the final standing phase. 2

Figure 4.1 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. . . . 9

Figure 5.1 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 v and ending value v. 16

Figure 6.1 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. 22

Figure 6.2 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. 23

Figure 6.3 One get-up controller adopting different strategies starting from the supine and prone positions. 24

Figure 6.4 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. 24
Figure 6.5 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 trajectories from the controller using different strategies when starting from supine and prone positions. Three initial states on the right are in supine positions while the left one is in prone positions. The starting states are circled while the final states are squared. . . 25
Figure 6.6 Paused get-up states. Each picture shows one state that the get-up motion can be paused at. 27
Figure 6.7 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. 28

Glossary

DRL deep reinforcement learning

FSM finite state machines

LSTM Long Short-Term Memory

MDP Markov Decision Process

MLP multilayer perceptrons

PD proportional-derivative

RSI random state initialization

SAC Soft Actor-Critic

T-SNE t-distributed stochastic neighbor embedding

VAE Variational Autoencoders

Acknowledgments

I would like to thank all the people who assisted me in accomplishing all the achievements during my master's study. I am sincerely grateful to my supervisor, Dr. Michiel van de Panne, who has mentored and guided me effortlessly over the time we worked together. I am very fortunate to have you as my supervisor. You are always willing to offer help and inspiration whenever I need it. I cannot reach where I am today without your earnest care and encouragement. Especially during the dark pandemic period, your words truly motivated me towards the right direction and calmed me down when I was stuck in my research.

I also want to express my gratitude to my wife, Ruolan Ye, who shares my concerns over the past years, supports me under any condition and takes care of my personal life. In addition, I would like to say thank you to my family who raised me and supported me in my life. Especially, I will forever remember the sacrifice of my mother to support me both financially and mentally, which is impossible to evaluate its value.

Finally, I would like to thank my lab mates and friends, Daniele Reda, Matthew Wilson, Ruiyu Gou, Hung Yu Ling and Dr. Zhaoming Xie, for their inspirational discussion and technical help.

Chapter 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 [46, 52]. 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 common 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 $5\times$ slower than the original speed.

Our method generates a diverse range of get-up styles across multiple runs. Each run has a different randomized policy initialization. The discovered motions 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.

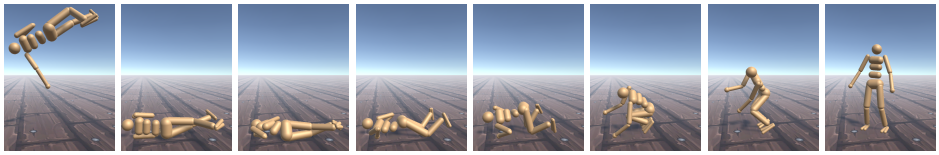


Figure 1.1: Teaser Figure. This shows the get-up motion starting from ragdoll fall to the final standing phase.

Chapter 2

Related Work

We provide an overview of physics-based and kinematics-based character animation, learning get-up controllers in animation and for robots, and lastly, we discuss the diverse strategies utilized by humans to get up.

2.1 Character Animation

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 [11]. Early work demonstrates significant success on locomotion tasks and often relies on compact manually-designed feedback rules and finite state machines (FSM), e.g., [8, 21, 55, 56, 61]. Value iteration methods have been explored to learn specific locomotion skills with physics-based characters [7, 44]. Trajectory optimization has also been used with considerable success to generate locomotion control for both human and non-human characters [27, 40, 54]. Various motion tracking controllers have been proposed to imitate available motion capture data using model-based approaches [30, 31, 60, 61] or sampling-based methods [15, 16, 33–35].

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. [19] 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 [2, 65]. Yin et al. [62] 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 [6, 59]. Alternative actuation models with muscle models and controlled via muscle activations have also been proposed to learn more human-like motions [24, 29, 43]. Additionally, DRL is widely studied to learn tracking controllers for reference motions. Peng et al. [45] proposed a DRL-based framework with random state initialization (RSI) and early termination to learn controllers capable of imitating highly diverse motions. 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 [42]. A hybrid action space of torque and proportional-derivative (PD) control is proposed to accelerate the training process [5]. Instead of manually designing the reward, GAIL [20] learns the reward function in an adversarial setting.The methods have also recently been made much more scalable, e.g., [57].

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) [32, 47], Long Short-Term Memory (LSTM) [17, 18, 37] and mixture of experts [49, 50, 66]. We refer readers to a survey paper for a more detailed overview on this topic [22].

2.2 Learning Get-up Motions

Learning a get-up controller has been of interest to computer animation and robotics. Pioneering work by Morimoto et al. [41] proposed a hierarchical reinforcement learning framework to master the get-up motion on a simplified 2D walker model. Kanehiro et al. [25, 26] 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 [23]. A wide range of sampling techniques have been used to successfully

discover get-up motions, e.g., [15, 46]. Despite their success, these sampling methods commonly suffer from limited motion quality and are less suited for online use 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., [5, 38]. Online trajectory optimization methods, i.e., model predictive control, are also capable of get-up motions for humanoid [51], 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 [4]. This uses large torque limits (300 Nm) and relies on local PD-control feedback for stability, rather than closed-loop full state feedback. In contrast to prior work, we propose a framework which does not need a reference motion, achieves fast runtime performance, and that can produce slow-and-weak motions that are more representative of most human get-up motions.

2.3 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 [63], including a variety of ways to get up without the intermediate use of hands [64]. A variety of the methods described require a significant 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., [3]. 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.

Chapter 3

Preliminaries

We formulate the DRL problem as a standard Markov Decision Process (MDP). MDP can be defined by states $s_t \in \mathscr{S}$, actions $a_t \in \mathscr{A}$, 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 $\pi_\theta(s_t)$ paramertized by θ 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 value, $R(s_t, a_t)$, is returned as the reward function at each timestep. The training objective of DRL is to maximize the expected return, defined as:

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right] \tag{3.1}$$

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

3.1 Soft Actor-Critic

We choose Soft Actor-Critic (SAC) [13] 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_θ, 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 $Q^\pi_\phi(s_t, a_t)$ as critic, which are commonly represented by multilayer perceptrons (MLP).

The parameters in the critic ϕ are updated by minimizing the squared Bellman error given the transition tuples $\tau_t = (s_t, a_t, s_{t+1}, r_t)$ stored in the replay buffer D:

$$\begin{aligned} \mathscr{L}(\phi) &= \mathbb{E}_{\tau\sim D}[Q_\phi(s_t, a_t) - (r_t + y(r_t, s_{t+1}))]^2, \\ y(r_t, s_{t+1}) &= \gamma(\min_{i=1,2} Q'_{\phi_i}(s_{t+1}, a') - \alpha \log_{\pi_\theta}(a'|s_{t+1})) \end{aligned} \tag{3.2}$$

The parameter α is the temperature value to balance the two terms, which is treated as a learned parameter as in [14]. Q' is a slowly updated copy of the critic to improve training stability. The actor is learned by maximizing the weighted objective of the expected return and the policy entropy as:

$$\mathscr{L}(\theta) = \mathbb{E}_{a\sim\pi}[Q_\phi(s_t, a) - \alpha \log_{\pi_\theta}(a|s_t)]. \tag{3.3}$$

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.

Chapter 4

Methods

4.1 System Overview

We illustrate our learning pipeline in Fig. 4.1. We split the overall training process into three sequential stages: (1) initial policy exploration with strong characters; (2) low-effort motion discovery through a strong-to-weak curriculum; and (3) slow motion refinement via motion imitation. Discovering get-up motion 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 π_{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 a high reward region, and therefore 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 π_{weak} suitable for much weaker versions of the character. In practice, we combine the training of the initial policy π_{strong} and its adaption to low-strength actions into one training process. Once the test reward of π_{strong} reaches a threshold ω, the strong-to-weak curriculum is automatically activated.

After the strong-to-weak curriculum, we obtain a state-indexed physics-based controller π_{weak} generating get-up motions not requiring large joint torques. To

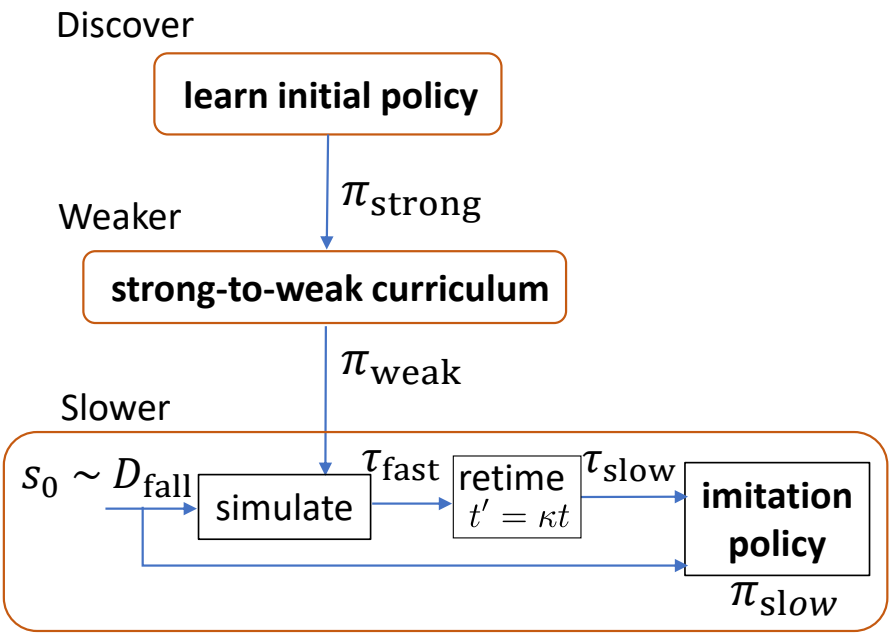


Figure 4.1: System overview. Our system explores an initial policy with a strong character, then refines the motion from a strong character to a weak character. Finally, we train an imitation policy to track the retimed trajectory produced by the weak policy.

achieve slower movements, we introduce a motion tracking objective for a second controller π_{slow} to imitate the retimed trajectories τ_{slow}. Given an initial state s_0, we first generate a fast get-up trajectory τ_{fast} using policy π_{weak}. This is then retimed by a factor of κ, $\kappa \in [0, 1]$. The newly trained controller, π_{slow}, can produce motions up to $5\times$ slower. At run time, users can specify the value of κ to adjust the speed of get-up motions. Moreover, we also train the controller π_{slow} to maintain balance once 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.

4.2 '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 [39, 45] and degraded learning performance [1]. Therefore, we design the default torque limits $\mathscr{T}$ according to documented values for humans [12], and scale them with respect to the weights of body parts.

We start the training with the designed torque limits $\mathscr{T}$ of the humanoid character to explore an initial policy π_{strong}. Then, the initial policy π_{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 ω, we advance the torque limit curriculum by setting the torque limits to be $\beta^i \times \mathscr{T}$ at the ith 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 ith stage of the curriculum π_{weak} to output actions bounded by $[-\beta^i, \beta^i]$ by modifying the squashing function to $\beta^i \times \tanh(.)$. 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 have different meanings.

At the ith stage of the strong-to-weak curriculum, we sample the torque limit multiplier β^i from a Gaussian distribution $\mathscr{N}(\beta^i, \varepsilon), \varepsilon = 0.04$ 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 [59, 62], 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 $\mathscr{M}$, which indicates that the current task is too dif-

ficult under the torque limit constraints. Rather than a constant, the threshold $\mathscr{M}$ grows as curriculum advances since discovering very low-energy get-up motions becomes more challenging with lower torque limits. We associate the threshold to the number of steps taken at the last stage of the curriculum N_{i-1}, and define the threshold at stage i, $\mathscr{M}_i$ according to $\mathscr{M}_i = clip(1.5 \times N_{i-1}, N_{min}, N_{max})$, where N_{min} and N_{max} are hyperparameters defining minimum and maximum steps for all the curriculum stages. After the end is triggered, the torque limit is reset to the torque limit at the previous stage of the curriculum as β^{i-1}, and the training continues until convergence.

4.3 '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 π_{weak}. Specifically, on every episode reset to an initial state s_0, we iteratively query the get-up policy π_{weak} to interact with the environment and generate a state trajectory $\tau_{fast} = (s_0, s_1, \ldots, s_T)$ of length T. During training, we sample a constant κ uniformly between κ_{low} and κ_{high} ($0 < \kappa_{low} \leq \kappa_{high} < 1$) as the retiming coefficient for slow get-up trajectories τ_{slow}. For retiming, we use linear interpolation on the state trajectory over $[0, T]$.

To accelerate training, random state initialization (RSI) [45] 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 ε-RSI. ε is a scalar between $[0, 1]$. With probability 1-ε, RSI is adopted; otherwise, the episode is started from the beginning. At training time, ε-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 start to end. 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 π_{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 π_{slow} outputs the a residual value q_r added to the reference orientation q' supplied by the reference trajectory τ_{slow}: $q = q_r + q'$. The user can adjust the speed of the get-up motion by controlling the value of the retiming coefficient κ.

After the get-up motion, the most common and natural succeeding movement is that of a quiescent stance. In support of this, we also train the controller π_{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.

Chapter 5

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. 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 [52] 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.

All the training tasks are simulated using Mujoco engine [53] 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. We document the joint torque limits and joint angle limits of the humanoid character in this work in Table 5.1.

To train the controllers, we apply soft actor critic (SAC) [13] with fully connected neural networks and ReLU activation function. We list the hyperparameters of the SAC algorithm in Table 5.2 we used for all the experiments. Controller π_{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 π_{slow}. PD controllers have been shown to outperform torque as actuation model for imitation tasks [43]. For the weak and slow get-up policies, they are trained on a 32-core CPU desktop with Nvidia

Table 5.1: Joint Configuration of the Humanoid Model

Joint Name	Torque Limit $\mathscr{T}$ (Nm)	Angle Limit (rad)	Rotation Axis
$abdomen_z$	40	$[-0.79, 0.79]$	$[0,0,1]$
$abdomen_y$	40	$[-1.31, 0.52]$	$[0,1,0]$
$abdomen_x$	40	$[-0.61, 0.61]$	$[1,0,0]$
$right\ hip_x$	40	$[-0.44, 0.09]$	$[1,0,0]$
$right\ hip_z$	40	$[-0.52, 0.44]$	$[0,0,1]$
$right\ hip_y$	120	$[-1.92, 0.35]$	$[0,1,0]$
$right\ knee$	80	$[-2.79, 0.03]$	$[0,-1,0]$
$right\ ankle_y$	20	$[-0.35, 0.79]$	$[0,1,0]$
$right\ ankle_x$	20	$[-0.87, 0.87]$	$[1,0,0.5]$
$left\ hip_x$	40	$[-0.44, 0.09]$	$[-1,0,0]$
$left\ hip_z$	40	$[-0.52, 0.44]$	$[0,0,-1]$
$left\ hip_y$	120	$[-1.92, 0.35]$	$[0,1,0]$
$left\ knee$	80	$[-2.79, 0.04]$	$[0,-1,0]$
$left\ ankle_y$	20	$[-0.35, 0.79]$	$[0,1,0]$
$left\ ankle_x$	20	$[-0.87, 0.87]$	$[1,0,0.5]$
$right\ shoulder1$	20	$[-1.48, 1.05]$	$[2,1,1]$
$right\ shoulder2$	20	$[-1.48, 1.05]$	$[0,-1,1]$
$right\ elbow$	40	$[-1.57, 0.87]$	$[0,-1,1]$
$left\ shoulder1$	20	$[-1.05, 1.48]$	$[2,-1,1]$
$left\ shoulder2$	20	$[-1.05, 1.48]$	$[0,1,1]$
$left\ elbow$	40	$[-1.57, 0.87]$	$[0,-1,-1]$

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 held 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 $\mathscr{M}_i$ 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 $[0.2, 0.8](\kappa_{low} = 0.2, \kappa_{high} = 0.8)$ range for all the motions. We set the probability of starting from the beginning of the trajectory to 0.2, i.e., $\varepsilon = 0.2$. In terms of

Table 5.2: Hyperparameters used for training the SAC algorithm.

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

the PD-controller, we set the gains to be the default joint limits $k_p = \mathscr{T}$, and the damping coefficient as $k_d = \frac{1}{10}k_p$. Our SAC implementation adopts the 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. Table 5.2 show the hypermeters of the SAC algorithm that we used for all the experiments.

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 $a \sim \mathscr{N}(0, 0.1)$, 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.

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. 5.1, 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 out-

side the bounds following a Gaussian curve until reaching value v at a distance of margin m [52].

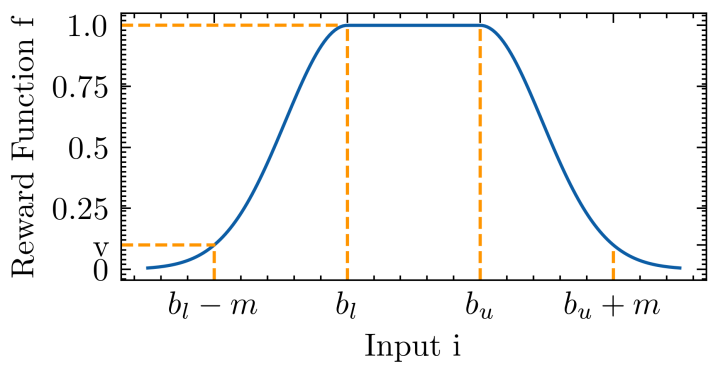


Figure 5.1: Reward Function $f(i,b,n,v)$. The general reward function is bounded in [0, 1] and specified by three parameters, bounds $b = [b_l, b_u]$, margin v and ending value v.

5.1 Exploring Initial Policies and its Low-torque 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_{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_{torso}, (5) the character strength parameter at i^{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. The action space is normalized to $[-\beta^i, \beta^i]$, which is scaled by the default torque limits $\mathscr{T}$ 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) $r_{straight}$: keep the torso vertically straight, (3) $r_{v_{com}^{xy}}$: prevent the character from walking or running, (4) r_{feet}: constrain the distance between two feet. Each

reward term is defined by function f illustrated in Fig. 5.1. The final reward for the get-up task can be summarized as:

$$R_{weak} = r_h \cdot r_{straight} \cdot r_{v^{xy}_{com}} \cdot r_{feet}. \tag{5.1}$$

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:

$$r_h = f(i = h_{head}, b = [1.55, \infty], m = 0.37, v = 0.1). \tag{5.2}$$

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^{up}_{torso}. The reward is only applied when the center of mass height h_{com} is above $0.5m$ as:

$$r_{straight} = \begin{cases} f(i = z^{up}_{torso}, b = [0.9, \infty], m = 1.9, \quad v = 0.0), & \\ & \text{if } h_{com} > 0.5 \\ 1.0, & \text{otherwise} \end{cases}. \tag{5.3}$$

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 $v^{xy}_{com} = [v^x_{com}, v^y_{com}]$:

$$r_{v^{xy}_{com}} = \frac{1}{2} \sum_{v' \in v^{xy}_{com}} f(i = v', b = [-0.3, 0.3], m = 1.2, v = 0.1), \tag{5.4}$$

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_{feet} no more than roughly twice the shoulder width:

$$r_{feet} = f(i = d_{feet}, b = [0, 0.9], m = 0.38, v = 0). \tag{5.5}$$

5.2 Slow Get-up Motions.

We aim to train a get-up controller receiving different speed commands by specifying the retiming coefficient κ. We train a motion tracking controller π_{slow} with DRL by imitating the retimed trajectory τ_{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 π_{weak}, another environment that tracks the retimed slow trajectories τ_{slow}. At the beginning of each episode, we obtain a fast get-up trajectory τ_{fast} until the head height h_{head} is above $1.2m$, which is later retimed to a slower trajectory τ_{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 κ as $\frac{T}{\kappa}$. In addition, the episode will be terminated as long as the center of mass height deviates from the reference by $0.5m$. For clarity and compactness, we denote the attributes in the reference trajectory τ_{slow} with an apostrophe as $(.)'$.

We design the imitation reward R_{slow} consisting of the following elements: (1) r_{com}: track the center of mass height, (2) r_{ori}: track the torso orientation vectors projected to the vertical axis, (3) r_{hip}: track the hip joint velocity to avoid shaking behavior. Each reward term is defined by function f illustrated in Fig. 5.1. Thus, the final reward for the imitation phase can be expressed as:

$$R_{slow} = r_{com} \cdot r_{ori} \cdot \frac{r_{hip} + 2}{3} \tag{5.6}$$

We first compare the center of mass height between the character and reference trajectory τ_{slow} at the current timestep by computing the distance $\Delta h_{com} = h_{com} -$

h'_{com}. The reward term can be formally expressed as:

$$r_{com} = f(i = \Delta h_{com}, b = [0,0], m = 0.5, v = 0.1). \tag{5.7}$$

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 $o_{torso} = [x^{up}_{torso}, y^{up}_{torso}, z^{up}_{torso}]$ to the Cartesian coordinate:

$$r_{ori} = \prod_{o \in \Delta o} f(i = o, b = [-0.03, 0.03], m = 0.6, v = 0.3), \tag{5.8}$$

where Δo 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 movement of 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 $v_{hip} = [v^l_{hip}, v^r_{hip}]$ to match the corresponding value in the reference trajectory:

$$r_{hip} = \frac{1}{2} \sum_{\Delta v \in v_{hip} - v'_{hip}} f(i = \Delta v, b = [-0.5, 0.5], m = 1.3, b = 0.1) \tag{5.9}$$

Furthermore, we train the standing task to maintain balance concurrently with the same policy π_{slow} by switching to a new reward function $R_{balance}$ 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 condition is met if the center of mass height is below $0.5m$. We manually designed a single standing pose $\hat{q}$ for the imitation policy π_{slow} to mimic. The designed standing pose $\hat{q}$ is concatenated to the end of reference trajectory τ_{slow}. The reward function for the balancing task mainly reuses reward terms from previous tasks, including $r_{v^{xy}_{com}}$, $r_{straight}$ and r_{com}. We also implement a new reward term r_{pose} to track the joint rotations of the designed standing pose. We slightly modify the center of mass velocity term $r_{v^{xy}_{com}}$ by setting the upper and lower bounds to 0 as $b = [0.0, 0.0]$ to penalize any movement. Further, we add a pose tracking reward r_{pose} on the local

joint rotations to minimize the distance between the current pose and the designed standing pose $\hat{q}$:

$$r_{pose} = exp\left[-\frac{1}{4}\sum_{j=0}^{J}||q_j - \hat{q}_j||^2\right], \tag{5.10}$$

where J is the number of free joints, and q_j represents the rotation angles in radians. The final reward for the rotation task can be expressed as:

$$R_{balance} = r_{v_{com}^{xy}} \cdot r_{straight} \cdot r_{com} \cdot r_{pose} \tag{5.11}$$

We also include additional variables in the state space to facilitate the training. In addition to s_{weak}, we select several attributes at two future steps from the reference trajectory τ_{slow} to augment the state space. The selected attributes are the local joint rotations q, the center of mass height h_{com} and the vertical projection of the torso orientation vector o_{torso}. The concatenation of those attributes forms a vector $\hat{s}'_t = [q_t, h_{com_t}, o_{torso_t}]$. The state space at timestep t can be eventually expressed as: $s_{slow} = [s_{fast}, \hat{s}'_{t+1}, \hat{s}'_{t+5}]$. During the get-up phase, the future pose is provided by the retimed slow trajectory τ_{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.

5.3 Get-up Variants.

In addition to the standard humanoid character, we also explore and study the generalization ability of our framework on some variants of the humanoid character. We attempt to learn get-up motions for a 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.

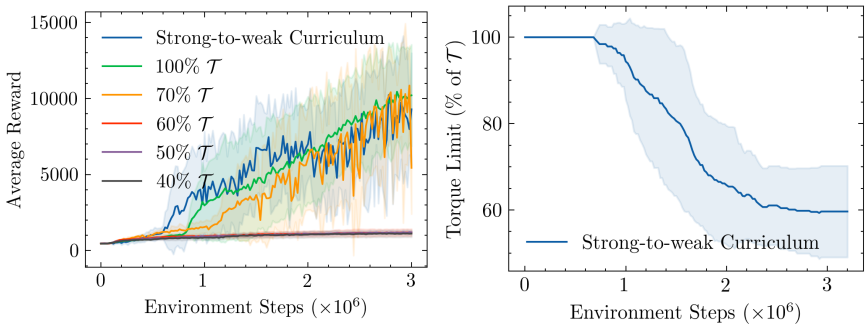
Chapter 6

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 π_{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.

6.1 Strong-to-weak Curriculum

We demonstrate the learning curves for both fixed torque limits and strong-to-weak curriculum in Fig. 6.1a and the value of torque limits with the curriculum in Fig. 6.1b. Fig. 6.1a 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 $\mathscr{T}$ 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. 6.1b, the final torque limit with curriculum can drop



(a) Test reward during training

(b) Torque limits during training

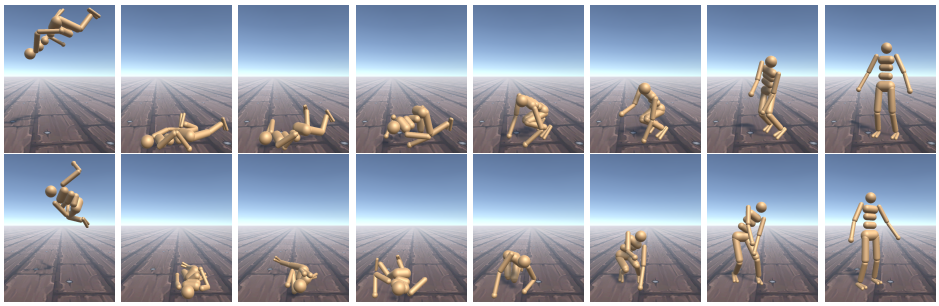
Figure 6.1: a) Average test reward curve for the strong-to-weak curriculum and different values of the fixed torque limits. b) Value of the torque limits in training with a curriculum. The results are averaged over 10 runs.

to below 60% of default torque limit $\mathscr{T}$ on average. The strong-to-weak curriculum achieves comparable learning speed as the full strength model but produces a more natural get-up motion.

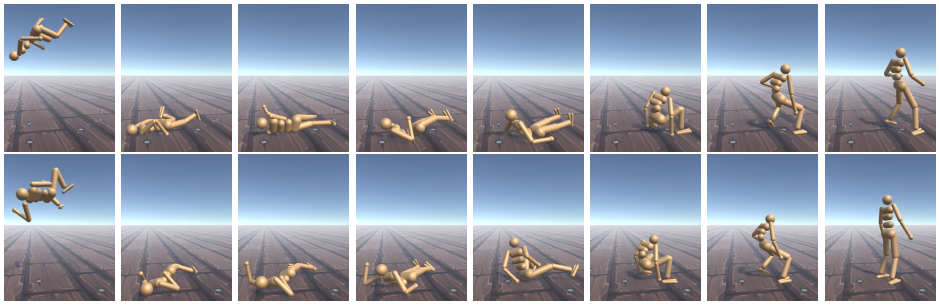
The strong-to-weak curriculum is terminated according to the rule described in Sec. 4.2. We find that the DRL policy tends to first adopt a fixed solution mode first and then refine it. Launching experiments with different seeding functions usually yields diverse get-up styles. Therefore, those get-up 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. We include the get-up motions of several π_{weak} in the supplementary video.

6.2 Slow Get-up Motion

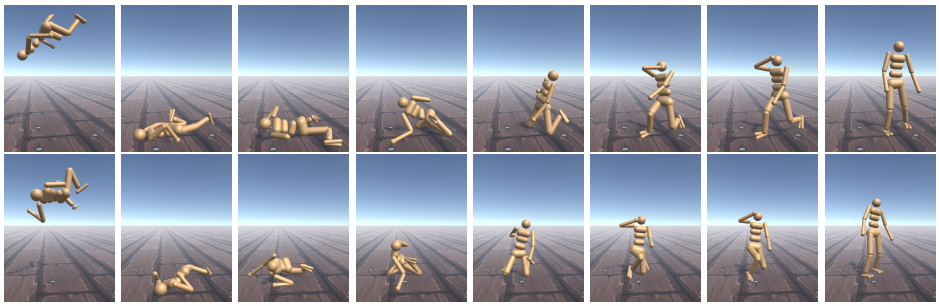
We next show results at medium speed ($\kappa = 0.5$), from the initial rag-doll fall to getting up from the ground and finally remaining to standing. Fig. 6.2 shows four different controllers labelled as A, B, C and D starting from two initial states. Each controller either prefers to get up from the 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,



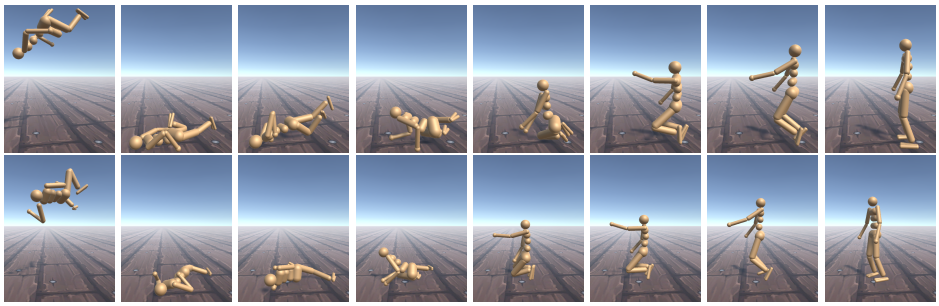
(a) Controller A



(b) Controller B



(c) Controller C



(d) Controller D

Figure 6.2: 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.

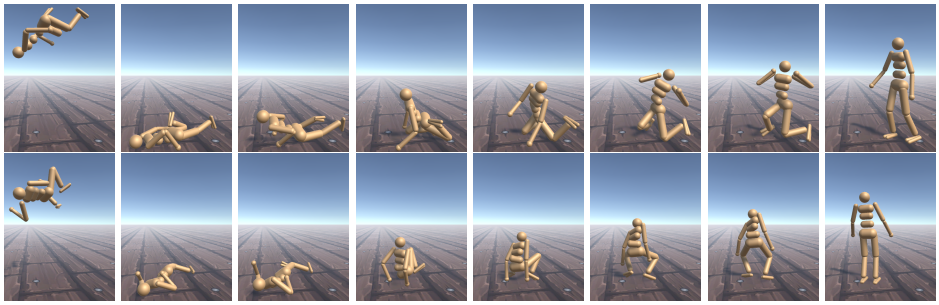
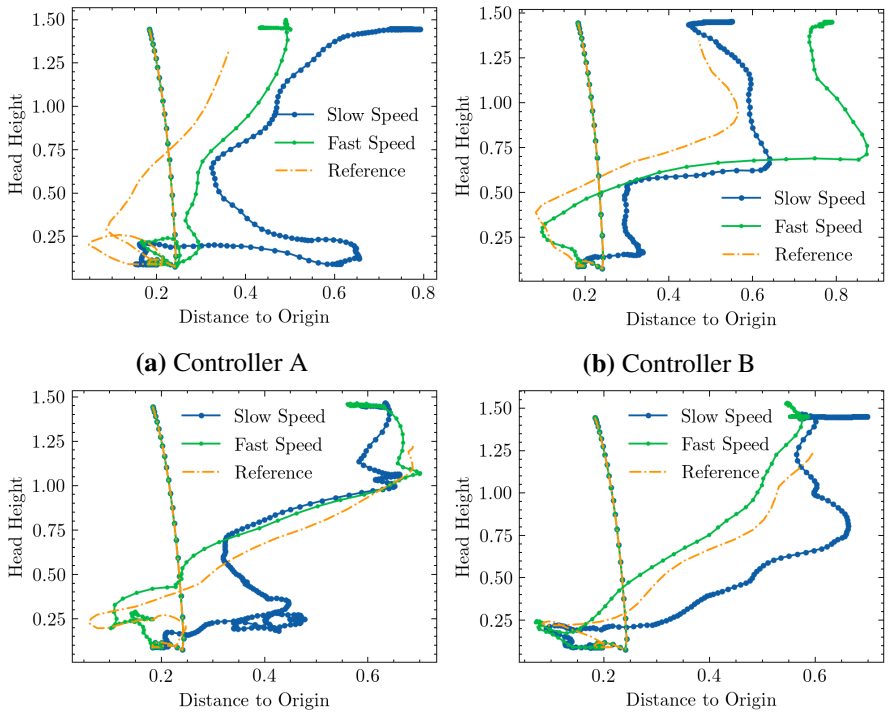


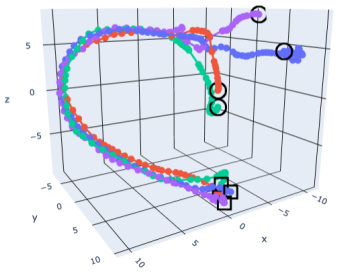
Figure 6.3: One get-up controller adopting different strategies starting from the supine and prone positions.



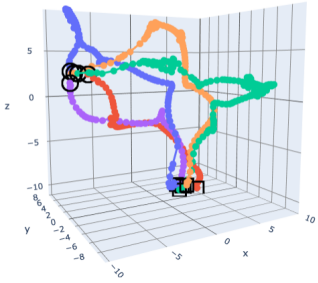
(a) Controller A **(b)** Controller B

(c) Controller C **(d)** Controller D

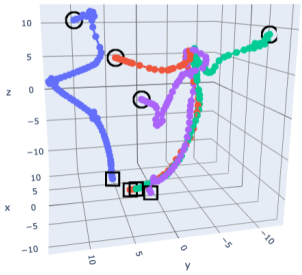
Figure 6.4: Head trajectories in the lateral space of the character. Each plot includes the head trajectory in the lateral view for slow get-up, fast get-up and reference motion. There are noticeable differences between the three trajectories. We choose $\kappa = 0.25, 0.75$ for the slow and fast trajectories respectively. Each marker in the path represents one control timestep.



(a) Different Initial States



(b) Different Controllers



(c) Controllers using Different Strategies

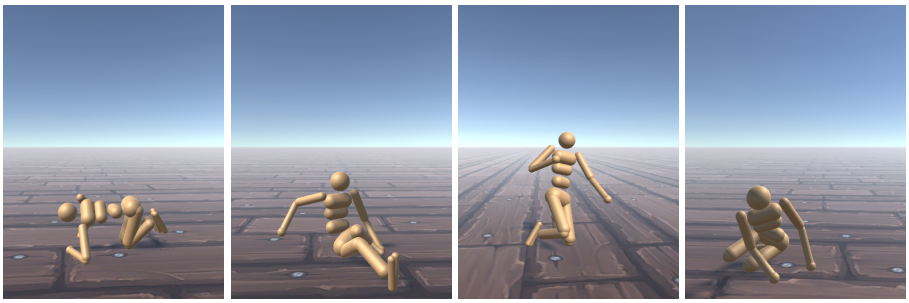
Figure 6.5: t-SNE plots of trajectories. Fig. a) shows four trajectories from the same controller starting from four different initial states. Fig. b) shows five trajectories from different controllers starting from the same initial state. Fig. c) shows trajectories from the controller using different strategies when starting from supine and prone positions. Three initial states on the right are in supine positions while the left one is in prone positions. The starting states are circled while the final states are squared.

exceptions also exist that attempt to get up from supine and prone positions with different strategies, as shown in Fig. 6.5c.

Our controller can produce get-up motions with different speeds by adjusting the retiming coefficient κ. To analyse the behavior of controllers running at different speeds, we plot the trajectory of the head in the lateral space. Fig. 6.4 plots the head height versus the distance to the origin projected to the *xy* plane for each controller we showed in Fig. 6.5c. The slow and fast trajectories share a similar path with the reference trajectory but make their adaptions to accomplish the tasks, which indicates the necessity to learn a physics-based controller π_{slow} rather than simply being an identical-but-slower copy of the reference motion τ_{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. 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^{up}_{torso}. 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. 6.5a and Fig. 6.5b, and 20 for Fig. 6.3.

Fig. 6.5a shows the get-up trajectories for a given controller for four different initial states. These start in different places 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 end up at the region representing the standing pose. Fig. 6.5b 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. 6.3 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.



(a) Controller A **(b)** Controller B **(c)** Controller C **(d)** Controller D

Figure 6.6: Paused get-up states. Each picture shows one state that the get-up motion can be paused at.

6.3 Paused Get-up Motion

As the slow get-up policy π_{slow} is conditioned on two future poses, we can manipulate the reference trajectory τ_{fast} in many interesting ways instead of uniform retiming. One idea is to repeat one specific state s' in the reference trajectory by multiple times such that the policy aims to reach the repeated pose $\tilde{q}$ first, then maintain the pose for a while, and finally continue 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 those more statically stable states as shown in Fig. 6.6, but the character often loses balance when asked to pause in those 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.

6.4 Get-up Motion for Humanoid with a Cast

Following the same pipeline, we can generate get-up motions adapted to humanoid characters with a leg and an arm in a cast. Fig. 6.7 illustrates the discovered get-up motions for this special character. The resulting motion is also demonstrated in the supplementary video. We show that this irregular humanoid model can still get up at various commanded speeds. Since two joints are removed for the character,

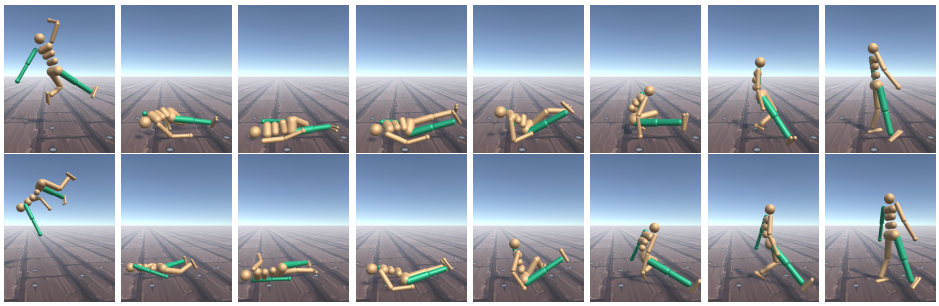


Figure 6.7: Get-up motions for humanoid with a leg and an arm in a cast. The left elbow joint and right knee joint are locked throughout the motion. The corresponding limbs are rendered in green. Each row of images shows the get-up motion from either the supine position or the prone position.

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. 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.

Chapter 7

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 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 π_{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 (§6.1), 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 π_{strong}. We find that π_{strong} provides excessively-dynamic get-up motions for π_{slow} to imitate. As a result, the character fails to get up at low speeds. Although π_{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 [10, 36, 59] 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 R_{slow} 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.

Chapter 8

Conclusion and Future Work

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. We mainly discuss the potential improvement of the current work in four aspects: policy distillation, get-up style control, adaption to diverse environments and optimal reward discovery.

8.1 Policy Distillation

Currently our method still requires a separate simulation using π_{weak} in order to generate the reference trajectory that is used to condition π_{slow}. It should be possible to learn a single policy that is directly conditioned on the current state and κ, mainly via a distillation. This would eliminate the need to store and use π_{weak}. Imitation learning has been shown to be effective for distilling a complex policy

to a simple neural network by supervised learning [58]. However, only providing expert demonstration can cause catastrophic failure when the agent gradually diverges from the expert trajectories. Data augmentation has been proposed to address this issue by querying the expert policy when the trajectory diverges [28, 48].

8.2 Get-up Style Control

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.

8.3 Adaption to Diverse Environments

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. Researchers has investigated to develop motion controller over diverse terrain mainly by incorporating the height field through convolutional filters [19]. Previous kinematics-based animation methods utilizes volumetric sensors and interaction sensors to recognize the surrounding geometry and provide contact information [49].

8.4 Optimal Reward Discovery

Our framework optimizes the reward designed with human heuristics. Discovering the reward function through examples can further automate the learning process with minimal manual design. With a small amount of get-up motion data, we can apply inverse reinforcement learning algorithms to infer the optimal reward func-

tions to learn the policy. Promising inverse reinforcement learning algorithms can potentially adopt the idea of maximum entropy [9] or adversarial optimization [20].

Bibliography

[1] F. Abdolhosseini. *Learning locomotion: symmetry and torque limit considerations*. PhD thesis, University of British Columbia, 2019. URL https://open.library.ubc.ca/collections/ubctheses/24/items/1.0383251. → page 10

[2] F. Abdolhosseini, H. Y. Ling, Z. Xie, X. B. Peng, and M. van de Panne. On learning symmetric locomotion. In *Motion, Interaction and Games*, pages 1–10. 2019. → page 4

[3] J. M. Adams and S. Tyson. The effectiveness of physiotherapy to enable an elderly person to get up from the floor: A single case study. *Physiotherapy*, 86(4):185–189, 2000. → page 5

[4] M. A. Borno, M. V. D. Panne, and E. Fiume. Domain of attraction expansion for physics-based character control. *ACM Transactions on Graphics (TOG)*, 36(2):1–11, 2017. → page 5

[5] N. Chentanez, M. Müller, M. Macklin, V. Makoviychuk, and S. Jeschke. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th annual international conference on motion, interaction, and games*, pages 1–10, 2018. → pages 4, 5

[6] A. Clegg, Z. Erickson, P. Grady, G. Turk, C. C. Kemp, and C. K. Liu. Learning to collaborate from simulation for robot-assisted dressing. *IEEE Robotics and Automation Letters*, 5(2):2746–2753, 2020. → page 4

[7] S. Coros, P. Beaudoin, and M. Van de Panne. Robust task-based control policies for physics-based characters. In *ACM SIGGRAPH Asia 2009 papers*, pages 1–9. 2009. → page 3

[8] S. Coros, P. Beaudoin, and M. Van de Panne. Generalized biped walking control. *ACM Transactions On Graphics (TOG)*, 29(4):1–9, 2010. → page 3

[9] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016. → page 33

[10] Z. Fu, A. Kumar, J. Malik, and D. Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. *arXiv preprint arXiv:2111.01674*, 2021. → page 30

[11] T. Geijtenbeek and N. Pronost. Interactive character animation using simulated physics: A state-of-the-art review. In *Computer graphics forum*, volume 31, pages 2492–2515. Wiley Online Library, 2012. → page 3

[12] M. Grimmer. *Powered Lower Limb Prostheses*. PhD thesis, 02 2015. → page 10

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. 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. → pages 6, 13

[14] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. → page 7

[15] P. Hämäläinen, S. Eriksson, E. Tanskanen, V. Kyrki, and J. Lehtinen. Online motion synthesis using sequential monte carlo. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014. → pages 3, 5

[16] P. Hämäläinen, J. Rajamäki, and C. K. Liu. Online control of simulated humanoids using particle belief propagation. *ACM Transactions on Graphics (TOG)*, 34(4):1–13, 2015. → page 3

[17] F. G. Harvey and C. Pal. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*, pages 1–4. 2018. → page 4

[18] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)*, 39(4):60–1, 2020. → page 4

[19] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017. → pages 3, 32

[20] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016. → pages 4, 33

[21] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 71–78, 1995. → page 3

[22] L. Hoyet, L. Mourot, F. Le Clerc, F. Schnitzler, and P. Hellier. A survey on deep learning for skeleton-based human animation. In *Computer Graphics Forum*, 2021. → page 4

[23] H. Jeong and D. D. Lee. Efficient learning of stand-up motion for humanoid robots with bilateral symmetry. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1544–1549. IEEE, 2016. → page 4

[24] Y. Jiang, T. Van Wouwe, F. De Groote, and C. K. Liu. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions On Graphics (TOG)*, 38(4):1–12, 2019. → page 4

[25] F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, S. Kajita, K. Yokoi, H. Hirukawa, K. Akachi, and T. Isozumi. 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)*, volume 2, pages 1633–1639. IEEE, 2003. → page 4

[26] F. Kanehiro, K. Fujiwara, H. Hirukawa, S. Nakaoka, and M. Morisawa. Getting up motion planning using mahalanobis distance. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2540–2545. IEEE, 2007. → page 4

[27] N. H. Kim, H. Y. Ling, Z. Xie, and M. van de Panne. Flexible motion optimization with modulated assistive forces. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(3):1–25, 2021. → page 3

[28] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020. → page 32

[29] S. Lee, M. Park, K. Lee, and J. Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)*, 38(4):1–13, 2019. → page 4

[30] Y. Lee, S. Kim, and J. Lee. Data-driven biped control. In *ACM SIGGRAPH 2010 papers*, pages 1–8. 2010. → page 3

[31] Y. Lee, M. S. Park, T. Kwon, and J. Lee. Locomotion control for many-muscle humanoids. *ACM Transactions on Graphics (TOG)*, 33(6): 1–11, 2014. → page 3

[32] H. Y. Ling, F. Zinno, G. Cheng, and M. Van De Panne. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)*, 39(4):40–1, 2020. → page 4

[33] L. Liu, K. Yin, M. van de Panne, T. Shao, and W. Xu. Sampling-based contact-rich motion control. In *ACM SIGGRAPH 2010 papers*, pages 1–10. 2010. → page 3

[34] L. Liu, K. Yin, and B. Guo. Improving sampling-based motion control. In *Computer Graphics Forum*, volume 34, pages 415–423. Wiley Online Library, 2015.

[35] L. Liu, M. V. D. Panne, and K. Yin. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)*, 35(3): 1–14, 2016. → page 3

[36] L.-K. Ma, Z. Yang, X. Tong, B. Guo, and K. Yin. Learning and exploring motor skills with spacetime bounds. In *Computer Graphics Forum*, volume 40, pages 251–263. Wiley Online Library, 2021. → page 30

[37] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017. → page 4

[38] J. Merel, Y. Tassa, D. TB, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017. → page 5

[39] J. Merel, A. Ahuja, V. Pham, S. Tunyasuvunakool, S. Liu, D. Tirumala, N. Heess, and G. Wayne. Hierarchical visuomotor control of humanoids. *arXiv preprint arXiv:1811.09656*, 2018. → page 10

[40] I. Mordatch, J. M. Wang, E. Todorov, and V. Koltun. Animating human lower limbs using contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 32(6):1–8, 2013. → page 3

[41] J. Morimoto and K. Doya. 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)*, volume 3, pages 1721–1726. IEEE, 1998. → page 4

[42] S. Park, H. Ryu, S. Lee, S. Lee, and J. Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Trans. Graph.*, 38(6), 2019. → page 4

[43] X. B. Peng and M. van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–13, 2017. → pages 4, 13

[44] X. B. Peng, G. Berseth, and M. Van de Panne. Dynamic terrain traversal skills using reinforcement learning. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015. → page 3

[45] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018. → pages 4, 10, 12

[46] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius. Sample-efficient cross-entropy method for real-time planning. *arXiv preprint arXiv:2008.06389*, 2020. → pages 1, 5

[47] D. Rempe, T. Birdal, A. Hertzmann, J. Yang, S. Sridhar, and L. J. Guibas. Humor: 3d human motion model for robust pose estimation. *arXiv preprint arXiv:2105.04668*, 2021. → page 4

[48] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. → page 32

[49] S. Starke, H. Zhang, T. Komura, and J. Saito. Neural state machine for character-scene interactions. *ACM Trans. Graph.*, 38(6):209–1, 2019. → pages 4, 32

[50] S. Starke, Y. Zhao, T. Komura, and K. Zaman. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)*, 39(4):54–1, 2020. → page 4

[51] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012. → page 5

[52] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess. dm_control: Software and tasks for continuous control, 2020. → pages 1, 13, 16

[53] E. Todorov, T. Erez, and Y. 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. → page 13

[54] K. Wampler, Z. Popović, and J. Popović. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014. → page 3

[55] J. M. Wang, D. J. Fleet, and A. Hertzmann. Optimizing walking controllers. In *ACM SIGGRAPH Asia 2009 papers*, pages 1–8. 2009. → page 3

[56] J. M. Wang, D. J. Fleet, and A. Hertzmann. Optimizing walking controllers for uncertain inputs and environments. *ACM Transactions on Graphics (TOG)*, 29(4):1–8, 2010. → page 3

[57] J. Won, D. Gopinath, and J. Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 39(4):33–1, 2020. → page 4

[58] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246. IEEE, 2018. → page 32

[59] Z. Xie, H. Y. Ling, N. H. Kim, and M. van de Panne. Allsteps: Curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum*, volume 39, pages 213–224. Wiley Online Library, 2020. → pages 4, 10, 30

[60] Y. Ye and C. K. Liu. Optimal feedback control for character animation using an abstract model. In *ACM SIGGRAPH 2010 papers*, pages 1–9. 2010. → page 3

[61] K. Yin, K. Loken, and M. Van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)*, 26(3):105–es, 2007. → page 3

[62] Z. Yin, Z. Yang, M. Van De Panne, and K. Yin. Discovering diverse athletic jumping strategies. *ACM Transactions on Graphics (TOG)*, 40(4):1–17, 2021. → pages 4, 10

[63] YouTube. The stand up challenge: 52 ways to get up, 2013. URL https://www.youtube.com/watch?v=1Af1DtnTLs8. → page 5

[64] YouTube. No hands get ups — 7 get up variations, 2019. URL https://www.youtube.com/watch?v=h376pQ6uFt4. → page 5

[65] W. Yu, G. Turk, and C. K. Liu. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018. → page 4

[66] H. Zhang, S. Starke, T. Komura, and J. Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018. → page 4