Acquiring Stylized Motor Skills for Physics-based Characters

by

Zeshi Yang

B.Sc., University of Science and Technology of China, 2018

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2023

Declaration of Committee

Name:	**Zeshi Yang**
Degree:	**Doctor of Philosophy**
Thesis title:	**Acquiring Stylized Motor Skills for Physics-based Characters**
Committee:	**Chair:** Manolis Savva Assistant Professor, Computing Science

KangKang Yin
Supervisor
Associate Professor, Computing Science

Libin Liu
Committee Member
Assistant Professor, Intelligence Science and Technology
Peking University

Yasutaka Furukawa
Committee Member
Associate Professor, Computing Science

Xue Bin (Jason) Peng
Examiner
Assistant Professor, Computing Science

Paul G. Kry
External Examiner
Associate Professor, Computer Science
McGill University

Abstract

Character animation is a core research topic in computer graphics, which studies synthesizing realistic movements for virtual characters. Physics-based character animation utilizes physics simulation to generate physically plausible character motions. One long-standing goal of physics-based character animation is to equip simulated characters with vast and agile motor skills. Most recent physics-based animation methods learn these agile motion skills by imitating motion capture data or human demonstrations. However, most of these methods mainly focus on motion tracking, and cannot discover novel skills that are visually fundamentally different from references. Therefore, these imitation-based methods cannot be applied to motor skill learning tasks where high-quality motion references are not available. In this thesis, we present several computational methods that enable simulated characters to learn diverse and stylized motor skills in the absence of task-specific motion data.

First, for motion tasks where a limited number of reference motions are available, we present a deep reinforcement learning framework to help simulated characters explore and develop stylized motor skills from reference motions. This system can be used to enrich the variations of the motor skills performed by physics-based characters. Second, for challenging motion tasks where no reference motions are available such as athletic jumping, we design a deep reinforcement learning framework to discover diverse high jumping strategies for simulated characters. Our framework can discover many well-known jumping strategies, like Fosbury flop and Scissor kick, without using task-specific mocap data. Third, we further extend and apply our method developed for full-body high jumping tasks to hand tool manipulation tasks, where we present a learning and control system to enable simulated hands to use chopsticks for object grasping. We demonstrate dexterous object relocation skills with chopsticks in different styles, holding positions and for various hand morphologies. Finally, motivated by the observation that many skill discovery problems can be formulated as hyperparameter optimization problems, we propose a novel multi-fidelity Bayesian Optimization algorithm to optimize hyperparameters of deep reinforcement learning-based animation systems. Our algorithm significantly outperforms state-of-the-art hyperparameter optimization methods applicable for physics-based character animation.

Keywords: Physics-based Character Animation; Deep Reinforcement Learning; Bayesian Optimization

Acknowledgements

First, I would like to express my gratitude to my supervisor: KangKang Yin, who gave me the opportunity to pursue my PhD degree at Simon Fraser University and guided me to do research patiently. Dr.Yin keeps helping me with my research projects even during her busy times. I am thankful to another supervisor of mine: Libin Liu, for his detailed discussions and suggestions on my projects. I can never forget that Dr.Liu did not sleep and helped me polish my papers before the SIGGRAPH deadline. I also want to thank Dr.Xiaoming Fu and Prof.Ligang Liu, who introduced me to the wonderful and exciting world of computer graphics and scientific research.

During my PhD life, I feel so fortunate to have had the opportunity to work in the GrUVi lab and the VCL lab, where I met many friends who mean a lot to me. I want to express my thanks to Yujie Wang, Manyi Li, Zhiqin Chen, Zhiqi Yin, and Li-ke Ma (in no particular order). Thank you for your company during my hard times. I can never get over these hard times without helps from you guys. Special thanks to my academic brother Zhiqin Yin, who cured my anxiety and desperation during the lows of my life and taught me so many valuable lessons.

Finally and most importantly, I would like to thank my father, Youxing Yang, my mother, Xiuhua Deng, and my sister Linghui Wang for their constant care and support throughout my life. Thank you for giving me the freedom to chase my dream. You are always the source of my courage and perseverance.

Table of Contents

List of Tables

List of Figures

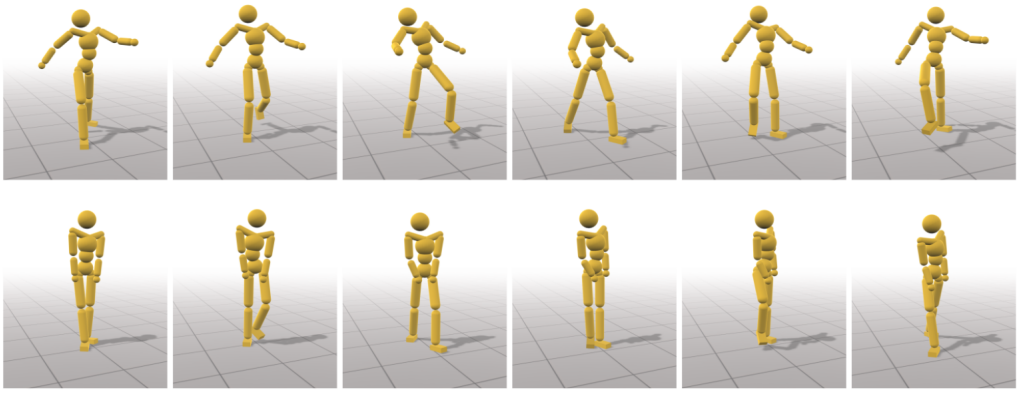
Chapter 1

Introduction

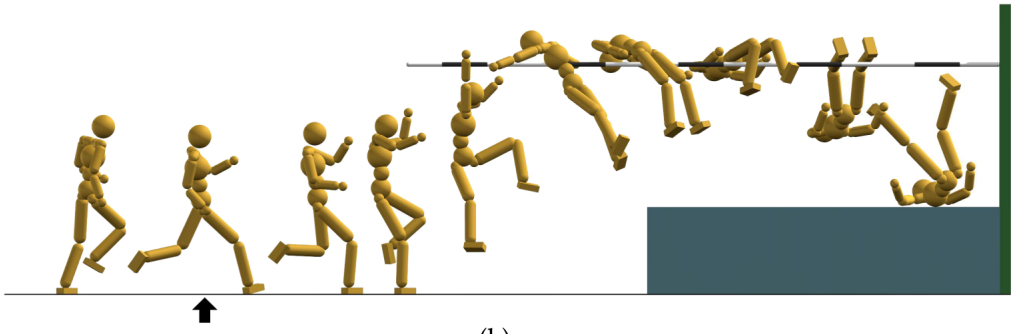
Humans and other animals are adept at performing various motor skills to finish diverse motor tasks, such as navigating around complex environments and manipulating different kinds of objects. A fundamental problem in physics-based character animation is how to develop virtual characters that behave like real creatures in a physically plausible manner. To solve this problem, we need to develop computational models to equip physics-based agents with a vast repertory of motor skills. Artificial agents endowed with such motor abilities has many far-reaching applications in various domains: in video games and films, virtual characters moving like humans could provide us with more immersive experiences. In robotics, control policies learned for simulated agents could be transferred and deployed in real robots, offering a powerful and general framework to help robots obtain physical abilities to move in the wild.

A lot of research efforts have been made to realize various motion skills for physics-based agents. The key challenge is to design robust control policies to drive the agents. A large number of control algorithms have been proposed to synthesize control policies for a great diversity of skills, including locomotion and manipulation [257, 226, 10, 119, 255]: some methods employ human-defined rules to design handcrafted controllers, such as SIMBICON [257], while other methods explicitly model the physics characteristics of dynamics systems and use optimization to search for good performing control policies, like model-predictive control algorithms deployed on the Atlas humanoid robots [40]. Although these traditional methods have achieved great success on many tasks, the controller designing and optimization process of these methods requires non-trivial expert knowledge or is specialized for certain tasks, limiting their applications to learning general and universal motor skills.

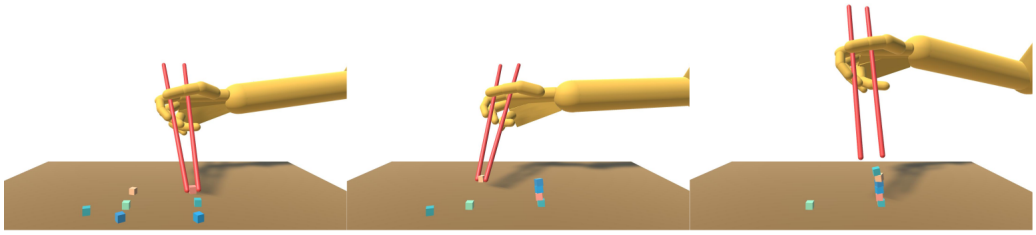
Recently physics-based character control has seen many advances with the development and adoption of deep reinforcement learning [163, 164, 165, 161, 15, 156, 170, 168, 48, 238, 253, 237]. Deep reinforcement learning offers a general and versatile framework to synthesize control policies for artificial agents. Instead of using expert-defined rules or modeling dynamic systems explicitly, RL agents collect data related to their interactions with en-



(a)



(b)



(c)

Figure 1.1: Stylized and discovered motion skills performed by simulated characters using our proposed methods.

vironments and optimize policies through trial and error. Due to its elegant formulation, DRL algorithms have been applied in character control to learn a wide range of motion skills: [62] showed that a diverse set of locomotion skills can emerge naturally using the same DRL framework. Later on, some following works combine DRL with readily available motion databases to learn impressive and challenging motion skills that are difficult to learn with traditional methods, such as playing basketballs [123] and stepping skills [245].

Despite the recent progress these DRL-based methods have made, most of these methods are imitation-based methods that require motion capture data or human demonstrations to be imitated by agents. The imitation objective of these methods encourages agents to behave like reference motions and inherently restricts agents from learning motor skills that are visually fundamentally different from the references. As a result, the variety of the produced skills is limited by the variety of reference data used in training. It remains non-trivial to produce motion skills that are out of the distribution of the reference motions. For example, it is unlikely to generate energetic running motions while provided with neutral walking motion databases. It is also impractical to capture all combinations of motions in different styles and performed by different actors. Moreover, there are many challenging motor tasks where motion examples are extremely difficult to obtain, such as dangerous parkour movements. These imitation-based methods cannot be applied to these motor tasks due to the absence of reference data.

In contrast to these physics-based characters whose development relies on imitations, humans or other animals can perform unseen stylized motions based on learned skills. e.g., We can perform a wide variety of stylized walking motions when we are proficient in basic locomotion skills. Humans can even discover novel motion strategies to solve challenging tasks through trial and error without referring to any previous successful demonstrations. For example, athletes have discovered a set of novel high jump techniques to jump over a crossbar, and some of the found strategies look counter-intuitive, like the Fosbury flop. In addition to the full-body motion skills, humans' motor skill discovery ability is also reflected in fine-manipulation motions like using tools. People have invented tremendous useful tools along with their using ways. One interesting example is chopsticks, a commonly-used eating utensil in Asia. Its using ways are so unique and special that many people wonder how ancient Chinese people invented and learned to use it. To conclude, as humans we have generalizable motor abilities and can perform unseen stylized skills easily and even discover novel motion strategies. We aim to equip artificial agents with stylized motor skills or strategy discovery abilities. To this end, we need to develop new computational models that do not rely on high-quality motion examples to enable physics-based characters to learn such stylized and novel skills. Some learned motion skills by our methods are visualized in Figure 1.1.

1.1 Overview

In this dissertation, we explore how to synthesize diverse stylized motor skills for physics-based characters without using task-specific motion examples. We propose several learning frameworks or algorithms to help simulated characters learn stylized locomotion, athletic jumping and tool manipulation skills.

1.1.1 Stylized Locomotion Skills

There are many publicly available motion capture databases that record human locomotions, and many physics-based control methods have succeeded in reproducing these recorded various locomotion skills faithfully. Instead of tracking or imitating these reference motions, we aim to build physics-based characters that are able to explore and learn new stylized locomotion skills from reference motions. To this end, we propose a DRL framework termed Spacetime bounds, to support stylized skill exploration. Our key insight is that the movement patterns exhibited by the reference motions are sufficient to guide simulated characters to learn successful skills for the given motion task, so the characters do not need to imitate the reference motions accurately. For example, the symmetric and cyclic locomotion pattern embedded in reference walking motions could provide strong priors for learning natural-looking and efficient locomotion skills. Spacetime bounds specify a set of spatial and temporal constraints of motions performed by characters, all motions that do not violate these constraints are considered valid. Therefore we do not need to design imitation objectives and can add style terms to encourage simulated characters to explore stylized skills within these constraints. We can also use more strict Spacetime bounds to learn some challenging skills that are difficult to learn by imitation-based methods.

1.1.2 Athletic Jumping Skills

High-quality motion capture data could be difficult to obtain for some motion tasks involving highly dynamic and dangerous motions. Therefore we cannot rely on motion examples to guide physics-based characters to learn such skills. As we mentioned previously, humans can discover diverse and ingenious motion strategies to accomplish challenging tasks without any access to successful demonstrations. We use a challenging and inspiring task: the high jump, to investigate motor skill learning without successful demonstrations. We propose a deep reinforcement learning framework to discover a set of visually different jumping strategies. We observe that the take-off states of simulated characters influence the resulting jumping strategies a lot. Different take-off states may lead to learning different jumping motions. Hence we design a diversity search algorithm to efficiently explore the take-off state space to discover as many jumping strategies as possible. To further improve the motion quality, we propose Pose Variational Autoencoder (P-VAE) to constrain the action space of simulated characters, thus eliminating the possibility of producing unnatural motions.

1.1.3 Tool Manipulation Skills

Human hands and tool usage are the special anatomy and function that helped drive the evolution of the human brain, which ultimately differentiated humans from the rest of the animal kingdom. An approach to enhance physics-based characters' abilities to interact with their surrounding environments is to enable these characters with dexterous hand manipulation skills. However, capturing accurate fine-scale hand manipulation motions could be impractical as severe occlusions, and subtle movements are involved. Building upon our work on learning diverse jumping skills, we design a learning and control system to enable physics-based characters to learn to use one of the most challenging tools: chopsticks, without using any motion examples. We decompose the original tool manipulation problem into two sub-problems: how to hold chopsticks properly and how to manipulate objects using chopsticks. We first combine DRL with Bayesian Optimization to discover diverse valid chopsticks gripping poses. Then the actual object manipulation skill is learned in two stages: a high-level kinematic motion planner synthesizes trajectories for the hands and chopsticks, and a low-level hand control policy is trained to track the generated motion trajectories using discovered gripping poses. Our framework can learn delicate object relocation skills using chopsticks in different styles, holding positions, and for multiple hand morphologies.

1.1.4 Hyperparmater Optimization

Based on our works on learning jumping and tool manipulation skills, we found that some novel motion skill discovery problems can be formulated as hyperparameter optimization problems. Compared with hyperparameter optimization for supervised learning, hyperparameter optimization for deep reinforcement learning is rarely explored. A common practice in physics-based character control is tuning hyperparameters manually. This tuning process requires expert knowledge and can be extremely time-consuming due to the heavy computation cost of DRL. To solve this problem, we propose a novel hyperparameter optimization algorithm: Curriculum-based Multi-Fidelity Bayesian Optimization to optimize hyperparameters for DRL-based motion skill learning systems automatically. Instead of evaluating candidate hyperparameters on the original difficult and computationally expensive tasks, we employ easier motor skill learning tasks to evaluate candidate hyperparameters. Easier tasks are faster to compute and can indicate unsuitable hyperparameters, thus enabling efficient search space pruning. Control policies learned on easier tasks can be transferred to the original difficult task to further improve data efficiency. Our hyperparameter optimization algorithm outperforms other state-of-the-art methods both in terms of final performance and data efficiency.

1.2 Contributions

The contributions of the work in this thesis can be summarized as the following:

- Chapter 3 proposes a deep reinforcement learning framework named Spacetime bounds, that enables physics-based characters to learn and explore motor skills from reference motions. Spacetime bounds can be used to learn various motion skills without using imitation or handcrafted rewards. It can be combined with style rewards to learn stylized motion skills. This work is published as: Li-Ke Ma, Zeshi Yang, Xin Tong, Baining Guo, KangKang Yin (Proc.EuroGraphics 2021). Learning and Exploring Motor Skills using Spacetime Bounds [132].

- In Chapter 4 we present a framework to help physics-based characters to discover diverse and natural-looking motor skills for challenging athlete sports, such as the high jump, without using any task-specific motion examples. Our framework can discover many well-known jumping strategies like the Fosbury flop. This work is published as: Zhiqi Yin, Zeshi Yang, Michel Vande Panne, KangKang Yin (Proc.SIGGRAPH 2021). Discovering Diverse Athletic Jumping Strategies [258].

- In Chapter 5 we extend our framework in Chapter 4 to equip physics-based characters with tool manipulation skills. More specifically, we show a learning and control system that enables anthropomorphic hands to use chopsticks to relocate objects. Our system does not require any motion examples and can learn chopsticks skills in different styles and for multiple hand morphologies. This work is published as: Zeshi Yang, KangKang Yin, Libin Liu (Proc.SIGGRAPH 2022). Learning to Use Chopsticks in Diverse Gripping Styles [251].

- Chapter 6 formulates the stylized skill discovery problem as a hyperparameter optimization problem, and proposes a novel optimization algorithm: Curriculum-based Multi-fidelity Bayesian Optimization to optimize hyperparameters for DRL-based animation systems. Compared with previous hyperparameter optimization methods, our method can find better hyperparameters to improve the performance of DRL-based animation systems in a data-efficient manner. This work is published as : Zeshi Yang, Zhiqi Yin (Proc.I3D 2021). Efficient Hyperparameter Optimization for Physics-based Character Animation [252].

Chapter 2

Background

The works in this thesis are built on several techniques, including deep reinforcement learning, with which to learn control policies for simulated characters to perform various motion skills, and Bayesian Optimization, with which to search for important parameters related to motor tasks. In this chapter, we first give a brief overview of basic concepts in deep reinforcement learning and Bayesian Optimization. We then provide a review of recent progress in physics-based character control. Since we leverage techniques related to kinematic character animations in some of our works, we also review the related literature on kinematic animation. Chapter-specific related works will be described in the corresponding chapters.

2.1 Deep Reinforcement Learning

2.1.1 Formulation

Many physics-based character control problems can be structured as a standard deep reinforcement learning problem: in an environment modeled as a Markov decision process, an agent interacts with this environment to maximize an given objective. More specifically, at each time step t, the agent observes the current state of the environment s_t, takes action a_t sampled from a distribution $\pi_\theta(a_t|s_t)$, which is usually modeled as a neural network. The environment responds to a_t and transits to the next state s_{t+1} according to a probabilistic dynamic model $p(s_{t+1}|s_t, a_t)$, and returns a reward $r(s_t, a_t)$. Starting from an initial state s_0, this procedure generates a trajectory $\tau = \{s_0, a_0, s_1, a_1, ...\}$. We calculate the accumulate reward of τ as $R(\tau) = \sum_t \gamma^t r(s_t, a_t)$. The goal of deep reinforcement learning is to find an optimal control policy π_θ to maximize the expected return of $R(\tau)$:

$$J(\theta) = \mathbb{E}_{\tau \sim p_{\theta(\tau)}} [R(\tau)] \tag{2.1}$$

where $p_{\theta(\tau)} = p(s_0) \prod_t \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$ is the possibility of τ induced by policy π_θ, and $p(s_0)$ is the initial state distribution. γ is the discount factor that lies between 0 and 1.

2.1.2 An Overview of Deep Reinforcement Learning Algorithms

Here we introduce several fundamental concepts in deep reinforcement learning and their notations. The value function $V_\pi(s_t)$ refers to a function that computes the accumulated rewards starting at state s_t using policy π. It is defined as:

$$V_\pi(s_t) = \mathbb{E}_{a\sim\pi(s)}[\sum_{i=0} \gamma^i r(s_{t+i}, a_{t+i})] \tag{2.2}$$

The state-action value function $Q_\pi(s_t, a_t)$ computes the future accumulated rewards by taking a_t at s_t and following policy π thereafter.

$$Q_\pi(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{a\sim\pi(s)}[\sum_{i=1} \gamma^i r(s_{t+i}, a_{t+i})] \tag{2.3}$$

The advantage function $A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$ measures the advantage of taking a_t at s_t, which can be interpreted as how many benefits we can get by taking a_t than the average return $V_\pi(s_t)$.

Deep reinforcement learning algorithms could be categorized into different paradigms, including Q-learning methods like Deep Q-Networks (DQN) [141], policy optimization methods like REINFORCE [235] and other methods that combine Q-learning and policy optimization, like Deep Deterministic Policy Gradient (DDPG) [115] and Soft Actor-Critic (SAC) [57]. Q-learning methods are mainly designed for tasks with discrete action spaces. They use data collected by agents to learn a Q value network $\hat{Q}_{\pi_\theta}(s_t, a_t)$ to approximate the optimal state-action value function $Q^*(s_t, a_t)$, then the actual policy can be induced as:

$$\pi_\theta(a_t|s_t) = \begin{cases} 1, & a_t = \arg\max_a \hat{Q}_{\pi_\theta}(s_t, a), \\ 0, & \text{Otherwise.} \end{cases} \tag{2.4}$$

One major advantage of Q-learning methods is that they are off-policy, which means that the Q-value network estimators can be trained using any previously collected data. They can use data effectively thus are sample efficient. However, Q-learning methods could be unstable, leading to learning policies that show limited performance.

Alternatively, policy optimization methods directly compute the gradient of $J(\theta)$ with regard to the parameters of the policy π_θ and use them to update π_θ. The $\nabla_\theta J(\theta)$ is derived as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(r(s_t, a_t) + \gamma V_{\pi_\theta}(s_{t+1}))] \tag{2.5}$$

This gradient estimation could be very noisy and has large variances, leading to unstable policy training. To alleviate this issue, [235, 207] introduced a baseline term $V_{\pi_\theta}(s_t)$ in the estimated gradient to reduce variances. Many following algorithms [189, 140] further replaced the value function $V_{\pi_\theta}(s)$ with a neural network approximator $\hat{V}_{\pi_\theta(s)}$. The updated

policy gradient can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(r(s_t,a_t) + \gamma \hat{V}_{\pi_\theta}(s_{t+1}) - \hat{V}_{\pi_\theta}(s_t))] \tag{2.6}$$

$$= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_{\pi_\theta}(s_t,a_t)] \tag{2.7}$$

In addition to these modified policy optimization algorithms, some advanced policy optimization algorithms such as Trust Region Policy Optimization (TRPO) [188] and Proximal Policy Optimization (PPO) [190] are proposed to further improve the stability of policy optimization. They are stable and reliable, making them suitable for various control tasks, such as robot control [167], playing video games [259] and solving NP-hard combinatorial optimization problems [74]. Compared with Q-learning methods that learn policies indirectly, policy optimization methods directly optimize the task objective, thus are more stable during training. Policy optimization methods can support tasks with both discrete and continuous action spaces. The main limitation of policy optimization methods is the data efficiency. They are on-policy methods that require data collected using the current policy to compute policy gradients. Data collected using history policies are not used anymore.

There exist a range of methods that combine both the advantages of Q-learning and policy optimization [57, 115]. Their main idea is to model the policy and state-action value functions using neural networks simultaneously. The state-action value network is optimized using previously collected data and then guides the optimization of the policy network. These methods carefully trade off between stability and data efficiency. Besides this, these methods can also support tasks with both continuous and discrete action spaces, which makes them perfect choices for continuous robot control tasks where data collection on hardware are quite expensive. Interested readers could refer to the corresponding papers for detailed descriptions of the algorithms.

In this thesis, we focus on policy learning on simulated characters, where data collection is fast and cheap by using state-of-the-art physics simulators. These modern simulators allow us to generate millions of simulation steps in just a few minutes. As a result, we adopt a widely-used and stable policy optimization method: Proximal Policy Optimization (PPO) [190] algorithm with different hyperparameters to optimize control policies for simulated characters. In each iteration, the agents collect a large amount of state transition tuples through interactions with the environment. Then the collected data is used to update the policy and value networks using TD(λ) [207] and GAE(λ) [189]. Readers could refer to the original paper [190] for more details about PPO.

2.2 Bayesian Optimization

In our work, we need to design reasonable motor skill simulation environments coupled with suitable DRL algorithms to achieve successful policy learning. There are many different

design choices for the simulation environments and DRL algorithms, and they are strongly related to the final performance of our designed systems. Hence, we must find adequate design choices to enable successful skill learning. Many design choice optimization problems could be formulated as expensive black-box function optimization problems solved by using Bayesian Optimization. Here in this section we briefly review the mathematical formulation of Bayesian Optimization.

2.2.1 Formulation

Bayesian Optimization refers to a class of sequential optimization strategies to optimize expensive black-box functions. Since objective functions are not expressed in closed forms and are computationally expensive, gradient information of objective functions is not available. BO optimizes objective functions purely through function evaluations and is designed to minimize the number of evaluations needed by querying the most promising and informative points. Given a set of current observations $D_t = \{(x_i, y_i)\}_{i=1}^{t}$, where y_i is a noisy measurement of $f(x_i)$, the *acquisition function* $a(x, D_t) : \mathcal{A} \to \mathbb{R}$ quantifies the utility of an arbitrary point. Maximizing the acquisition function will give us the point most worth trying next.

The acquisition function is designed for finding candidate points with both large values and rich information. Querying a point close to existing ones in D_t is less informative. Gaussian Process Upper Confidence Bound (GP-UCB) [199] is a popular acquisition function defined as:

$$a(x, D_t) = \mu_t(x) + \beta^{\frac{1}{2}}\sigma_t(x) \tag{2.8}$$

where $\mu_t(x)$ and $\sigma_t(x)$ are approximated posterior mean value and standard deviation of $f(x)$ respectively. $\mu_t(\cdot)$ favors candidates which are likely to have large values. $\sigma_t(\cdot)$ encourages querying informative points with high uncertainty. β enables a trade-off between exploitation and exploration.

Closed-form estimations for $\mu_t(\cdot)$ and $\sigma_t(\cdot)$ are available through a Gaussian Process (GP) [180] surrogate model of the objective function trained on D_t. A GP contains a prior mean function $m(x)$ and a kernel function $k(x, x')$. $m(x)$ encodes our prior belief of the objective function value. Kernel function $k(x, x')$ measures correlations between $f(x)$ and $f(x')$. Given $m(\cdot)$, $k(\cdot, \cdot)$ and existing observations D_t, μ_t and σ_t could be computed as:

$$\begin{aligned} \mu_t(x) &= k(x, X)(K + \eta^2 I)^{-1}(Y - m(x)) + m(x) \\ \sigma_t^2(x) &= k(x, x) + \eta^2 - k(x, X)(K + \eta^2 I)^{-1}k(X, x) \end{aligned} \tag{2.9}$$

where $Y \in \mathbb{R}^t, Y_i = y_i$; $X \in \mathbb{R}^{t \times d}, X_i = x_i$; $K \in \mathbb{R}^{t \times t}, K_{i,j} = k(x_i, x_j)$; $k(x, X) = [k(x, x_1), k(x, x_2), ...k(x, x_t)]$; η is the standard deviation of the observation noise.

Many choices of kernel functions exist, including Square Exponential (SE) kernel [180] and Matérn kernel [135]. In our works we adopt the SE kernel and Matérn$^5/_2$ kernel, defined

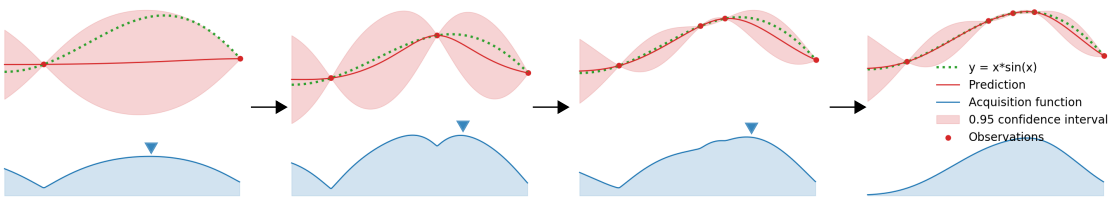


Figure 2.1: An example of applying BO to a one-dimensional function. Green dotted curve represents the unknown black-box function. Red points represent the queried points. Red curve along with the rosy shaded region indicate the predicted mean and 95% confidence interval. Blue curve is the acquisition function (It is scaled for the convenience of visualization). Blue downside arrows indicate the maximum of the acquisition function which is the point to be queried next.

as:

$$k_{SE}(x, x') = \sigma_f^2 \exp(-\frac{1}{2} d_\Lambda^2(x, x')) \tag{2.10}$$

$$k_{5/2}(x, x') = \theta(1 + \sqrt{5} d_\Lambda(x, x') + \frac{5}{3} d_\Lambda^2(x, x')) e^{-\sqrt{5} d_\Lambda(x, x')} \tag{2.11}$$

where θ, Λ and σ_f are learnable parameters of the GP. $d_\Lambda(x, x') = \sqrt{(x - x')^T \text{diag}(\Lambda)(x - x')}$ is the Mahalanobis distance. $\Lambda = \text{diag}(\lambda_1, \lambda_2, ...\lambda_t)$, where λ_i is the length scale of the i-th component of inputs. $f(\cdot)$ is flat across the i-th component of the input when λ_i is large.

Figure 2.1 shows a visual illustration of applying BO to a one-dimensional function. There are also other choices of acquisition functions and Bayesian surrogate models, such as Entropy search [63], Expected improvement [83] and deep gaussian process [32]. We refer interested readers to [191] for more in-depth reviews.

2.2.2 Applications in Hyperparameter Optimization

Parameter tuning and optimization is a ubiquitous task in many domains. Take designing physical robots as an example. Some parameters are derived or optimized in the designing and learning process, such as the parameters of a learned control policy, while other parameters are predetermined before the learning process, such as the morphology and actuator settings of robots. These parameters that are external to the optimization process are hyperparameters. Hyperparameters greatly affect the final performance of systems, but how the hyperparameters influence the system usually cannot be expressed by using functions in closed forms. Besides this, it is difficult to tell if the chosen hyperparameters are suitable before they are applied to target systems, while evaluating hyperparameters on target systems could take a lot of time and money. A common practice is to tune hyperparameters manually, which requires non-trivial expert knowledge and becomes prohibitive when given limited time and resources. As a result, an automatic and efficient hyperparameter optimization algorithm is in great need.

Mathematically, hyperparameter optimization problems can be formulated as expensive black-box function optimization problems:

$$\arg\max_{x \in \mathcal{A}} f(x) \tag{2.12}$$

where $\mathcal{A}$ is the hyperparameter space. $f(\cdot) : \mathcal{A} \to \mathcal{R}$ is the objective function, i.e. performance of control policy given hyperparameter x such as robot morphologies. The evaluation of $f(\cdot)$ involves optimizations, thus is costly and cannot be computed in closed forms. Due to its advantage of optimizing expensive black-box functions, BO is an ideal choice for this challenging task and has been used to optimize hyerparameters for a lot of systems. For example, BO has been used in computer graphics to find suitable settings with which to produce desirable visual effects, such as smoke animation tuning [17] and bidirectional reflectance distribution function (BRDF) design [18]. [175, 176, 75, 97] leveraged BO to search for suitable parameters of low-dimensional locomotion controllers and morphology parameters for real robots. Recently with the advancement of deep learning, there is a great appeal for automatic hyperparameter optimization of deep learning models [195, 100, 89]. BO has demonstrated its potential on such tasks for its promising sample efficiency [195, 196, 148].

In this thesis, we show that many motor skill discovery problems can be formulated as hyperparameter optimization problems: some low-dimensional structures, like the taking-off states of characters and tool gripping pose, have a big influence on the final learned skills. They can be viewed as hyperparameters of the motor skill learning tasks and are efficiently optimized by BO, leading to learning diverse and realistic motion skills for the high jump and chopsticks manipulation tasks.

2.3 Physics-based Character Animation

Physics-based character control methods generate motions with physical realism and environment interactions. The key challenge is the design or learning of robust controllers with which to drive simulated characters. Significant amounts of research endeavors have been invested into synthesizing motion controllers for a wide range of motor tasks. Here we classify these related methods into three categories: manually-designed controllers, trajectory optimization methods, and deep reinforcement learning methods.

2.3.1 Manually-designed Controllers

Conventionally manually designed controllers have achieved significant success for locomotion, e.g., [27, 28, 50, 44, 80, 226, 257, 110, 149]. The seminal work from Hodgins *et al.* demonstrated impressive controllers for athletic skills such as a handspring vault, a standing broad jump, a vertical leap, somersaults to different direction, and platform dives [67, 240]. Such handcrafted controllers are mostly designed with finite state machines (FSM) and

heuristic feedback rules, which require deep human insight and domain knowledge, and tedious manual trial and error. [273] thus proposed an interface to ease such a design process, and demonstrated controllers for diving, skiing and snowboarding. Some methods [226, 227] adopted Covariant Matrix Adaptive Evolution Strategy (CMA-ES) to automatically optimize parameters of locomotion controllers. Controls can also be designed using objectives and constraints adapted to each motion phase, e.g., [81, 34, 226, 227] or developed using a methodology that mimics human coaching [55]. In general, manually designed controllers remain hard to generalize to different strategies or tasks.

2.3.2 Trajectory Optimization

Trajectory optimization methods formulate a motor learning task as an optimization problem, in which an objective describing desirable behaviors is defined, and a sequence of actions taken by simulated characters are optimized to maximize the task objective. Due to its general mathematical formulation, trajectory optimization methods are employed to synthesize various skills [174, 9, 125, 124, 123, 71, 143, 142]. [9] designed a trajectory optimization framework with handcrafted task objectives to produce a lot of motions, including walking, running, getting up and gymnastic flips, etc. However, some of the resulting motions contain artifacts and show limited realism. When high-quality motion capture data is available, some trajectory optimization methods use motion clips to define a tracking objective that encourages simulated characters to imitate the reference motions. These methods can generate highly realistic motions with the help of motion examples, e.g, [125, 124]. For some tasks where motion examples are not available, motion quality can be improved by using sophisticated objective functions and domain knowledge. For example, [143, 144] proposed contact-invariant-optimization to synthesize many contact-rich motions such as hand manipulations and collaborative movements. Generally speaking, trajectory optimization methods without motion examples require non-trivial human insights and sometimes only work for simplified physical models [143]. Other drawbacks of trajectory optimization methods are that they are off-line methods and are sensitive to big perturbations, making them unsuitable for real-time and interactive applications. To overcome these limitations, some works [123, 122] learn feedback control laws on the trajectories optimized by trajectory optimization to further support real-time applications to some extent.

2.3.3 Deep Reinforcement Learning

Unlike trajectory optimization methods that directly optimize sequential actions, deep reinforcement learning methods optimize a control policy with which to output actions. Once trained, the learned policy network can be deployed in target environments and resists unexpected perturbations to some extent. Deep reinforcement learning has been widely used in physics-based character control to learn a wide range of motion skills. [62] demonstrated a rich set of locomotion behaviors emerging from complex environ-

ment interactions using simple reward functions. Similar to trajectory optimization methods with motion examples, some DRL-based methods focus on tracking-based controllers, which are capable of reproducing high-quality motion skills by imitating motion examples [163, 164, 165, 161, 71, 246, 108, 260]. These tracking-based methods can be combined with kinematic motion generators to support interactive control of simulated characters [156, 15, 236]. The most recent works [170, 168] adopted the generative adversarial imitation learning (GAIL) framework, in which the reward is given by a discriminator network trained on large motion databases, to learn various motion skills without using tracking rewards. These methods can be used to learn a low-dimensional skill spaces and further support generative motion control. Although these imitation-based methods have demonstrated their effectiveness on many tasks, the variety and quality of the learned skills strongly depend on the motion examples used during training. Similar to trajectory optimization, it is also difficult to use DRL-based methods to generate natural motions without using motion data. In addition to this drawback, the imitation objectives in these methods inherently restrict them from learning motion skills that are visually fundamentally different from references, thus do not support motion skill discovery tasks. [262] used expert knowledge such as symmetry gait and low-energy to learn natural-looking motions for humanoids, quadrupeds and even hexapods, while it is limited to locomotion and remains non-trivial to generalize to other motor tasks. In general, learning diverse and natural motion skills using DRL without motion examples remains an open and challenging problem.

2.4 Kinematic Character Animation

Different from physics-based character animation which predicts actions like torques at each joint to drive characters through physics simulation, kinematic-based methods directly predict poses of characters according to the current character pose and user inputs. Kinematic animation methods generally employ a set of motion clips to synthesize realistic character motions. The key challenge is to learn or construct a compact and generalized model to output character motions that satisfy a set of constraints like matching user inputs and generating natural motion transitions. Such kinematic models have evolved from graph structures [103, 183], to Gaussian Processes [112, 254], and recently deep neural networks [70, 69, 68, 106, 116, 202, 203, 266, 201, 267, 107]. The early works in kinematic animation methods are non-parametric methods in that they need to store the original motion capture data. Some non-parametric methods usually model the motion database as a graph structure, such as a motion graph [103] or a finite state machine, and design smart mechanisms to determine which motion clip to play to synthesize desirable character behaviours. These methods are widely used in game industries. Other non-parametric methods use classical machine learning techniques such as Gaussian process [112] and principal component analysis (PCA) [20, 138] to learn the low-dimensional motion features to improve the

quality and generality of the synthesized motions. These traditional machine learning methods usually suffer from high memory and computation costs. They need to store the whole motion dataset and the inference time could increase with using larger motion datasets. Compared with classical non-parametric machine learning models, deep neural networks are faster to infer, more compact at storing information and more powerful at extracting spatial-temporal features from motion sequences. Therefore, motion synthesis using deep neural networks has attracted much attention from the computer graphics and the machine learning community and demonstrated highly realistic and responsive kinematic character control. [45] first proposed to use RNN to model the human motion dynamics and predict the next frame of human pose given past poses. [70] used CNN to learn a low-dimensional human motion manifold from motion databases. This learned motion manifold can support various motion editing operations such as denoising, blending and style transfer. Later the authors of [70] designed a novel neural network names Phase-Functioned Neural Network (PFNN), whose weights change cyclically in accordance with motions, to synthesize responsive and environment-adaptive locomotion motions. This line of work starting from PFNN keeps improving and can produce impressive quadruped locomotions [266], human-object interactions [202, 267], and sports like martial arts [204] and playing basketball [203]. The most recent work DeepPhase [201] proposed a general neural network structure to learn an embedding of human motions and demonstrated that the learned periodic embedding can significantly help to improve neural motion synthesis in a number of tasks.

Although these recent deep learning methods can synthesize highly-realistic and responsive character motions, completely novel motions are still beyond their reach, as deep learning models are unlikely to generalize to data that is significantly out of the distribution of training data. As a result, kinematic methods cannot be directly applied to our skill discovery problem. Besides this, the physics constraints are usually not enforced in kinematic character control, so most kinematic methods cannot synthesize motions supporting physics interactions. However, we could adopt some useful techniques from kinematic models to facilitate learning physical skills, such as using the model in [70] to describe the style of a sequence of motions and using kinematic models to learn a low-dimensional action space for deep reinforcement learning. We will describe them in detail in Chapter 3 and Chapter 4 respectively.

Chapter 3

Learning and Exploring Stylized Motor Skills using Reference Motions

Humans are capable of performing stylized motor skills based on some basic skills. For example, people can locomote in different styles: happy walking and angry running. Virtual characters that are equipped with stylized motor skills can produce various and realistic motions, providing more immersive experiences for a large variety of applications like games and films. Our goal is to enable simulated characters to learn and explore stylized skills using reference skills: given a number of motion clips, how to learn motor skills in different styles to accomplish the same motor task?

In this chapter, we introduce a deep reinforcement learning framework that enables physics-based characters to learn and explore stylized motor skills from demonstrated skills. The key insight is that the movement patterns embedded in references are sufficient to guide simulated characters to accomplish task objectives. Therefore characters do not need to imitate references precisely. We introduce to use space-time constraints, termed spacetime bounds, to regulate the state space of characters. As we only rely on references to specify loose spacetime bounds, the learning is free of complex reward engineering. Our framework is robust with respect to low-quality motion references and provides opportunities to learn stylized skills that are visually different from references. Moreover, given hard spacetime bounds, our framework can learn challenging motion segments, which can be ignored by previous imitation-only methods. We compare our method with state-of-the-art tracking-based DRL methods and show that our method can guide stylized skills exploration. The supplementary video of this work is available at the link [1].

[1] `https://www.youtube.com/watch?v=_ziBY0k7irI`

3.1 Introduction

Recent years has seen many advances in physics-based character animation, especially since the application of Deep Reinforcement Learning (DRL) algorithms [123, 161, 239, 15, 156, 262]. These modern methods produce physically plausible motor skills either by tracking high quality reference motions [161, 156], or via smartly designed rewards [262]. However, tracking reference motions requires the existence of high quality example motions, and inherently prohibits any exploration of the potentially large feasible region of some motor skills for style variations. Designing good reward signals for DRL systems requires nontrivial domain knowledge and human insights, and does not directly support style exploration either.

We propose a simple DRL framework that can be used either standalone, or combined with imitation or hand-designed rewards. The proposed framework imposes spacetime constraints, hereafter referred to as *spacetime bounds*, as they mainly bound the character states in space and time, during the reinforcement training process. That is, the DRL system only samples and accepts states within the spacetime bounds specified.

The advantages of the proposed bounding-constraint-based framework over a tracking-based system include:

- Reward Simplification: Our framework can learn various motor skills with just a binary survival reward correlated to violations of spacetime bounds. An imitation or a hand-designed reward are not necessary anymore to reproduce motor skills. Simplified reward design and parameter tuning can potentially enhance a wider adoption of DRL methods in physics-based character animation.

- Increased Robustness: Tracking-based DRL methods stay close to a reference motion as much as possible. So when the reference motions are not in high quality, such as interpolated sparse keyframes, tracking methods may fail due to the physical implausibility of the reference motion. Spacetime bounds, however, are looser constraints. They allow freer exploration of the state space and thus may still succeed in finding physically plausible motions that resemble the low quality references. Meanwhile, spacetime bounds are hard constraints on the states. Challenging parts of reference motions, such as a quick 360° turn in a dynamic dance, can be ignored by an imitation reward to favor task success, but have to be respected when spacetime bounds are specified. Therefore our framework can reproduce skills more robustly and more faithfully.

- Style Exploration: As spacetime bounds only loosely constrain the DRL exploration, multiple styles of a motor skill may be discovered. We show that using simple heuristic terms to reward metrics such as energy levels, different locomotion styles can be easily discovered. Such style exploration is quite challenging if possible at all using

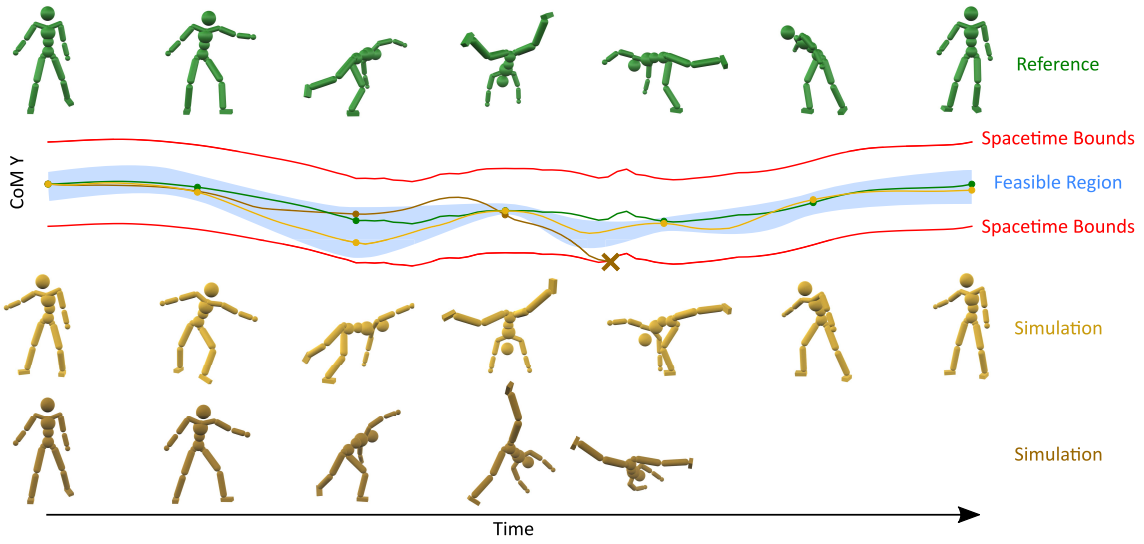


Figure 3.1: Learning cartwheels with spacetime bounds. The top green motion shows the reference, and the bottom yellow motions are simulations. The curves represent the Y position of the character's center of mass, and are colored to represent the reference (green), the simulations (yellow), and the spacetime bounds (red). The blue region illustrates the nonuniform feasible region under the given spacetime bounds. During training, episodes are terminated immediately once any spacetime bounds are violated, as shown in the bottom simulation.

only imitation-based methods, as the style-encouraging terms conflict with reference tracking rewards.

We review the most relevant prior works in Section 3.2. Then we detail the concept of spacetime bounds and its interaction with the feasible region of the dynamic skill in Section 3.3. Our DRL training framework and important parameter setups are described in Section 3.4. Various results are showcased in Section 3.5. Finally we conclude with discussions on limitations and future work in Section 3.6.

3.2 Related Work

In Chapter 2 we give a detailed review of the recent state-of-the-art physics-based animation methods, here we only focus on most relevant works about style exploration in character animations.

3.2.1 Kinematic Methods

Kinematic animation models have been used extensively for motion style generation or transfer. In the pre-deep learning era, traditional kinematic style transfer methods employ manually-defined features to transform input motions to target motions in different styles. Hsu *et al.* [73] used linear-time invariant systems (LTI) to describe the style differences

between stylized motions. After the LTI model is estimated, it can translate input motions to another style through simple linear operations. Ma *et al.* [133] proposed to use Bayesian network to model the relationships between user-defined style parameters and latent parameters that describes motion variations. Xia *et al.* [243] constructed a series of local mixtures of autoregressive models (MAR) to capture the complex relationships between styles of motion and demonstrated real-time heterogeneous motion style transfer. Another line of work [263, 220] modeled the human motion style in frequency domain, and successfully transferred motion styles between independent motions. In general, these conventional methods have demonstrated realistic and believable motion style transfer on some tasks. However, these methods usually require manually-designed features and additional preprocessing like motion alignment and are mainly demonstrated on locomotion tasks.

Recently with the development of deep learning, deep neural networks stand out as a more versatile and powerful tool to automatically extract style features from motions. Holden *et al.* [70, 69] showed that motion style features can be represented by the Gram matrix of the latent motion features extracted by neural networks and demonstrated motion style transfer by optimizing latent features. [194] further proposed a deep neural network module consisting of a pose network, a timing network and a foot-contact network to support real-time and efficient human motion style transfer. Some works [3, 157] used Generative Adversarial Neural Networks (GAN) that learns from an unpaired database of stylized human motions and enabled human motion transferring for styles not seen during training. In addition to GAN, other deep generative models such as Variational Autoencoder [38] and Flow-based models [233] are also adopted to achieve realistic and controllable motion style transfer. In our work, we aim to use deep reinforcement learning to learn physics-based motion skills in different styles. We draw on ideas from [70, 69] to use Gram matrices to define style rewards to guide the stylized skill learning. The technical details will be described in Section 3.3.3.

3.2.2 Physics-based methods

Recently physics-based character animation has seen many advances with the adoption of deep reinforcement learning [123, 161, 239, 156, 15, 170, 168]. Most of these methods are imitation-based methods, where physics-based characters learn skills by imitating corresponding references. However, these imitation-based methods do not work on low-quality references and might prohibit the exploration of new motion styles. An intuitive solution to encourage style exploration is to construct a family of parameterized motions using the given motion clip and then apply imitation-based methods to these parameterized motions. [109] proposed a parameterization neural network to adapt a given motion clip to different physical conditions, then the parameterization network and control policies are jointly optimized to produce a parameterized family of motion skills. Our work is directly inspired by prior works that imposed motion constraints instead of tracking references [6, 120]. [120]

transformed input sketches into physically plausible motions by detecting and imposing environment constraints. [6] encouraged motion variations with respect to hand-crafted goal constraints. Our method imposes spacetime constraints during deep reinforcement learning, thus enabling robust learning and exploration of a diverse set of skills.

3.3 Spacetime Bounds

Reference tracking-based DRL methods evaluate the quality of motor skills based on rewards that are real numbers. On the one hand, it is difficult to tell if a physics-based skill is successful or optimal from these numbers. On the other hand, such rewards only encourage but do not guarantee the similarity between the learned skill and the reference motion. For example, local joint angles or some portion of the skills may resemble the reference, while not the overall behavior or the full course of the motion.

We propose to constrain DRL learning with spacetime bounds instead. Spacetime bounds are constraints in space and time that correspond well with intuitive definitions of motor tasks. For example, a jump is a motor skill that for some duration of the motion both feet should leave the ground, and at no time of the motion should the character fall. In order to derive and analyze spacetime bounds, we start with the following definitions:

- The *state space*, denoted as $\mathcal{S}$, is the space of all possible states of a dynamical system.
- An *event* is a state-time tuple (s, t), and represents the system in state s at time t.
- A *Spacetime*, denoted as $\mathcal{M}$, is the space of all possible events.
- A *trajectory* is a sequence of events in spacetime $\mathcal{M}$. A trajectory is *causal* if all points on the trajectory obey applicable physical laws and dynamic constraints.

Definition 3.3.1. *A **spacetime bound** b is a subset of spacetime $\mathcal{M}$. We denote a set of spacetime bounds as $\mathcal{B}$.*

3.3.1 Feasible Regions and Spacetime Bounds

Spacetime bounds influence control learning via shaping the feasible region of a motor skill performed by a dynamical system.

Definition 3.3.2. *The **feasible region** associated with spacetime bounds $\mathcal{B}$, denoted as $Feasible(\mathcal{B})$, is the set of points on causal trajectories that $\mathcal{B}$ encloses.*

We first illustrate the influence of spacetime bounds on feasible regions using a toy problem. We restrict the motion of a mass point to the X axis. So its state can be fully described by (x, v), where x is the position and v is the velocity of the mass point. We then add forces to accelerate the mass point, but clip the acceleration at $a_{max} = 2m/s^2$.

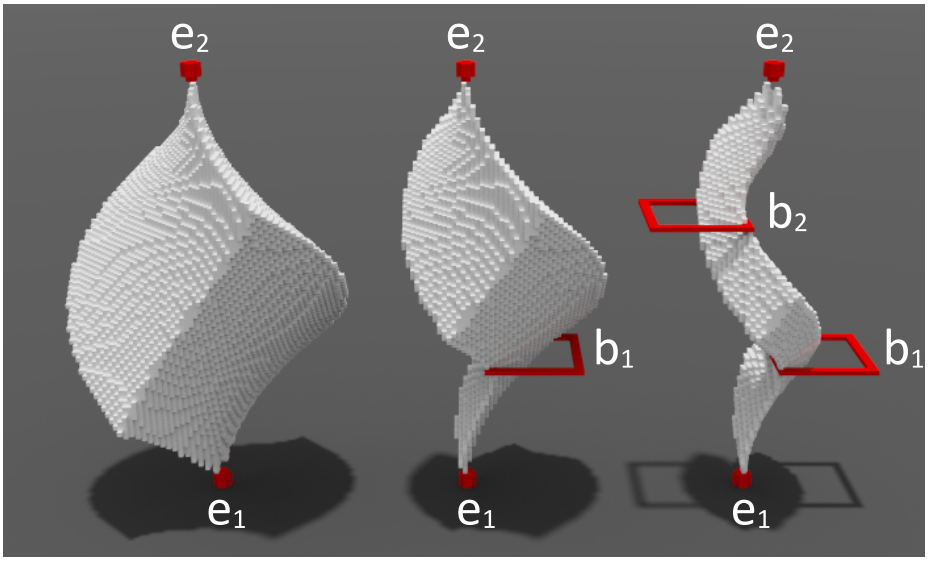


Figure 3.2: Feasible regions for a toy problem when more and more spacetime bounds are specified. Left: $\mathcal{B} = \{\{e_1\}, \{e_2\}\}$; middle: $\mathcal{B} = \{\{e_1\}, \{e_2\}, b_1\}$; right: $\mathcal{B} = \{\{e_1\}, \{e_2\}, b_1, b_2\}$. Note that in the right most case, the feasible region is much thinner than the specified red squarish spacetime bounds.

In Figure 3.2, we visualize the feasible regions that correspond to more and more imposed spacetime bounds. Events $e_1 = (x, v, t) = (0, 0, 0)$ and $e_2 = (x, v, t) = (0, 0, 5)$ are initially specified as the start and end of a causal trajectory. $Feasible(\mathcal{B})$ where $\mathcal{B} = \{\{e_1\}, \{e_2\}\}$ directly reflects the amount of trajectories that connect e_1 and e_2. We then impose more and more spacetime bounds to shrink the feasible region. For example, two spacetime bounds

$$\begin{aligned} b_1 &= \{(x, v, T_1) | x \in [0.5, 2.5], v \in [-1, 1], T_1 = 5/3\} \; and \\ b_2 &= \{(x, v, T_2) | x \in [-2.5, -0.5], v \in [-1, 1], T_2 = 10/3\} \end{aligned} \tag{3.1}$$

are added to generate the middle and right feasible regions in Figure 3.2. We can see a significant shrinkage of the feasible region with each added spacetime bound.

In more complicated systems such as a human-like character, the Degrees of Freedom (DoFs) are much larger and the system dynamics are much more complicated. We expect the feasible regions to shrink even faster. We illustrates such interactions between spacetime bounds and feasible regions in Figure 3.3 for a run jump skill. Note that this figure is only conceptual and not mathematically accurate as in Figure 3.2.

Generally speaking, different motor tasks performed by different dynamical systems have different intrinsic difficulties, which correlate to the volume of their feasible regions. For example, highly dynamic skills, such as a gymnastic backflip, are usually highly constrained. Their intrinsic feasible regions are usually quite small to start with. Consequently only professional athletes can perform such difficult motions in some optimal way. Low dynamic under-constrained motions, such as normal walking on flat ground, usually have larger initial

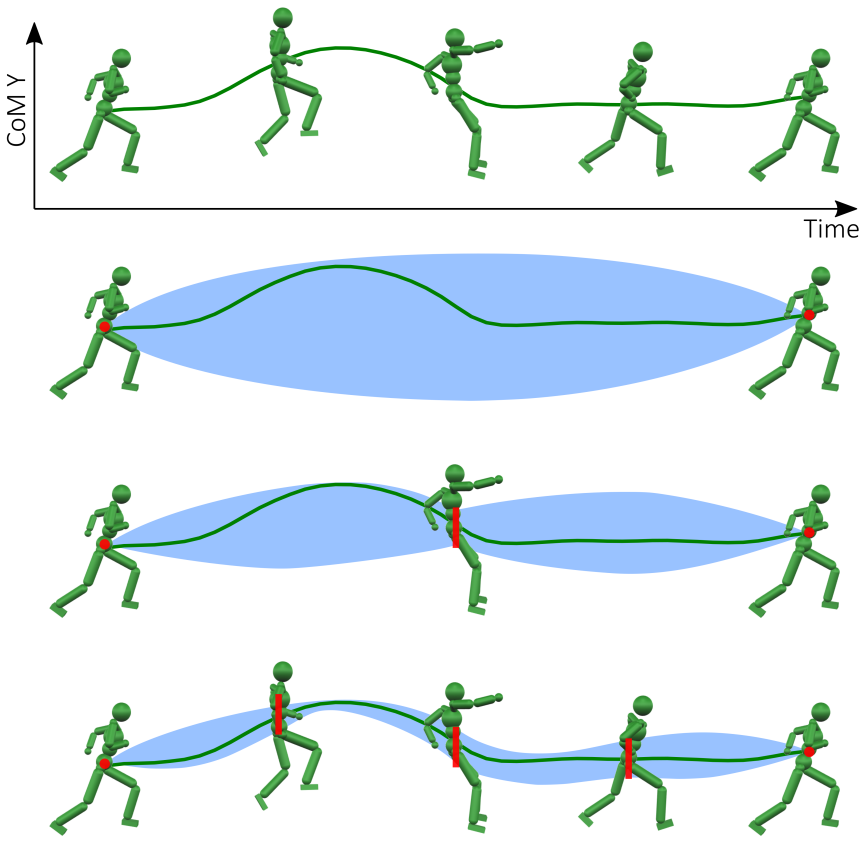


Figure 3.3: Feasible regions of a run jump skill when more and more spacetime bounds are specified. Green curves are the character's CoM Y positions in the reference motion. Red dots and bars represent spacetime bounds. Blue regions indicate the feasible regions associated with the specified spacetime bounds. Note that this is only a conceptual illustration. For all our results reported in Section 3.5, spacetime bounds are directly derived from the reference motion and applied to every frame of the simulation.

feasible regions to start with. As a result, normal people can locomote and even in different styles. In either case, feasible regions shrink rapidly when more and more spacetime bounds are imposed. Therefore we do not need to specify spacetime bounds precisely or tightly for control policy learning.

3.3.2 Policy Learning with Spacetime Bounds

Within a DRL framework, we sample an initial event e in $\mathcal{M}$ and run the current policy π_θ to generate a trajectory as long as it stays inside $\mathcal{B}$. Once the trajectory violates $\mathcal{B}$, we terminate the current episode immediately. If the learning converges, then the final optimal controller π_{θ^*} can guarantee to generate trajectories within $Feasible(\mathcal{B})$.

We can construct spacetime bounds from the reference trajectory $m(t)$. More specifically, we define a spacetime bound at t by restricting the state of the character to be within a region of size σ centered at $m(t)$. For example, the root orientation should be within 50° to the reference angle; or the end-effector positions should be within $0.5m$ to the reference positions. We set

$$\mathcal{B} = \{m(t, \sigma) | t \in [0, T]\}, \tag{3.2}$$

where $m(t, 0)$ is exactly the reference trajectory, and $m(t, \sigma)$ is the spacetime bound of size σ at time t. σ can be set uniformly for the whole duration of the motion, or as a function of time for finer control of the feasible region.

At the beginning of learning, the policy is bad and therefore the trajectories violate the spacetime bounds very fast. This causes early termination of the training episodes. As the policy improves, training episodes will automatically become longer and longer. We thus do not need to employ a time-based curriculum strategy as in DeepMimic [161], where episodes are heuristically scheduled to run longer and longer.

For motions that are highly constrained or in unstable equilibrium, the intrinsic corresponding feasible regions are narrow so loose spacetime bounds can already result in good controllers. For example, for a cartwheel we only need to bound its CoM positions (within $0.3m$) and orientations (within 40°). While for motions like locomotion, the initial feasible regions are relatively large so we need to tighten the spacetime bounds to learn skills that can reproduce the reference in a high fidelity. Alternatively, we can employ appropriate reward terms to guide the policy learning for more stylish skills.

We note that special cases of the spacetime bounds have been used before. For example, DeepMimic [161] employs an early termination scheme that terminates an episode whenever certain links, such as the torso, make contact with the ground. This is equivalent to the following spacetime bound:

$$\mathcal{B} = \{b_t | b_t = \{(s, t) | s \in S_{noUndesiredContacts}\}\}. \tag{3.3}$$

Our spacetime bounds, however, are more general and customizable. These bounds impose stronger constraints than the DeepMimic early termination scheme alone, and thus greatly improve the sampling efficiency by not wasting time on large unrecoverable regions of the state space. Our spacetime bounds are also more flexible than imitation rewards, so that style exploration is possible during policy learning as we discuss next.

3.3.3 Style Exploration

For motor skills with large default feasible regions, we can use style-related reward terms to explore different motion styles during DRL training. This is possible within our framework using spacetime bounds, as the bounds we specify are generally loose. In contrast, style exploration would be hard, if possible at all, using tracking-based methods, as the imitation reward and the style reward may conflict with each other or hard to tune.

Heuristic Style Reward – We first achieve style exploration with two heuristic reward terms:

- Kinematic Energy: We denote the kinematic energy calculated in the local frame defined at the CoM as E. Then the style reward for discovering motions at various energy levels is

$$r_s = \begin{cases} clamp(\frac{E_{max}-E}{E_{max}-E_{min}}, 0, 1), & \text{to decrease energy} \\ \\ clamp(\frac{E-E_{min}}{E_{max}-E_{min}}, 0, 1), & \text{to increase energy} \end{cases} \tag{3.4}$$

  where $[E_{min}, E_{max}]$ is the range where kinematic energy is linearly rewarded.

- Volume: We denote the convex hull volume of selected points on the character as V [11]. Then the style reward for discovering motions that span various volumes is

$$r_s = \begin{cases} e^{-\frac{V}{\alpha}}, & \text{to decrease volume} \\ 1 - e^{-\frac{V}{\alpha}}, & \text{to increase volume} \end{cases} \tag{3.5}$$

  where α is a scale parameter.

Data-driven Style Reward – We also illustrate style exploration with the data-driven style term described in [69], where the style of a motion is encoded by the Gram matrix of its features extracted from a deep autoencoder Φ. We directly use the autoencoder Φ trained from locomotion data in [69] for our experiments:

$$r_s = e^{-\frac{\|G_s - G\|_2}{\alpha}}, \tag{3.6}$$

where G_s is the Gram matrix of the motion in a desired style, and G is the Gram matrix of the simulated motion. We can then use stylized locomotion as our style descriptor to encourage the training to acquire policies that produce locomotion in similar styles. We also employ a regularization term to penalize large kinematic energy, large body linear accelerations and large joint angular accelerations, which may occur during style exploration with loose spacetime bounds. The regularization term is defined as:

$$r_{reg} = e^{-\sum_{i=1}^{N} w_i a_i / \beta_i}, \tag{3.7}$$

where a_i is the total kinematic energy, body linear accelerations, or joint angular accelerations. β_i is a scale factor and w_i is a weight that sums up to 1 for $i \in \{1, \cdots, N\}$. N is 1 plus the number of body parts and the number of joints. The final reward is thus

$$r = r_s \cdot r_{reg}. \tag{3.8}$$

The heuristic and data-driven style rewards as defined above are simple to implement and effective in discovering interesting motion styles as will be shown in Section 3.5.3.

3.4 DRL System

The policy learning task can be formulated as a standard reinforcement learning problem, where the agent interacts with the environment and learn a policy $\pi_\theta(a|s)$ to maximize its accumulated rewards. Readers can refer to Section 2.1 for more details of deep reinforcement learning.

3.4.1 States and Actions

The state vector s describes the current configuration of our system. Here we adopt a similar state representation as in DeepMimc. The state vector constitutes a phase index ϕ and the position p, orientation q, linear velocity v and angular velocity ω of each link and positions of chosen end- effectors. All these kinematic quantities, except for the root, are computed in a local frame attached to the root and aligned with the motion direction introduced in [131].

Similar to DeepMimic, the action vector a is a target pose described by internal joint rotations, which are then converted to torques by PD-controllers. Then the torques are applied to the corresponding joints to drive the simulated characters. Each rotational joint is either a 1-DoF revolute joint or a 3-DoF spherical joint. We choose to parameterize input orientations in quaternions, and target and output orientations in exponential maps [52].

Figure 3.4 shows our policy network, which consists of an open-loop feedforward controller (FFC) and a feedback controller (FBC). The FFC looks up the kinematic reference

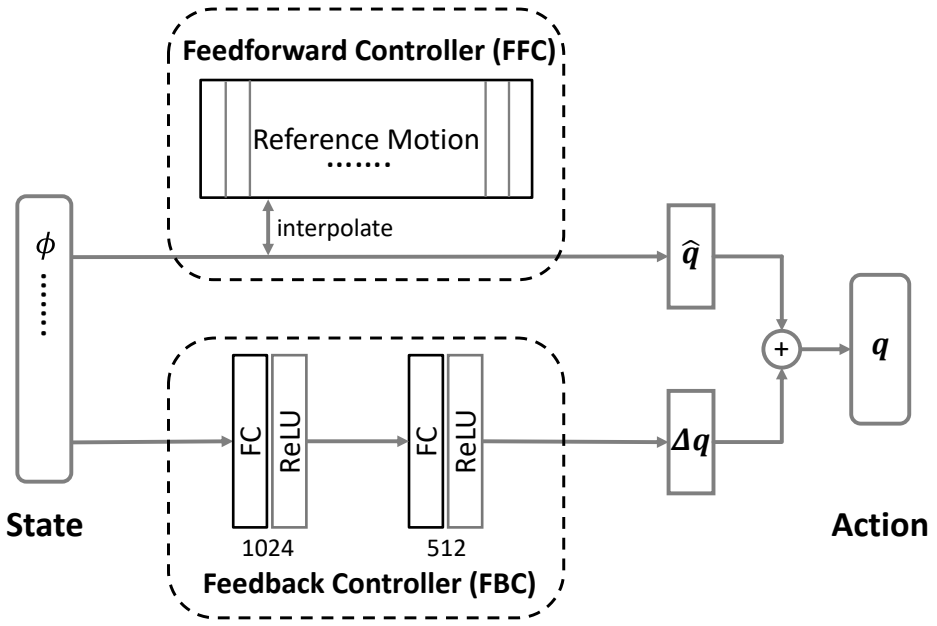


Figure 3.4: Our policy network consists of a feedforward controller (FFC) and a feedback controller (FBC). The FFC outputs default joint angles $\hat{q}$ according to the phase index ϕ and the reference motion. The FBC is a two-layer fully-connected neural network with 1024 and 512 hidden units respectively. It takes in the full state vector and outputs offset joint angles Δq. The final control signal $q = \hat{q} + \Delta q$.

motions according to the phase index ϕ and outputs the default target joint angles. The FBC outputs corrective offsets to the default target angles. Then the final PD targets are computed by adding the default angles and their corresponding offsets. The FBC network consists of two fully-connected layers with 1024 and 512 hidden units respectively, with ReLU activations for each layer. Our value network is similar to the feedback control branch of the policy network, except that the final output is a scalar that estimates the value function $V_\pi(s)$.

3.4.2 Initial States Adaptation

The initial state distribution ρ_0 determines the states in which an agent begins each episode. Reference State Initialization (RSI) proposed in [161] has been proven to help the agent to access desirable states early in the learning, and thus improves the efficiency and robustness of DRL algorithms. In our framework, the RSI strategy is equivalent to setting the initial events to $E = m(t, 0)$ where t is uniformly sampled. The RSI strategy does not work well for challenging skills or low-quality references, however. First, the feasible regions are not uniform in size across time. At critical points where the motions are more likely to fail, drawing more samples will likely help. Second, the sampled initial states could be infeasible from low-quality references. A strategy to help evolve the set of initial states into the feasible regions will be beneficial. We thus develop the following two adaptation strategies to further

improve the robustness of learning: importance sampling from E, and evolving E by training experiences.

Importance sampling of reference motion

Generally speaking, learning is more likely to fail early from initial states sampled around critical points of a motor skill. We therefore sample more around critical points of the reference motion. More specifically, we first uniformly divide the reference motion into n segments. Denote the average estimated return starting from states in segment k as w_k. Then for each new episode, we sample each segment with probability.

$$p(k) = (1-u)\frac{\exp\left(-w_k/v\right)}{\sum_{i=1}^{n}\exp\left(-w_i/v\right)} + \frac{u}{n}, \tag{3.9}$$

where $u = 0.2$ is the probability to sample uniformly for all our experiments, and v is a scale parameter adaptively set to $(\max w_k - \min w_k)/3$. This is similar to the adaptive sampling scheme described in [156]. There are also other adaptive schemes to facilitate learning in the literature, such as utilizing value functions to guide the sampling of body shapes [239], or learning progressively from easier tasks to harder tasks [245], which is equivalent to the adaptive sampling proposed in [156, 239].

Initial States Evolution

When the reference motion is of low quality, such as hand animation from sparse keyframes, sampled initial states may be outside of the feasible region. We develop a scheme to select elite states from experiences to gradually guide them into the feasible region. More specifically, we assign a buffer for each motion segment to hold the current set of elite initial states. These buffers are initialized with m events sampled from the original reference motion segments. We then sample initial states from these buffers for training. After each epoch, we use the Boltzmann distribution to draw m elite samples from all collected samples to overwrite the buffer:

$$p(l) = \frac{\exp\left(-w_l/v\right)}{\sum_i \exp\left(-w_i/v\right)}, \tag{3.10}$$

where w_l is the estimated return of the l-th state in the buffer, and v is a scale parameter that we set to $(\max w_l - \min w_l)$. We note that the ASI (Adaptive State Initialization) scheme described in [169] and the CMA (Covariance Matrix Adaptation) scheme described in [123] share a similar motivation, but our scheme is much simpler to implement and works well for all the results shown in this chapter.

Training

We train our control policies using the Proximal Policy Optimization(PPO) algorithm [190]. We use $TD(\lambda)$ [207] and $GAE(\lambda)$ [189] to train our value network and actor network. We

terminate training episodes when the specified spacetime bounds are violated as shown in Figure 3.1, or when the end state is reached, or when the time limit is exceeded.

3.5 Results

We implement our system using Pybullet [29] and Pytorch [159]. Our character model weighs 45 kg, and has 15 internal joints and 34 DoFs in total. Each joint except for the root is actuated by a stable PD controller [210]. We run the simulation at 600 Hz, and the control at 30 Hz.

For DRL training, we use a binary survival reward at each control step. If the state is within the spacetime bounds, the character earns a reward 1, otherwise 0 and the training episode is terminated immediately. When there are other rewards, such as style encouraging rewards, we simply multiply the binary survival reward with the other rewards. Since the survival reward is 1, the reward value is simply the value of the other rewards. We set the reward discount factor $\gamma = 0.95$, and $\lambda = 0.95$ for both $TD(\lambda)$ and $GAE(\lambda)$. The learning rate is 2.5×10^{-6} for the actor network and 1.0×10^{-2} for the critic network. In each training iteration, we sample 4096 state-action tuples in parallel on multi-core processors. The training batch size is 256. We report the performance statistics on a desktop with an 18-core Intel i9-7980XE CPU, where training takes about 30 minutes to 24 hours, depending on the length and difficulty of the motor skills.

We first show that our method can train controllers performing basic tasks with only spacetime bounds in Section 3.5.1. Then in Section 3.5.2, we present more challenging cases where imitation-based methods fail but our method can still succeed. Next in Section 3.5.3, we demonstrate a variety of motion styles synthesized by our method, either using heuristic or data-driven style rewards. Lastly in Section 3.5.4 we conduct ablation studies on the sensitivity of spacetime bounds, and the effects of FFC and the initial states adaptation.

3.5.1 Learning without Tracking

We first train the character to learn to follow reference motions without any imitation reward, but with loose spacetime bounds $m(t, \sigma)$ with σ set as follows:

- CoM x, y, z positions: 0.2 m
- Root and joint orientation: 0.7 $rad \approx 40^\circ$
- Endeffector distance: 0.5 m

CoM positions are compared in each dimension x, y, z separately. Joint orientations are compared in their local frames. Endeffector distances are compared in a direction-invariant local frame, same as the one that we use to derive the state representations as described in Section 3.4.1. Table 3.1 lists the number of samples needed for learning each skill, and

Figure 3.5: Cyclic skills trained with loose spacetime bounds and no imitation reward.

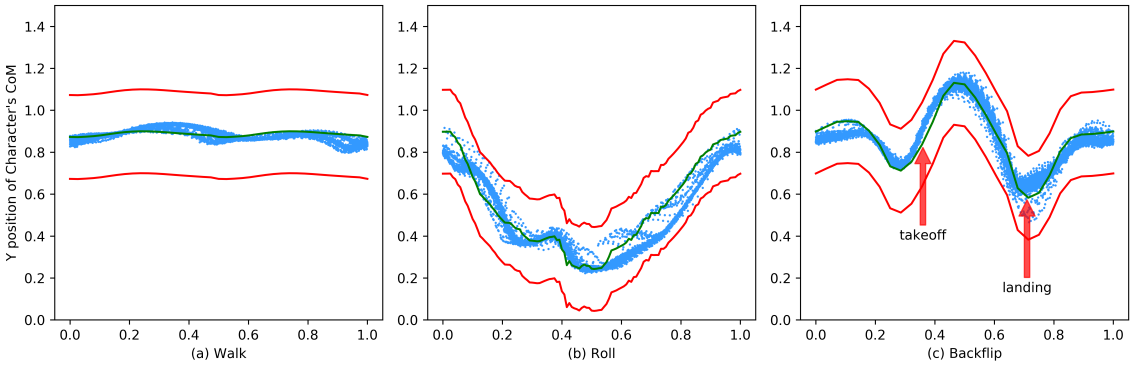


Figure 3.6: CoM Y position with respect to the phase index. The green line is the reference and the red lines represent the spacetime bounds. Blue dots are samples from 100 episodes of the learned policy. We can see that the feasible regions of the learned skills lie inside the spacetime bounds; and their sizes are not uniform. For example in backflip (c), it is narrower around the takeoff and wider around the landing.

task	$T_{\text{cycle}}(s)$	Ours $N_s(10^6)$	DeepMimic $N_s(10^6)$
walk	1.26	4.08	23.80
run	0.80	4.11	19.31
jump	1.77	41.63	25.65
roll	2.02	12.31	23.00
cartwheel	2.72	17.35	30.45
dance	1.62	10.00	24.59
run jump	1.53	11.02	24.07
backflip	1.75	41.20	31.18

Table 3.1: Number of training samples N_s needed for character to be able to perform tasks for 20 seconds without falling.

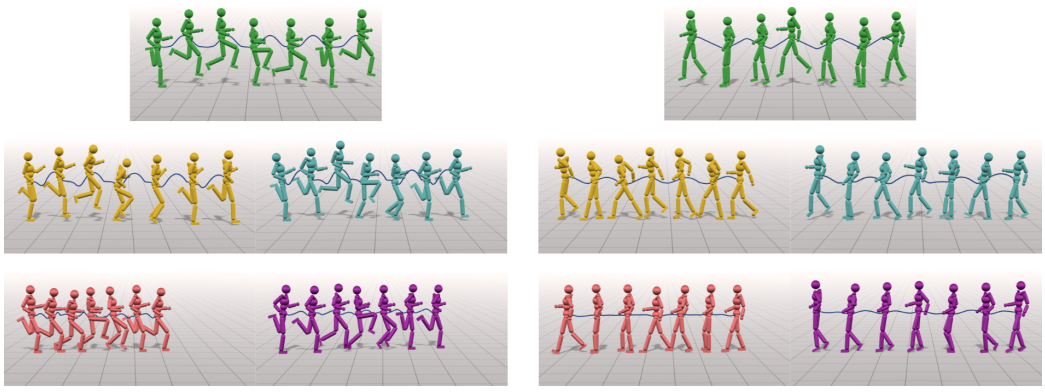
Figure 3.5 shows the snapshots of learned skills. We note that not all spacetime bounds are needed for all skills. For highly dynamic motions such as the cartwheel, learning can be successful with just the bounds on CoM position and root orientation. Generally speaking, spacetime bounds for COM position and root orientation bound the overall behavior of the character, such as moving forward or moving upward. Bounds on local joint orientations address the local pose similarity. Bounds on end-effectors prevent accumulated errors on a chain caused by individual joint angle deviations.

Figure 3.6 illustrates the relationship between the reference motion and the spacetime bounds. It plots the CoM Y position with respect to the phase index. The reference $m(t)$ is centered by the spacetime bounds $m(t, \sigma)$. As shown in Figure 3.6(b), sampled CoM Y positions from simulations controlled by the learned policy lie inside the spacetime bounds, and can notably deviate from the reference. Figure 3.6(c) reveals that the feasible region is narrower around critical points such as the taking off phase of a backflip, and wider around stable regions such as the landing phase.

Tuning spacetime bounds for internal joints is usually easy, as the reference can be tracked well through PD controllers, especially for those joints that do not support the body weight, such as upper body joints in locomotion skills. CoM positions and root orientations, however, are not directly actuated and controlled. Yet it is critical to follow them to achieve the desired motor skills. In addition, errors for the CoM horizontal position accumulate in time. Therefore, it takes some time to experiment proper spacetime bounds for the CoM and the root. Nevertheless, we are able to find one set of spacetime bounds for all the motions that we tested.

3.5.2 Robustness to Challenging Cases

Tracking-based DRL systems may fail for difficult motor skills or due to references in poor qualities. For example, when the reference motion contain fast body rotations or sophisticated foot steps, the tracking-based methods tend to sacrifice tracking fidelity of the challenging motion segments in order to gain longer survival which leads to bigger total



(a) Leg-lifting Jog (b) Backward Jog

Figure 3.7: Robustness comparison between policies trained using our DRL framework with different options and DeepMimic, for low quality references from keyframed sparse poses. Green characters represent the reference motions. Yellow, blue, and red characters represent policies trained using our DRL framework with spacetime bounds only, with both spacetime bounds and imitation rewards, and with imitation rewards only. Purple characters represent policies trained using DeepMimic. For both skills, policies trained with spacetime bounds can better reproduce the intended skills.

rewards. Our proposed spacetime bounds, however, set hard boundaries for how much the learned skill can deviate from the reference, so the final motion quality will be guaranteed to be within a certain neighborhood of the original motion. On the other hand, our spacetime bounds treat all trajectories within the preset bound equally, so that deviation from low quality reference is easier to learn robust skills. In contrast, tracking-based methods have to compromise the quality of learned skills for more accurate tracking of the bad reference to gain more rewards.

We compare policies trained using our DRL framework with different options and the original DeepMimic, keeping same parameter settings wherever possible. Figure 3.7 shows two reference motions of rather low quality, keyframed from sparse jogging poses. For both tasks, policies trained with spacetime bounds can reproduce the reference tasks in similar styles, while policies trained without spacetime bounds or the original DeepMimic cannot. Figure 3.8 shows that the policy trained with spacetime bounds can reproduce 360° jump turns in a break dance, but policies trained without spacetime bounds or DeepMimic cannot.

We also perform retargeting experiments using our framework, as shown in Figure 3.9. We use the built-in Atlas robot model from Pybullet [29]. The morphology of this robot is significantly different from that of humans. The model also uses three revolute joints to model spherical joints, so we parameterize rotations of spherical joints, such as shoulders and hips, using Euler angles. We directly use the same motions captured from human performers as references without any kinematic retargeting. Our framework is able to physically retarget locomotion and gymnastics skills onto the robot model using spacetime bounds only.

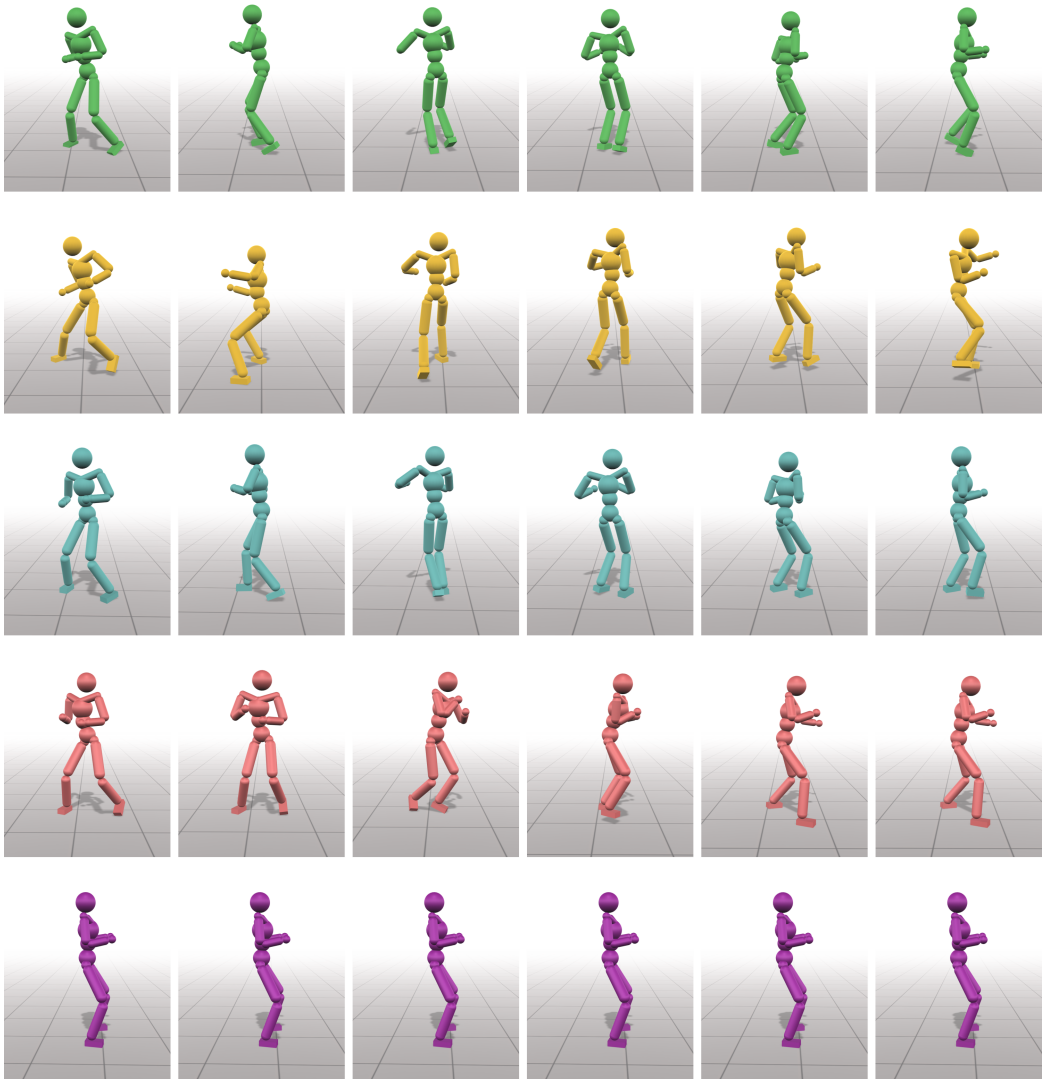


Figure 3.8: Robustness comparison between policies trained using our DRL framework with different options and DeepMimic, for a challenging skill break dance. Green characters represent the reference motions. Yellow, blue, and red characters represent policies trained using our DRL framework with spacetime bounds only, with both spacetime bounds and imitation rewards, and with imitation rewards only. Purple characters represent policies trained using DeepMimic. The reference character performs two 360° jump turns during the dance. The policies trained with spacetime bounds are able to reproduce the reference motion, but the one trained with imitation rewards only cannot reproduce the challenging parts. The policy trained using DeepMimic fails completely.

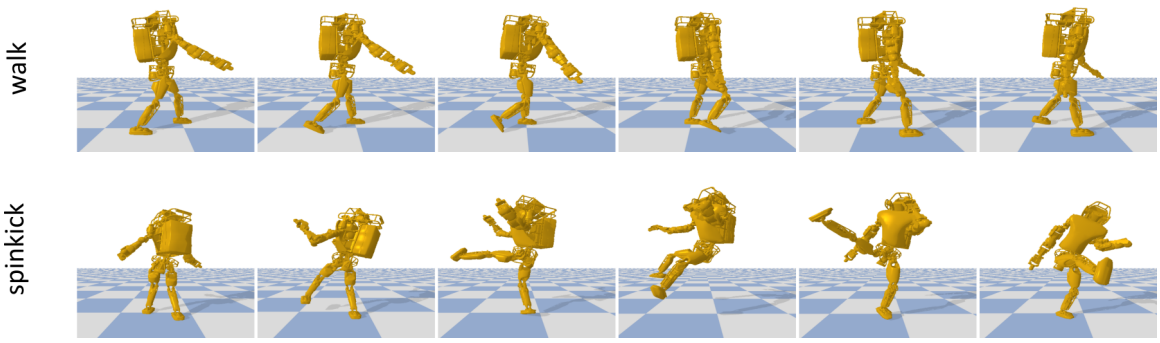


Figure 3.9: Retargeting human motions to an Atlas robot. Policies are trained using our DRL framework with spacetime bounds only.

3.5.3 Style Exploration

Heuristic Styles

We combine spacetime bounds with the heuristic style rewards as described in Section 3.3.3 to generate stylized motions, some of which are shown in Figure 3.10. We set $[E_{min}, E_{max}] = [20, 100]$ for the kinematic energy term in Equation 3.4, and $\alpha = 0.12$ for the volume term in Equation 3.5. In order to generate visually different styles, we deliberately loosen the spacetime bounds in Section 3.5.1. For example, we only bound the CoM positions, and root, ankle and neck orientations for the cartwheel.

Data-driven Styles

We test style exploration using the data-driven style reward as described in Section 3.3.3 for motions selected from the CMU mocap database [1], as shown in Figure 3.11. We directly use the autoencoder from [69] to encode stylistic walking motions in Figure 3.11 (b) and (d) to high level features for Gram matrix computation. A neutral run as shown in Figure 3.11(a) is used to derive relevant spacetime bounds for DRL training. The Gram matrix for the simulated motion is computed from the current state backward in time for a fixed duration of one locomotion cycle. Then the data-driven style term can be evaluated by Equation 3.6 from the two Gram matrices. We again use larger spacetime bounds than those given in Section 3.5.1 to support more aggressive style explorations for these cases.

Comparison

We conduct comparative experiments to validate the necessity of spacetime bounds in style exploration within our DRL framework as given in Section 3.4. We use a weighted average of an imitation reward term and the style reward terms as described in Section 3.3.3. That

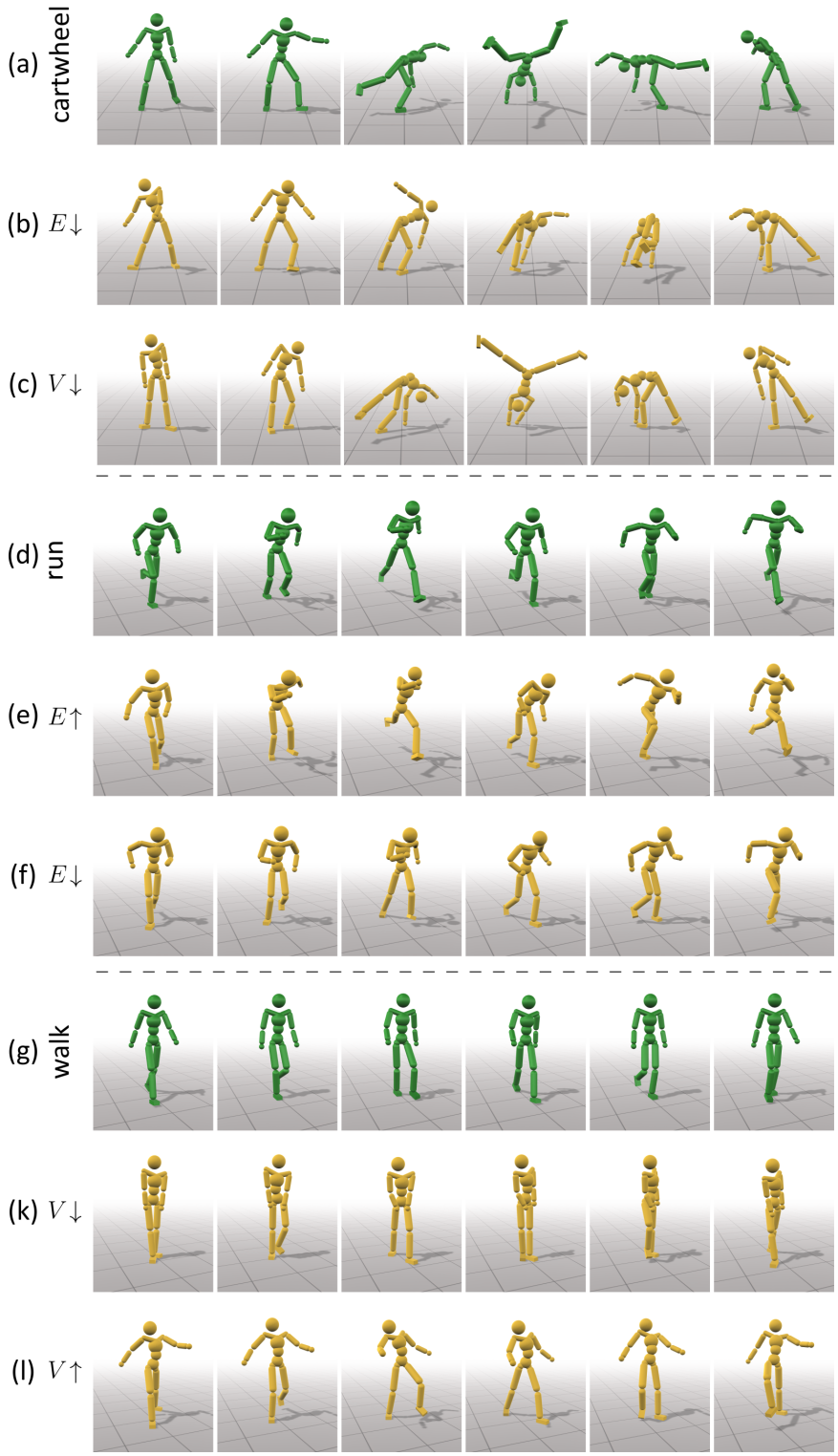


Figure 3.10: Styles explored by using heuristic style rewards with spacetime bounds. Reference motions are colored in green and stylized motions in yellow. $E\uparrow$ and $E\downarrow$ show motions with kinematic energy encouraged and discouraged respectively. $V\uparrow$ and $V\downarrow$ show motions with full body volume encouraged and discouraged.

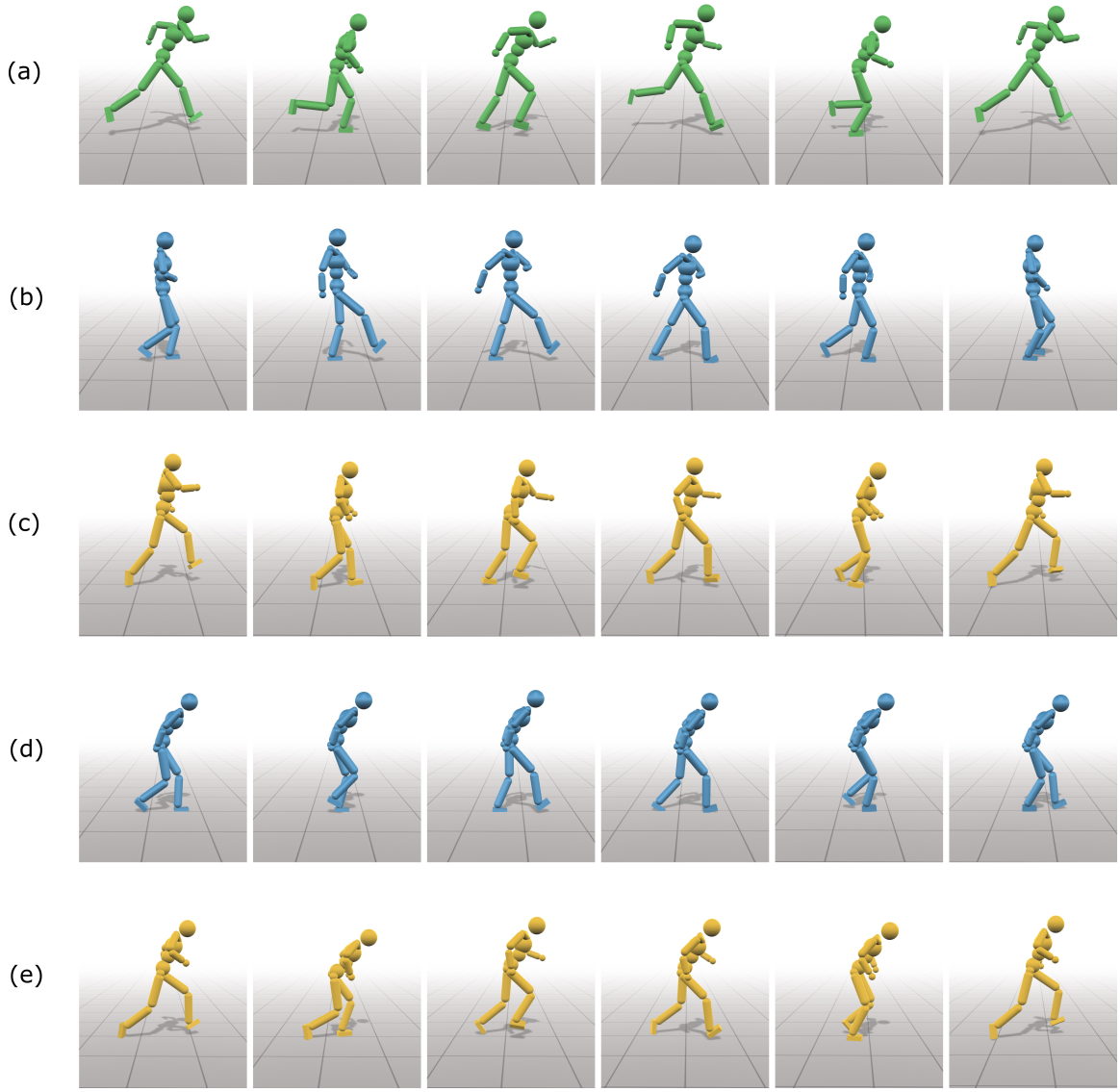


Figure 3.11: Style exploration with data-driven style rewards. (a) a neutral run; (b) a happy walk; (c) a happy run trained from the spacetime-bounded neutral run with the style descriptor extracted from the happy walk; (d) a bent walk; (e) a bent run trained from the spacetime-bounded neutral run with the style descriptor extracted from the bent walk.

w_s	0.5	0.6	0.7	0.8
cartwheel $E \downarrow$	no style	unstable	failed	failed
cartwheel $V \downarrow$	no style	no style	unstable	failed
dance $E \downarrow$	no style	weird style	failed	failed
dance $E \uparrow$	no style	slight style	failed	failed

Table 3.2: Style exploration using an imitation reward rather than spacetime bounds. $E \uparrow$ and $E \downarrow$ show motions with kinematic energy encouraged and discouraged respectively, and $V \uparrow$ and $V \downarrow$ show motions with full body volume encouraged and discouraged respectively. Most results are style-less motions, or unstable or failed skills.

is, using a total reward defined as follows:

$$r = (1 - w_s)r_i + w_s r_s, \tag{3.11}$$

where r_i is the imitation reward from DeepMimic [161] and r_s is the style term. We test a range of w_s listed in Table 3.2. As we can see, lower w_s results in successful motor skills, but prohibits exploration of new styles. While higher w_s results in either unstable or failed motor skills. We also conduct another experiment with both the spacetime bounds and the composite imitation and style rewards in Equation 3.11. In such case, stylized skills can be learned for all w_s without any failure. The learned skills are slightly less stylized as compared with just using the spacetime bounds and the style rewards, due to interference from the imitation term.

3.5.4 Ablation Study

Spacetime Bounds Sensitivity

We analyze the sensitivity of spacetime bounds by training a series of controllers using spacetime bounds of different sizes, varying from tight to loose. For under-constrained motions with large initial feasible regions, such as walking, the learned policies change notably with respect to the size of the specified spacetime bounds. The looser the sapcetime bounds are, the more relaxed and less constrained the learned walk is. For highly constrained motions with narrow initial feasible regions, such as a cartwheel, too tight spacetime bounds result in training failures, and too loose bounds do not influence the learned skills notably. These results reveal the interactions between the spacetime bounds and the inherent feasible regions of dynamic skills.

Effect of the Feedforward Controller

Integrated neural network models without separating FBC and FFC can successfully learn many motor skills [161]. However, separating FFC from FBC can result in much faster learning, as proven by a few recent works [15, 156]. The disadvantage of using FFC is that

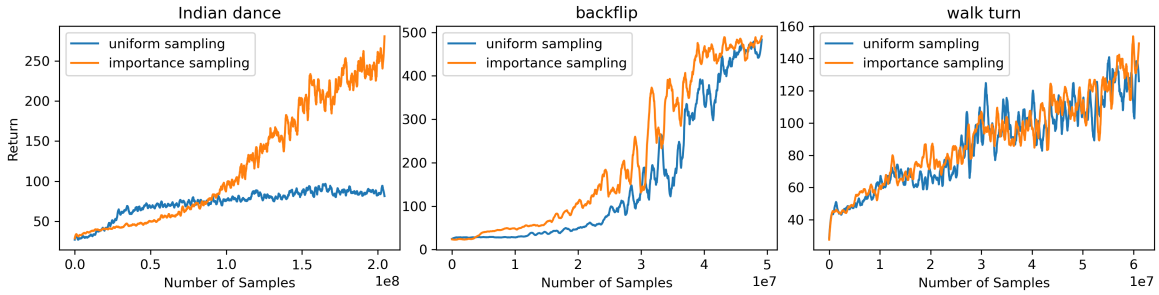


Figure 3.12: Uniform vs. importance sampling for learning three skills: Indian dance, backflip and walk turn. (Left) Importance sampling is much more superior for the Indian dance, which contains a few 360° turns. (Middle) Importance sampling is beneficial for the backflip, which contains critical points such as the takeoff. (Right) The two sampling methods do not differ much for the walk turn.

the original reference data need to be stored in a memory to compute the final policy. Another option is to make future trajectories as part of the inputs of the policy, as used in [166, 228]

Initial States Adaptation

Figure 3.12 shows training with and without the importance sampling of reference motion as described in Section 3.4.2. For challenging tasks such as the Indian dance, importance sampling greatly improves the learning and convergence speed. For less challenging tasks, the benefit of importance sampling will gradually diminish. Regarding the effect of initial states evolution as described in Section 3.4.2, we visualize the evolved initial states together with the reference states for a break dance in Figure 3.13. Around sharp turns of the motion, evolved initial states significantly deviate from the reference initial states, which enables the successful learning of this challenging skill.

3.6 Conclusion and Discussion

In this work we present a deep reinforcement learning framework that robustly learns various motor skills via spacetime bounds. We show that our method can learn motor skills without any imitation or hand-crafted rewards. Thus our method is more robust to low-quality references. Spacetime bounds impose hard constraints to the training process, so the learned skills are guaranteed to be close to challenging parts of reference skills. Moreover, spacetime bounds can be easily combined with style exploration rewards to further encourage motion skill style explorations.

All the spacetime bounds used in this work are deprived from the reference motions, e.g., the target orientations plus or minus 40 degrees. The working range of spacetime bounds are usually quite large. Intuitively, when we aim to reproduce the reference skills faithfully, the bounds should be tighter. When we need to reproduce the reference, the bounds should

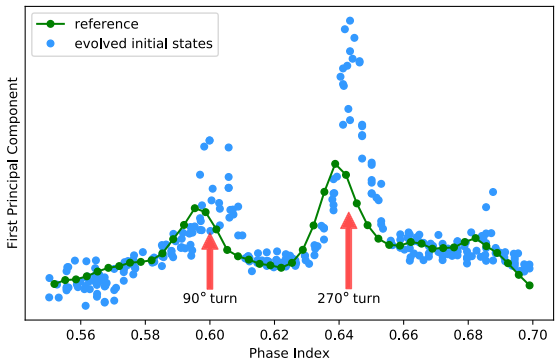


Figure 3.13: The first principal components of the evolved initial states (blue dots) and the reference states (green dots) for a segment of a break dance. We use the modified locally linear embedding [270] for principal component analysis. Around sharp turns, the evolved initial states significantly deviate from the reference initial states, which enables the successful learning of this challenging skill.

be tighter. While when we aim to explore different styles, the bounds should be looser. The working range of spacetime bounds also depends on the type of motions. For highly constrained motions like flips, the size of the bounds do not affect the results that much, and therefore minimal tuning is required. For under-constrained motions such as locomotions, the size of the bounds affect the styles of the output.

We would like to note that DRL training with imitation rewards alone already works well for reconstructing high-quality reference skills without challenging parts. However, spacebounds can still be used together with imitation rewards for such cases to replace ad-hoc early termination techniques such as undesired body-ground contacts. For low-quality reference motions, skills containing challenging parts, or style exploration, spacetime bounds can be used, either alone or with imitation rewards, to lead the learning to more faithful reconstruction of desired skills or more stylish skills.

In this work, we use some heuristic rewards, such as energy minimization, to improve the naturalness of produced skills. However, using heuristic rewards alone cannot guarantee the synthesized motions look natural. We find that our method could output motions with some artifacts. It is possible to adopt the idea of Generative Adversarial Imitation Learning (GAIL): using a trained discriminator neural network to grade the naturalness of learned skills, to further improve the motion quality. Besides this, this framework still relies on motion references for style exploration and thus cannot be applied to motor tasks where references are extremely lacking. We will describe how to learn stylized skills for these tasks in Chapter 4 and 5.

Lastly, currently we use static spacetime bounds. It would be interesting to investigate how to adaptively change the size of the bounds during training, which might lead to faster convergence of policy learning. For example, for critical motion segments important to

successful skill execution, the bounds should be tight to guarantee the successful learning of the skill. While for other less important segments, we can impose looser bounds to encourage style exploration.

Chapter 4

Diverse Motor Skills Discovery for Jumping Tasks

For motor skill learning tasks where motion capture data is available, we can use our method described in chapter 3 to learn various stylized skills for simulated characters. However, some dangerous and challenging movements like parkour or high jump are not suitable for motion capture, and it is difficult to obtain high-quality motion capture data for these skills. As a result, we cannot use imitation-based methods to learn such skills directly. Moreover, even though we have access to the task-specific motion capture data, the imitation-based methods inherently restrict simulated characters from learning motion skills that are fundamentally different from the reference skills. How to synthesize diverse motion skills without using task-specific motion examples remains a challenging open problem.

In this chapter we focus on one of the most challenging athletic endeavors: the high jump, in which athletes are asked to jump unaided over a horizontal bar placed at measured heights without dislodging it. Over the past 150 years, various high jump techniques have been invented and redefined world records, culminating in the now well-known and dominant Fosbury flop. Interestingly, athletes did not discover these innovative techniques through imitation. They developed such ingenious skills by trial and error, which motivates us that we can also learn these challenging skills in a data-free manner. Among all the machine learning methods, reinforcement learning stands out as a perfect choice for our task, as it is also a trial-and-error process where agents gradually refine their policies to maximize the final returned rewards. Therefore, we use reinforcement learning to help simulated characters discover diverse athletic high jump skills without using task-related references.

We present a framework that enables the discovery of diverse and natural-looking motion skills for athletic movements such as the high jump. Given a task objective and an initial character configuration, we use deep reinforcement learning for automatic skill learning. To facilitate the learning of realistic human motions, we propose a Pose Variational Autoencoder (P-VAE) to constrain the actions to a subspace of natural poses. In contrast to motion imitation methods, a rich variety of novel jumping skills can naturally emerge

by exploring initial character states through a sample-efficient Bayesian diversity search (BDS) algorithm. A second stage of optimization that encourages novel policies can further enrich the unique strategies discovered. Our method allows for the discovery of diverse and novel strategies for athletic jumping motions such as high jumps and obstacle jumps with no motion examples and less reward engineering than prior work. The supplementary video of this work is available at the link [1].

4.1 Introduction

Athletic endeavors are a function of strength, skill, and strategy. For the high-jump, the choice of strategy has been of particular historic importance. Innovations in techniques or strategies have repeatedly redefined world records over the past 150 years, culminating in the now well-established Fosbury flop (Brill bend) technique. In this work, we demonstrate how to discover diverse strategies, as realized through physics-based controllers which are trained using reinforcement learning. We show that natural high-jump strategies can be learned without recourse to motion capture data, with the exception of a single generic run-up motion capture clip. We further demonstrate diverse solutions to a box-jumping task.

Several challenges stand in the way of being able to discover iconic athletic strategies such as those used for the high jump. The motions involve high-dimensional states and actions. The task is defined by a sparse reward, i.e., successfully making it over the bar or not. It is not obvious how to ensure that the resulting motions are natural in addition to being physically plausible. Lastly, the optimization landscape is multimodal in nature.

We take several steps to address these challenges. First, we identify the take-off state as a strong determinant of the resulting jump strategy, which is characterized by low-dimensional features such as the net angular velocity and approach angle in preparation for take-off. To efficiently explore the take-off states, we employ Bayesian diversity optimization. Given a desired take-off state, we first train a run-up controller that imitates a single generic run-up motion capture clip while also targeting the desired take-off state. The subsequent jump control policy is trained with the help of a curriculum, without any recourse to motion capture data. We make use of a pose variational autoencoder to define an action space that helps yield more natural poses and motions. We further enrich unique strategy variations by a second optimization stage which reuses the best discovered take-off states and encourages novel control policies. Figure 4.1 shows two representative jumping strategies discovered by our method.

In summary, our contributions include:

[1] `https://www.youtube.com/watch?v=DAhZ6oDoNHg`

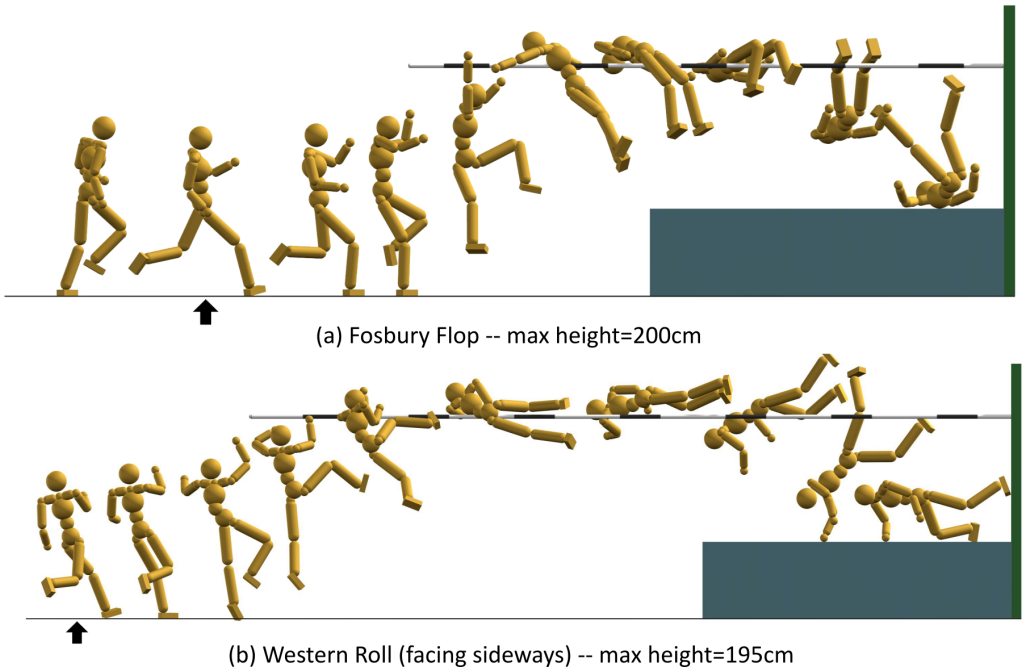


Figure 4.1: Two of the eight high jump strategies discovered by our two-stage learning framework, as achieved by physics-based control policies. The first stage is a sample-efficient Bayesian diversity search algorithm that explores the space of take-off states, as indicated by the black arrows. In the second stage we explicitly encourage novel policies given a fixed initial state discovered in the first stage.

- A deep reinforcement learning system which can discover common athletic high jump strategies. The discovered strategies include the Fosbury flop (Brill bend), Western roll, and a number of other styles. We further evaluate the system on box jumps and on a number of high-jump variations and ablations.
- The use of Bayesian diversity search for sample-efficient exploration of take-off states, which are strong determinants of resulting strategies.
- Pose variational autoencoders used in support of learning natural athletic motions.

4.2 Related Work

We build our work upon previous work from several areas, including character animation, diversity optimization, human pose modeling, and high jump analysis from biomechanics and kinesiology. A review of recent character animation methods is provided in Chapter 2.3 and Chapter 2.4. Here we mainly describe and discuss other related topics.

4.2.1 Diversity Optimization

Our goal is to discover as many as possible diverse jumping strategies, so we need computational models to optimize the diversity of motions. Diversity optimization is a problem of great interest in artificial intelligence [61, 221, 200, 25, 173, 111]. It is formulated as searching for a set of configurations such that the corresponding outcomes have a large diversity while satisfying a given objective. Diversity optimization has also been utilized in computer graphics applications [137, 6]. For example, a variety of 2D and simple 3D skills have been achieved through jointly optimizing task objectives and a diversity metric within a trajectory optimization framework [6]. Such methods are computationally prohibitive for our case as learning the athletic tasks involve expensive DRL training through non-differentiable simulations, e.g., a single strategy takes six hours to learn even on a high-end desktop. We propose a diversity optimization algorithm based on the successful Bayesian Optimization (BO) philosophy for sample efficient black-box function optimization.

In BO, objective functions are optimized purely through function evaluations as no derivative information is available. Usually a BO algorithm consists of two main components: a Bayesian surrogate model and an acquisition function. The surrogate model, usually modeled as a Gaussian Process [180], is maintained to predict the value of the objective function along with the uncertainty of the estimation. The acquisition function is then repeatedly maximized for fast decisions on where to sample next for the actual function evaluation. The next sample needs to be promising in terms of maximizing the objective function predicted by the surrogate model, and also informative in terms of reducing the uncertainty in less-explored regions of the surrogate model. Chapter 2 provides a detailed description of various BO algorithms.

We propose a novel acquisition function to encourage the discovery of motion skills. We also decouple the exploration from the maximization for more robust and efficient strategy discovery. We name this algorithm Bayesian Diversity Search (BDS). The BDS algorithm searches for diverse strategies by exploring a low-dimensional space defined at the take-off moment. Initial states exploration has been applied to find appropriate initial conditions for desired landing controllers [56]. In the context of DRL training, initial states are usually treated as hyperparameters rather than being explored.

Recently a variety of DRL algorithms have been proposed to discover diverse control policies in machine learning, e.g, [41, 269, 206, 4, 192, 57, 26, 72, 65, 187]. These methods usually mainly encourage exploration of unseen states or actions by jointly optimizing the task and novelty objectives [269], or optimizing intrinsic rewards such as heuristically defined curiosity terms [41, 192]. We adopt a similar idea for novelty seeking in Stage 2 of our framework after BDS, but with a novelty metric and reward structure more suitable for our goal. Coupled with the Stage 1 BDS, we are able to learn a rich set of strategies for challenging tasks such as athletic jumping.

4.2.2 Natural Human Pose

In biomechanics and neuroscience, it is well known that the muscle synergies, or muscle co-activations, serve as motor primitives for the central nervous system to simplify movement control of the underlying complex neuromusculoskeletal systems [154, 272]. In character animation, human-like character models are much simplified, but are still parameterized by 30+ DoFs. Yet the natural human pose manifold learned from motion capture databases is of much lower dimensions [69]. The movement of joints are highly correlated as typically they are strategically coordinated and co-activated. Such correlations have been modeled through traditional dimensionality reduction techniques such as PCA [20], or more recently, Variational AutoEncoders (VAE) [58, 116].

We rely on a VAE learned from mocap databases to produce natural target poses for our DRL-based policy network. Searching behaviors in low-dimensional spaces has been employed in physics-based character animation to both accelerate the non-linear optimization and improve motion quality [184]. Throwing motions based on muscle synergies extracted from human experiments have been synthesized on a musculoskeletal model [31]. Most recently DRL methods either imitate mocap references directly [15, 239], making strategy discovery hard if possible, or adopt a *de novo* approach with no examples at all [62], which often results in extremely unnatural motions for human-like characters. Close in spirit to our work is [179], where a low-dimensional PCA space learned from a single mocap trajectory is used as the action space of DeepMimic for tracking-based control. We aim to discover new strategies without tracking, and we use a large variety of generic motions to deduce a task-and-strategy-independent natural pose space. We also add action offsets to the P-VAE output poses so that large joint activations can be achieved for powerful take-offs.

Reduced or latent parameter spaces based on statistical analysis of poses have been used for grasping control [10, 24, 152]. A Trajectory Generator (TG) can provide a compact parameterization that can enable learning of reactive policies for complex behaviors [78]. Motion primitives can also be learned from mocap and then be composed to learn new behaviours [166].

4.2.3 History and Science of High Jump

The high jump is one of the most technically complex, strategically nuanced, and physiologically demanding sports among all track and field events [37]. Over the past 150 years, high jump has evolved dramatically in the Olympics. Here we summarize the well-known variations [37, 85].

- The Hurdle: the jumper runs straight-on to the bar, raises one leg up to the bar, and quickly raises the other one over the bar once the first has cleared. The body clears the bar upright.

- Scissor Kick: the jumper approaches the bar diagonally, throws first the inside leg and then the other over the bar in a scissoring motion. The body clears the bar upright.
- Eastern Cutoff: the jumper takes off like the scissor kick, but extends his back and flattens out over the bar.
- Western Roll: the jumper also approaches the bar diagonally, but the inner leg is used for the take-off, while the outer leg is thrust up to lead the body sideways over the bar.
- The Straddle: similar to Western Roll, but the jumper clears the bar face-down.
- Fosbury Flop: The jumper approaches the bar on a curved path and leans away from the bar at the take-off point to convert horizontal velocity into vertical velocity and angular momentum. In flight, the jumper progressively arches their shoulders, back, and legs in a rolling motion, and lands on their neck and back. The jumper's Center of Mass (CoM) can pass under the bar while the body arches and slides above the bar. It has been the favored high jump technique in Olympic competitions since used by Dick Fosbury in the 1968 Summer Olympics. It was concurrently developed by Debbie Brill.

In biomechanics, kinesiology, and physical education, high jumps have been analyzed to a limited extent. We adopt the force limits reported in [150] in our simulations. Dapena simulated a higher jump by making small changes to a recorded jump [33]. Mathematical models of the Center of Mass (CoM) movement have been developed to offer recommendations to increase the effectiveness of high jumps [5].

4.3 System Overview

We now give an overview of our learning framework as illustrated in Figure 4.2. Our framework splits athletic jumps into two phases: a run-up phase and a jump phase. The *take-off* state marks the transition between these phases, and consists of a time instant midway through the last support phase before becoming airborne. The take-off state is key to our exploration strategy, as it is a strong determinant of the resulting jumping motion. We characterize the take-off state by a feature vector that captures key aspects of a state, such as the net angular velocity and body orientation. This defines a low-dimensional take-off feature space that we can sample in order to explore and evaluate a variety of motion strategies. While random sampling of the take-off state features is straightforward, it is extremely computationally impractical as evaluating one sample involves an expensive DRL learning process which takes hours even on modern machines. Therefore, we introduce a sample-efficient Bayesian Diversity Search (BDS) algorithm as a key part of our Stage 1 optimization process.

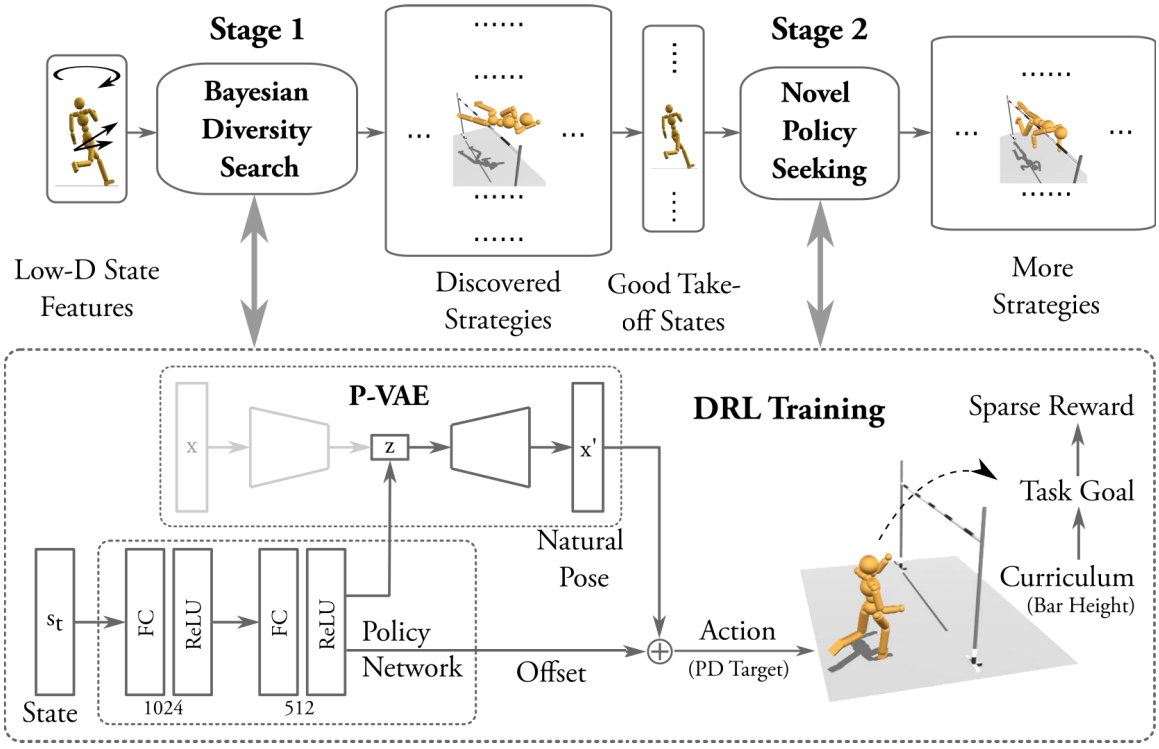


Figure 4.2: Overview of our strategy discovery framework. The Stage 1 Bayesian Diversity Search algorithm explores a low-dimensional feature vector of the take-off states to discover multiple jumping strategies. The output strategies from Stage 1 and their corresponding take-off states are taken as input to warm-start further training in Stage 2, which encourages novel policies that lead to additional visually distinct strategies. Both stages share the same DRL training component, which utilizes a P-VAE to improve the motion quality and a task curriculum to gradually increase the learning difficulty.

Given a specific sampled take-off state, we then need to produce an optimized run-up controller and a jump controller that result in the best possible corresponding jumps. This process has several steps. We first train a run-up controller, using DRL, that imitates a singe generic run-up motion capture clip while also retargeting the desired take-off state. For simplicity, the run-up controller and its training are not shown in Figure 4.2. These are discussed in Section 4.6.1. The main challenge lies with the synthesis of the actual jump controller which governs the remainder of the motion, and for which we wish to discover strategies without any recourse to known solutions.

The jump controller begins from the take-off state and needs to control the body during take-off, over the bar and to prepare for landing. This poses a challenging learning problem because of the demanding nature of the problem, the sparse rewards and the difficulty of also achieving natural human-like movement. We apply two key insights to make this task learnable using deep reinforcement learning. First, we employ an action space defined by a subspace of natural human poses as modeled with a Pose-Variational AutoEncoder (P-VAE). Given an action parameterized as a target body pose, individual joint torques are

then realized using PD-controllers. We additionally allow for regularized offset PD-targets that are added to the P-VAE target to enable strong take-off forces. Second, we employ a curriculum that progressively increases the task difficulty, i.e., the height of the bar, based on current performance.

A diverse set of strategies can already emerge after the Stage 1 BDS optimization. To achieve further strategy variations, we reuse the take-off states of the existing discovered strategies for another stage of optimization. The diversity is explicitly incentivized during this Stage 2 optimization via a novelty reward, which is focused specifically on features of body poses at the peak height of the jump. As shown in Figure 4.2, Stage 2 makes use of the same overall DRL learning procedure as in Stage 1, albeit with a slightly different reward structure.

4.4 Learning Natural Strategies

Given a character model, an environment, and a task objective, we aim to learn feasible natural-looking motion strategies using deep reinforcement learning. We first describe our DRL formulation in Section 4.4.1. To improve the learned motion quality, we propose a Pose Variational Autoencoder (P-VAE) to constrain the policy actions in Section 4.4.2.

4.4.1 DRL Formulation

Our strategy learning task is formulated as a standard reinforcement learning problem, where the character interacts with the environment to learn a control policy which maximizes a long-term reward.

States The state s describes the character configuration. We use a similar set of pose and velocity features as those proposed in DeepMimic [161], including relative positions of each link with respect to the root, their rotations parameterized in quaternions, along with their linear and angular velocities. Different from DeepMimic, our features are computed directly in the global frame without direction-invariant transformations for the studied jump tasks. The justification is that input features should distinguish states with different relative transformations between the character and the environment obstacle such as the crossbar. In principle, we could also use direction-invariant features as in DeepMimic, and include the relative transformation to the obstacle into the feature set. However, as proved in [132], there are no direction-invariant features that are always singularity free. Direction-invariant features change wildly whenever the character's facing direction approaches the chosen motion direction, which is usually the global up-direction or the Y-axis. For high jump techniques such as the Fosbury flop, singularities are frequently encountered as the athlete clears the bar facing upward. Therefore, we opt to use global features for simplicity and robustness. Another difference from DeepMimic is that time-dependent phase variables

are not included in our feature set. Actions are chosen purely based on the dynamic state of the character.

Initial States The initial state s_0 is the state in which the agent begins each episode in DRL training. We explore a chosen low-dimensional feature space $(3-4D)$ of the take-off states for learning diverse jumping strategies. As shown by the work we present in the last chapter, the take-off moment is a critical point of jumping motions, where the volume of feasible region of the dynamic skill is the smallest. In another word, bad initial states will fail fast, which in a way help our exploration framework to find good ones quicker. Alternatively, we could place the agent in a fixed initial pose to start with, such as a static pose before the run-up. This is problematic for several reasons. First, different jumping strategies need different length for the run-up. The planar position and facing direction of the root is still a three dimensional space to be explored. Second, the run-up strategies and the jumping strategies are not correlated in a one-to-one fashion. Visually, the run-up strategies do not look as diverse as the jumping strategies. Lastly, starting the jumps from a static pose lengthens the learning horizon, and makes our learning framework based on DRL even more costly. Therefore we choose to focus on just the jumping part of the jumps in this work, and leave the run-up control learning to DeepMimic, which is one of the state-of-the-art imitation-based DRL learning methods.

Actions The action a is a target pose described by internal joint rotations. We parameterize $1D$ revolute joint rotations by scalar angles, and $3D$ joint rotations by exponential maps [52]. Given a target pose and the current character state, joint torques are computed through a stable PD controller [210]. Our control frequencies $f_{control}$ ranges from $10Hz$ to $30Hz$ depending on both the task and the curriculum. For challenging tasks like high jumps, it helps to quickly improve the initial policies through stochastic evaluations in early training stages. A low-frequency policy enables faster learning by reducing the needed control steps, or in other words, the dimensionality and complexity of the actions $(a_0, a_1 ... a_T)$. This is in spirit similar to the $10Hz$ control fragments used in SAMCON-type controllers [123]. Successful low-frequency control policies can then be gradually transferred to high-frequency ones according to a curriculum to achieve finer controls and thus smoother motions. We discuss the choice of control frequency in more detail in Section 4.6.1.

Reward We use a reward function consisting of the product of two terms for all our strategy discovery tasks as follows:

$$r = r_{\text{task}} \cdot r_{\text{naturalness}} \tag{4.1}$$

where r_{task} is the task objective and $r_{\text{naturalness}}$ is a naturalness reward term computed from the P-VAE to be described in Section 4.4.2. For diverse strategy discovery, a simple r_{task}

which precisely captures the task objective is preferred. For example in high jumping, the agent receives a sparse reward signal at the end of the jump after it successfully clears the bar. In principle, we could transform the sparse reward into a dense reward to reduce the learning difficulty, such as to reward CoM positions higher than a parabolic trajectory estimated from the bar height. However in practice, such dense guidance reward can mislead the training to a bad local optimum, where the character learns to jump high in place rather than clears the bar in a coordinated fashion. Moreover, the CoM height and the bar height may not correlate in a simple way. For example, the CoM passes underneath the cross bar in Fosbuary flops. As a result, a shaped dense reward function could harm the diversity of the learned strategies. We will discuss reward function settings for each task in more details in Section 4.6.1.

Policy Representation We use a fully-connected neural network parameterized by weights θ to represent the control policy $\pi(a|s)$. Similar to the settings in [161], the network has two hidden layers with 1024 and 512 units respectively. ReLU activations are applied for all hidden units. Our policy maps a given state s to a Gaussian distribution over actions $a = \mathcal{N}(\mu(s), \Sigma)$. The mean $\mu(s)$ is determined by the network output. The covariance matrix $\Sigma = \sigma I$ is diagonal, where I is the identity matrix and σ is a scalar variable measuring the action noise. We apply an annealing strategy to linearly decrease σ from 0.5 to 0.1 in the first 1.0×10^7 simulation steps, to encourage more exploration in early training and more exploitation in late training.

Training We train our control policies with PPO [190]. PPO involves training both a policy network and a value function network. The value network architecture is similar to the policy network, except that there is only one single linear unit in the output layer. We train the value network with TD(λ) multi-step returns [207]. We estimate the advantage of the PPO policy gradient by the Generalized Advantage Estimator GAE(λ) [189].

4.4.2 Pose Variational Autoencoder

The dimension of natural human poses is usually much lower than the true degrees of freedom of the character model. We propose a generative model to produce natural PD target poses at each control step. More specifically, we train a Pose Variational Autorncoder (P-VAE) from captured natural human poses, and then sample its latent space to produce desired PD target poses for control. Here a pose only encodes internal joint rotations without global root transformations. We use publicly available human motion capture databses to train our P-VAE. Note that none of these databases consist of high jumps or obstacle jumps specifically, but they already provide enough poses for us to learn the natural human pose manifold already.

P-VAE Architecture and Training Our P-VAE adopts the standard Beta Variational Autoencoder (β-VAE) architecture [66]. The encoder maps an input feature x to a low-dimensional latent space parameterized by a Gaussian distribution with a mean μ_x and a diagonal convariance σ_x. The decoder maps a latent vector sampled from the Gaussian distribution back to the original feature space as x'. The training objective is to minimize the following loss function:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}}(x, x') + \beta \cdot \text{KL}(\mathcal{N}(\mu_x, \Sigma_x), \mathcal{N}(0, I)), \tag{4.2}$$

where the first term in the MSE reconstruction loss, and the second term shapes the latent variable distribution to a standard Gaussian distribution by measuring their Kullback-Leibler divergence. We set $\beta = 1.0 \times 10^{-5}$ in our experiments, so that the two terms in the loss function are within the same order of magnitude numerically.

We train the P-VAE on a dataset consisting of roughly 20000 poses obtained from the CMU and SFU motion capture databases [1, 2]. We include a large variety of motion skills, including walking, running, jumping, breakdancing, cartwheels, flips, kicks, martial arts, etc. The input features consists of all link and joint positions relative to the root in the local root frames, and all joint rotations with respect to their parents. We parameterize joint rotations by a $6D$ representations for better continuity as in [116, 275].

We model both the encoder and the decoder as fully-connected neural networks with two hidden layers, each having 256 units with tanh activation. We perform PCA on the training data and choose $d_{\text{latent}} = 13$ to cover 85% of the training data variance, where d_{latent} is the dimension of the latent variable. We use the Adam optimizer to update network weights [98], with the learning rate set to 1.0×10^{-4}. Using a mini-batch size of 128, the training takes 80 epochs within 2 minutes on an NVIDIA GeForce GTX 1080 GPU and an Intel i7-8700k CPU. We use this single pre-trained P-VAE for all our strategy discovery tasks to be described.

Composite PD Targets PD controllers provide actuation based on positional errors. So in order to reach the desired pose, the actual target pose needs to be offset by a certain amount. Such offsets are usually small to just counter-act the gravity for free limbs. However, for joints that interact with the environment, such as the lower body joints for weight support and ground take-off, large offsets are needed to generate powerful ground reaction forces to propel the body forward or into the air. Such complementary combined with the default P-VAE targets help realize natural poses during physics-based simulations. Our action space $\mathcal{A}$ is therefore $d_{\text{latent}} + d_{\text{offset}}$ dimensional, where d_{latent} is the dimension of the P-VAE latent space, and d_{offset} is the dimension of the DoFs that we wish to apply offsets for. We simply apply offsets to all internal joints in this work. Given $a = (a_{\text{latent}}, a_{\text{offset}}) \in \mathcal{A}$ sampled from the policy $\pi_\theta(a|s)$, where a_{latent} and a_{offset} correspond to the latent and offset

part of a respectively, the final PD target is computed by $D_{\text{pose}}(a_{\text{latent}})+a_{\text{offset}}$. Here $D_{\text{pose}}(\cdot)$ is a function that decodes the latent vector a_{latent} to full-body joint rotations. We minimize the usage of rotation offsets by a penalty term as follows:

$$r_{\text{naturalness}} = 1 - \text{Clip}\left(\left(\frac{||a_{\text{offset}}||_1}{c_{\text{offset}}}\right)^2, 0, 1\right), \tag{4.3}$$

where c_{offset} is the maximum offset allowed. For tasks with only a sparse reward signal at the end, $||a_{\text{offset}}||_1$ in Equation 4.3 is replaced by the average offset norm $\frac{1}{T}\sum_{t=0}^{T}||a_{\text{offset}}^{(t)}||_1$ across the entire episode. We use $L1$-norm rather than the commonly adopted $L2$-norm to encourage sparse solutions with fewer non-zero components [214, 22], as our goal is to only apply offsets to essential joints to complete the task while staying close to the natural pose manifold prescribed by the P-VAE.

4.5 Learning Diverse Strategies

Given a virtual environment and a task objective, we would like to discover as many strategies as possible to complete the task at hand. Without human insights and demonstrations, this is a challenging task. To this end, we propose a two-stage framework to enable stochastic DRL to discover solution modes such as the Fosbury flop.

The first stage focuses on strategy discovery by exploring the space of initial states. For example in high jump, the Fosbury flop technique and the straddle technique require completely different initial states at the take-off, in terms of the approaching angle with respect to the bar, the take-off velocities, and the choice of inner or outer leg as the take-off leg. A fixed initial state may lead to success of one particular strategy but can miss other drastically different ones. We systematically explore the initial state space through a novel sample-efficient Bayesian Diversity Search (BDS) algorithm to be described in Section 4.5.1.

The output of Stage 1 is a set of diverse motion strategies along with their corresponding initial states. Take such a successful initial state as input, we then apply another pass of DRL learning to further explore more motion variations permitted by the same initial state. The intuition is to explore different local optima while maximizing the novelty of the current policy, compared with previously found ones. We describe our detailed settings for the Stage 2 novel policy seeking algorithm in Section 4.5.2.

4.5.1 Stage 1: Initial States Exploration with Bayesian Diversity Search

In Stage 1, we perform diverse strategy discovery by exploring initial states variations, such as pose and velocity variations, at the take-off moment. We first extract a feature vector f from a motion trajectory to characterize and differentiate between different strategies. A straightforward way is to compute the Euclidean distance between time-aligned motion trajectories, but we hand pick a low-dimensional visually-salient feature set as detailed in

Section 4.6.1. We also define a low-dimensional exploration space $\mathcal{X}$ for initial states, as exploring the full state space is computationally prohibitive. Our goal is to search for a set of representations $X_n = \{x_1, x_2, ..., x_n | x_i \in \mathcal{X}\}$, such that the corresponding feature set $F_n = \{f_1, f_2, ..., f_n | f_i \in \mathcal{F}\}$ has a large diversity. Note that as DRL training and physics-based simulation are involved in producing the motion trajectories from an initial state, the computation of $f_i = g(x_i)$ is a stochastic and expensive black-box function. We therefore design a sample-efficient Bayesian Optimization (BO) algorithm to optimize for motion diversity in a guided fashion.

Our BDS algorithm iteratively selects the next sample to evaluate from $\mathcal{X}$, given the current set of observations $X_t = \{x_1, x_2, ..., x_t\}$ and $F_t = \{f_1, f_2, ..., f_t\}$. More specifically, the next point x_{t+1} is selected based on an acquisition function $a(x_{t+1})$ to maximize the diversity in $F_{t+1} = F_t \cup \{f_{t+1}\}$. We choose to maximize the minimum distance between f_{t+1} and all $f_i \in F_t$:

$$a(x_{t+1}) = \min_{f_i \in F_t} ||f_{t+1} - f_i||. \tag{4.4}$$

Since evaluating f_{t+1} through $g(\cdot)$ is expensive, we employ a surrogate model to quickly estimate f_{t+1}, so that the most promising sample to evaluate next can be efficiently found through Equation 4.4.

We maintain the surrogate statistical model of $g(\cdot)$ using a Gaussian Process [180], similar to standard BO methods. A GP contains a prior mean $m(x)$ encoding the prior belief of the function value and a kernel function $k(x, x')$ measuring the correlation between $g(x)$ and $g(x')$. More details of our specific $m(x)$ and $k(x, x')$ are given in Section 4.6.1. Hereafter we assume a one-dimensional feature space $\mathcal{F}$. Generalization to a multi-dimensional feature space is straightforward as multi-output Gaussian Process implementations are readily available, such as [223]. Given $m(\cdot)$, $k(\cdot, \cdot)$, and current observations $\{X_t, F_t\}$, posterior estimation of $g(x)$ for an arbitrary x is given by a Gaussian distribution with mean μ_t and variance σ_t^2 computed in closed forms:

$$\begin{aligned} \mu_t(x) &= k(x, X_t)(K + \eta^2 I)^{-1} (F_t - m(x)) + m(x), \\ \sigma_t^2(x) &= k(x, x) + \eta^2 - k(x, X_t)(K + \eta^2 I)^{-1} k(X_t, x), \end{aligned} \tag{4.5}$$

where $X_t \in \mathbb{R}^{t \times \dim(\mathcal{X})}$, $F_t \in \mathbb{R}^t$, $K \in \mathbb{R}^{t \times t}$, $K_{i,j} = k(x_i, x_j)$, and $k(X_t, x) = [k(x, x_1), ...k(x, x_t)]^T$. I is the identity matrix, and η is the standard deviation of the observation noise. Equation 4.4 can then be approximated by

$$\hat{a}(x_{t+1}) = \mathbb{E}_{\hat{f}_{t+1} \sim \mathcal{N}(\mu_t(x_{t+1}), \sigma_t^2(x_{t+1}))} \left[\min_{f_i \in F_t} ||\hat{f}_{t+1} - f_i|| \right]. \tag{4.6}$$

Equation 4.6 can be computed analytically for one-dimensional features, but gets more and more complicated to compute analytically as the feature dimension grows, or when

the feature space is non-Euclidean as in our case with rotational features. Therefore, we compute Equation 4.6 numerically with Monte-Carlo integration for simplicity.

The surrogate model is just an approximation to the true function, and has large uncertainty where observations are lacking. Rather than only maximizing the function value when picking the next sample, BO methods usually also take into account the estimated uncertainty to avoid being overly greedy. For example, GP-UCB (Gaussian Process Upper Confidence Bound), one of the most popular BO algorithms, adds a variance term into its acquisition function. Similarly, we could adopt a composite acquisition function as follows:

$$a'(x_{t+1}) = \hat{a}(x_{t+1}) + \beta\sigma_t(x_{t+1}), \tag{4.7}$$

where $\sigma_t(x_{t+1})$ is the heuristic term favoring candidates with large uncertainty, and β is a hyperparameter trading off exploration and exploitation (diversity optimization in our case). Theoretically well justified choice of β exists for GP-UCB, which guarantees optimization convergence with high probability [199]. However in our context, no such guarantees hold as we are not optimizing f but rather the diversity of f, the tuning of the hyperparameter β is thus not trivial, especially when the evaluation function $g(\cdot)$ is extremely costly. To mitigate this problem, we decouple these two terms and alternate between exploration and exploitation following a similar idea proposed in [197]. During exploration, our acquisition function becomes:

$$a_{\text{exp}}(x_{t+1}) = \sigma_t(x_{t+1}). \tag{4.8}$$

The sample with the largest posterior standard deviation is chosen as x_{t+1} to be evaluated next:

$$x_{t+1} = \arg\max_x \sigma_t(x). \tag{4.9}$$

Under the condition that $g(\cdot)$ is a sample from GP function distribution $\mathcal{GP}(m(\cdot), k(\cdot,\cdot))$, Equation 4.9 can be shown to maximize the Information Gain I on function $g(\cdot)$:

$$x_{t+1} = \arg\max_x I\left(X_t \cup \{x\}, F_t \cup \{g(x)\}; g\right), \tag{4.10}$$

where $I(A;B) = H(A) - H(A|B)$, and $H(\cdot) = \mathbb{E}\left[-\log p(\cdot)\right]$ is the Shannon entropy [30].

We summarize our BDS algorithm in Algorithm 1. The alternation of exploration and diversity optimization involves two extra hyperparameters N_{exp} and N_{opt}, corresponding to the number of samples allocated for exploration and diversity optimization in each round. Compared to β in Equation 4.7, N_{exp} and N_{opt} are much more intuitive to tune. We also found that empirically the algorithm performance is insensitive to the specific values of N_{exp} and N_{opt}. The exploitation stage directly maximizes the diversity of motion strategies. We optimize $\hat{a}(\cdot)$ with a sampling-based method DIRECT (Dividing Rectangle) [83], since derivative information may not be accurate in the presence of function noise due to the

Monte-Carlo integration. This optimization does not have to be perfectly accurate, since the surrogate model is an approximation in the first place. The exploration stage facilitates the discovery of diverse strategies by avoiding repeated queries on well-sampled regions. We optimize $a_{\text{exp}}(\cdot)$ using a simple gradient-based method L-BFGS [121].

Algorithm 1: Bayesian Diversity Search

Input: Strategy evaluation function $g(\cdot)$, exploration count N_{exp} and diversity optimization count N_{opt}, total sample count n.
Output: Initial states $X_n = \{x_1, x_2, ..., x_n\}$ for diverse strategies.
$t = 0$; $X_0 \leftarrow \varnothing$; $F_0 \leftarrow \varnothing$;
Initialize GP surrogate model with random samples;
while $t < n$ **do**
  if $t\%(N_{exp} + N_{opt}) < N_{exp}$ **then**
    $x_{t+1} \leftarrow \arg\max a_{\text{exp}}(\cdot)$ by L-BFGS; `// Equation 4.8`
  else
    $x_{t+1} \leftarrow \arg\max \hat{a}(\cdot)$ by DIRECT; `// Equation 4.6`
  end
  $f_{t+1} \leftarrow g(x_{t+1})$;
  $X_{t+1} \leftarrow X_t \cup \{x_{t+1}\}$; $F_{t+1} \leftarrow F_t \cup \{f_{t+1}\}$;
  Update GP surrogate model with X_{t+1}, F_{t+1}; `// Equation 4.5`
  $t \leftarrow t + 1$;
end
return X_n

4.5.2 Stage 2: Novel Policy Seeking

In Stage 2 of our diverse strategy discovery framework, we explore potential strategy variations given a fixed initial state discovered in Stage 1. Formally, given an initial state x and a set of discovered policies $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$, we aim to learn a new policy π_{n+1} which is different from all existing $\pi_i \in \Pi$. This can be achieved with an additional policy novelty reward to be jointly optimized with the task reward during DRL training. We measure the novelty of policy π_i with respect to π_j by their corresponding motion feature distance $||f_i - f_j||$. The novelty reward function is then given by

$$r_{\text{novelty}}(f) = \text{Clip}\left(\frac{\min_{\pi_i \in \Pi} ||f_i - f||}{d_{\text{threshold}}}, 0.01, 1\right), \tag{4.11}$$

which rewards simulation rollouts showing different strategies to the ones presented in the existing policy set. $d_{\text{threshold}}$ is a hyperparameter measuring the desired policy novelty to be learned next. Note that the feature representation f here in Stage 2 can be the same as or different from the one used in Stage 1 for initial states exploration.

Our novel policy seeking is in spirit similar to the idea of [206, 269]. However, there are two key differences. First, in machine learning, policy naively metrics have been designed and

validated only on low-dimensional control tasks. For example in [269], the policy novelty is measured by the reconstruction error between the states from the current rollout and previous rollouts encapsulated as a deep autoencoder. In our case of high-dimensional $3D$ character control tasks, however, novel state sequences do not necessarily correspond to novel motion strategies. We therefore opt to design discriminative strategy features whose distances are Incorporated into the DRL training reward.

Second, we multiply the novelty reward with the task reward as the training reward, and adopt a standard gradient-based method PPO to train the policy. Additional optimization techniques are not required to for learning novel strategies, such as the Task-Novelty Bisector method proposed in [269] that modifies the policy gradients to encourage novelty learning. Our novelty policy seeking procedure always discovers novel policies since the character is forced to perform a different strategy. However, the novel policies may exhibit unnatural and awkward movements, when the given initial state is not capable of multiple natural strategies.

4.6 Task Setup and Implementation

We demonstrate diverse strategy discovery for two challenging motor tasks: high jumping and obstacle jumping. We also tackle several variations of these tasks. We describe task specific settings in Section 4.6.1, and implementation details in Section 4.6.2.

4.6.1 Task Setup

The high jump task follows the Olympics rules, where the simulated athlete takes off with one leg, clears the crossbar, and lands on a crash mat. We model the landing area as a rigid block for simplicity. The crossbar is modeled as a rigid wall vertically extending from the ground to target heights to prevent the character from cheating during early training, i.e., passing through beneath the bar. A rollout is considered successful and terminated when the character lands on the rigid box with all body parts at least $20cm$ away from the wall. A rollout is considered as a failure and terminated immediately, if any body part touches the wall, or any body part other than the take-off foot touches the ground, or if the jump does not succeed within two seconds after the take-off.

The obstacle jump shares most of the settings of the high jump. The character takes off with one leg, clears a box-shaped obstacle of $50cm$ in height with variable widths, then lands on a crash mat. The character is required to complete the task within two seconds as well, and not allowed to touch the obstacle with any body part.

Run-up Learning

A full high jump or obstacle jump consists of three phases: run-up, take-off and landing. Our framework described so far can discover good initial states at take-off that lead to

Task	$z_{\min}$(cm)	$z_{\max}$(cm)	Δz(cm)	R_T
High jump	50	200	1	30
Obstacle jump	5	250	5	50

Table 4.1: Curriculum parameters for learning jumping tasks. z parameterizes the task difficulty, i.e., the crossbar height in high jumps and the obstacle width in obstacle jumps. $z_{\min}$ and $z_{\max}$ specify the range of z, and Δz is the increment when moving to a higher difficulty level. R_T is the accumulated reward threshold to move on to the next curriculum difficulty.

diverse jumping strategies. What is lacking is the matching run-up control policies that can prepare the character to reach these good initial states at the end of the run. We train the run-up controllers with DeepMimic, where DRL learning reward consists of a task reward and an imitation reward. The task reward encourages the end state of the run-up to match the desired take-off state of the jump. The imitation reward guides the simulation to match the style of the reference run. We use a curved sprint as the reference run-up for high jump, and a straight sprint for the obstacle jump run-up. For high jump, the explored initial state space is four-dimensional: the desired approach angle α, the X and Z component of the root angular velocity ω, and the magnitude of the Z component of the root linear velocity v_z in a facing-direction invariant frame. We fix the desired root Y angular velocity to $3rad/s$, which is taken from the reference curved sprint. In summary, the task reward r_G for the run-up control of a high jump is defined as:

$$r_{\mathrm{G}} = \exp\left(-\frac{1}{3} \cdot ||\omega - \bar{\omega}||_1 - 0.7 \cdot (v_z - \bar{v}_z)^2\right), \tag{4.12}$$

where $\bar{\omega}$ and $\bar{v}_z$ are the corresponding targets for ω and v_z. α does not appear in the reward function as we simply rotate the high jump suite in the environment to realize different approach angles. For the obstacle jump, we explore a three-dimensional take-off state space consisting of the root angular velocities along all axes. Therefore the run-up control task reward r_{G} is given by

$$r_{\mathrm{G}} = \exp(-\frac{1}{3} \cdot ||\omega - \bar{\omega}||_1). \tag{4.13}$$

Reward Function

We use the same reward function structure for both high jumps and obstacle jumps, where the character gets a sparse reward only when it successfully completes the task. The full reward function is defined as in Equation 4.1 for Stage 1. For Stage 2, the novelty bonus r_{novelty} as discussed in Section 4.5.2 is added:

$$r = r_{\mathrm{task}} \cdot r_{\mathrm{naturalness}} \cdot r_{\mathrm{novelty}}. \tag{4.14}$$

Parameter	Simulated Athlete	Mocap Athlete
Weight (kg)	60	70
Height (cm)	170	191
hip height (cm)	95	107
knee height (cm)	46	54

Table 4.2: Model parameters of our virtual athlete and the mocap athlete.

$r_{\text{naturalness}}$ is the motion naturalness term discussed in Section 4.4.2. For both stages, the task reward consists of three terms:

$$r_{\text{task}} = r_{\text{complete}} \cdot r_{\omega} \cdot r_{\text{safety}}. \tag{4.15}$$

r_{complete} is a binary reward precisely corresponding to task completion. $r_{\omega} = \exp(-0.02||\omega||)$ penalizes excessive root rotations where $||\omega||$ is the average magnitude of the root angular velocities across the episode. r_{safety} is a term to penalize unsafe head-first landings. We set it to 0.7 for unsafe landings and 1.0 otherwise. r_{safety} can also be further engineered to generate more landing styles, such as a landing on feet as shown in Figure 4.14.

Curriculum and Scheduling

The high jump is a challenging motor skill that requires years of training even for professional athletes. We therefore adopt curriculum-based learning to gradually increase the task difficulty z, defined as the crossbar height in high jumps or the obstacle width in obstacle jumps. Detailed curriculum settings are given in Table 4.1, where $z_{\min}$ and $z_{\max}$ specify the range of z, and Δz is the increment when moving to a higher difficulty level.

We adaptively schedule the curriculum to increase the task difficulty according to the DRL training performance. At each training iteration, the average sample reward is added to a reward accumulator. We increase z by Δz whenever the accumulated reward exceeds a threshold R_T, and then reset the reward accumulator. Detailed settings for Δz and R_T are listed in Table 4.1. The curriculum could also be scheduled following a simpler scheme adopted in [245], where task difficulty is increased when the average sample reward in each iteration exceeds a threshold. We found that for athletic motions, such average sample reward threshold is hard to define uniformly for different strategies in different training stages.

Throughout training, the control frequency f_{control} and the P-VAE offset penalty coefficient c_{offset} in Equation 4.3 are also scheduled according to the task difficulty, in order to encourage exploration and accelerate training in early stages. We set $f_{\text{control}} = 10 + 20 \cdot \text{Clip}(\rho, 0, 1)$ and $c_{\text{offset}} = 48 - 33 \cdot \text{Clip}(\rho, 0, 1)$, where $\rho = 2z - 1$ for high jumps and $\rho = z$ for obstacle jumps. We find that in practice the training performance does not depend sensitively on these hyperparameters.

Strategy Features

We choose low-dimensional and visually discriminative features f of learned strategies for effective diversity measurement of discovered strategies. In the sports literature, high jump techniques are usually characterized by the body orientation when the athlete clears the bar at his peak height. The rest of the body limbs are then coordinated in an optimal way to clear the bar as high as possible. Therefore we use the root orientation when the character's CoM lies in the vertical crossbar plane as f. This three-dimensional root orientation serves well as a Stage 2 feature for high jumps. For Stage 1, this feature can be further reduced to one dimension, as we will shown in Section 4.7. More specifically, we only measure the angle between the character's root direction and the global up vector, which corresponds to whether the character clears the bar facing upward or downward. Such a feature does not require a non-Euclidean GP output space that we need to handle in Stage 1. We use the same set of features for obstacle jumps, except that root orientations are measured when the character's CoM lies in the center vertical plane of the obstacle.

Note that it is not necessary to train to completion, i.e., the maximum task difficulty, to evaluate the feature diversity, since the overall jumping strategy usually remains unchanged after a given level of difficulty, which we denote by z_{freeze}. Based on empirical observations, we terminate the training after reaching $z_{\text{freeze}} = 100cm$ for both high jump and obstacle jump tasks for strategy discovery.

GP Priors and Kernels

We set GP prior $m(\cdot)$ and kernel $k(\cdot, \cdot)$ for BDS based on common practices in the Bayesian optimization literature. Without any knowledge on the strategy feature distribution, we set the prior mean $m(\cdot)$ to be the mean of the value range of a feature. Among the many common choices for kernel functions, we adopt the Matérn$^5/_2$ kernel [135], defined as:

$$k_{5/2}(x, x') = \theta(1 + \sqrt{5}d_\Lambda(x, x') + \frac{5}{3}d_\Lambda^2(x, x'))e^{-\sqrt{5}d_\Lambda(x,x')} \tag{4.16}$$

where θ and λ are learnable parameters of the GP. $d_\Lambda(x, x') = \sqrt{(x - x')^T \text{diag}(\Lambda)(x - x')}$ is the Mahalanobis distance.

4.6.2 Implementation

We implemented our system in PyTorch [159] and PyBullet [29]. The simulated athlete has 28 internal DoFs and 34 DoFs in total. We run the simulation at 600 Hz. Torque limits for the hips, knees and ankles are taken from Biomechanics estimations for a human athlete performing a Fosbury flop [150]. Torque limits for other joints are kept the same as [161]. Joint angle limits are implemented by penalty forces. We captured three standard high

jumps from a university athlete, whose body measurements are given in Table 4.2. For comparison, we also list these measurements for our virtual athlete.

For DRL training, we set $\lambda = 0.95$ for both TD(λ) and GAE(λ). We set the discount factor $\gamma = 1.0$ since our tasks have short horizon and sparse rewards. The PPO clip threshold is set to 0.02. The learning rate is 2.5×10^{-5} for the policy network and 1.0×10^{-2} for the value network. In each training iteration, we sample 4096 state-action tuples in parallel and perform five policy updates with a mini-batch size of 256. For Stage 1 diverse strategy discovery, we implement BDS using GPFlow [223] with both N_{exp} and N_{opt} set to three. $d_{\text{threshold}}$ in Stage 2 novel policy seeking is set to $\pi/2$. Our experiments are performed on a Dell Precision 7920 Tower workstation, with dual Intel Xeon Gold 6248R CPUs (3.0 GHz, 48 cores) and an Nvidia Quadro RTX 6000 GPU. One strategy evaluation for a single initial state, i.e. Line 9 in Algorithm 1, typically takes about six hours. Network updates are performed on the GPU.

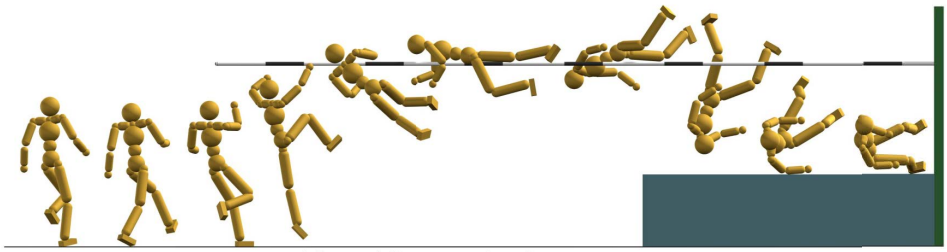
4.7 Results

We demonstrate multiple strategies discovered through our framework for high jumping and obstacle jumping in Section 4.7. We validate the effectiveness of BDS and P-VAE in Section 4.7.1. Comparison with motion capture examples, and interesting variations of learned policies are given in Section 4.7.2.

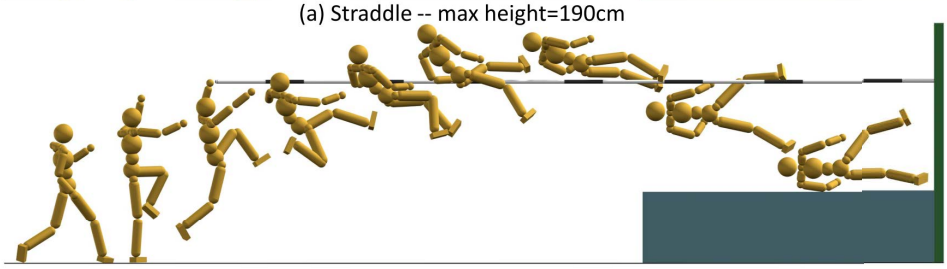
High Jumps

In our experiments, six different high jump strategies are discovered during the Stage 1 initial state exploration within the first 10 BDS samples: Fosbury Flop, Western Roll (facing-up), Straddle, Front Kick, Side Jump, Side Dive. The first three are high jump techniques standard in the sports literature. The last three strategies are not commonly used in sporting events, but still physically valid so we name them according to their visual characteristics. The other four samples generated either repetitions or failures. Strategy repetition are generally not avoidable due to model errors and large flat regions in the motion space. Since the evaluation of one BDS sample takes about six hours, the Stage 1 exploration takes about 60 hour in total. The discovered distinct strategies at $z_{\text{freeze}} = 100cm$ are further optimized separately to reach their maximum difficulty level, which takes another 20 hours. Representative take-off state feature values of the discovered strategies can be found in Table 4.3.

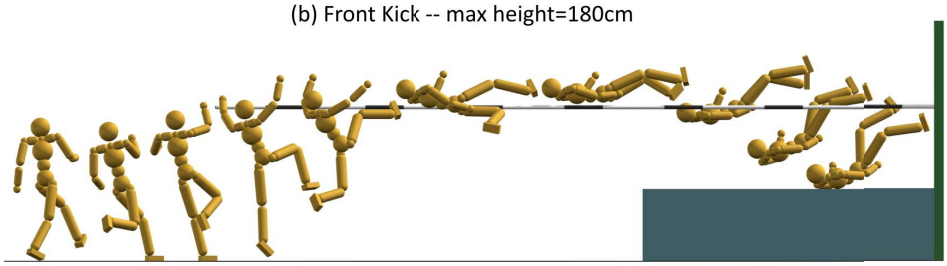
In Stage 2, we perform novel policy search for five DRL iterations from each good initial states of Stage 1. Training is warm started with the associated Stage 1 policy for efficient learning. The total time required for Stage 2 is roughly 60 hours. More strategies are discovered in Stage 2, but most are repetitions and only two of them are novel strategies not discovered in Stage 1: Western Roll (facing sideways) and Scissor Kick. Western Roll



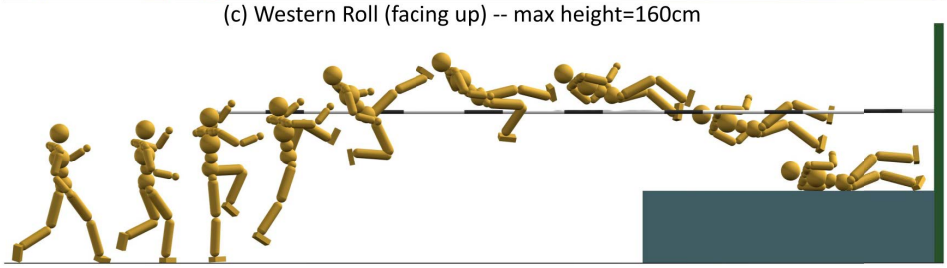
(a) Straddle -- max height=190cm



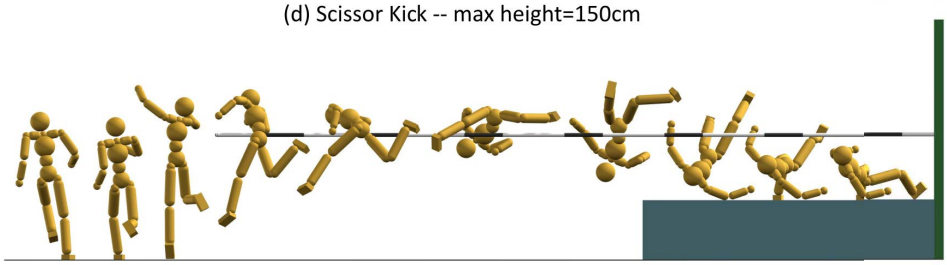
(b) Front Kick -- max height=180cm



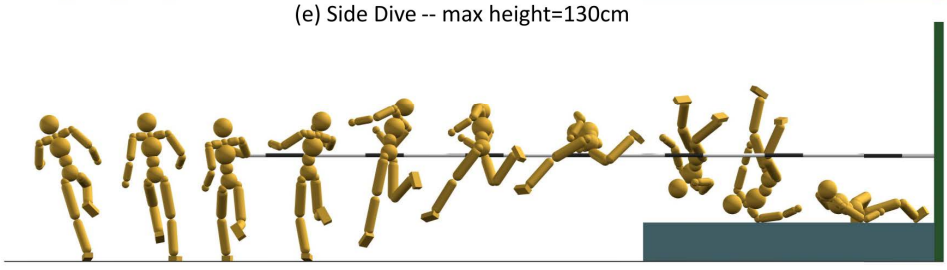
(c) Western Roll (facing up) -- max height=160cm



(d) Scissor Kick -- max height=150cm



(e) Side Dive -- max height=130cm



(f) Side Jump -- max height=110cm

Figure 4.3: Six of the eight high jump strategies discovered by our learning framework.

Strategy	v_z	ω_x	ω_z	α
Fosbury Flop	-2.40	-3.00	1.00	-0.05
Western Roll (up)	-0.50	1.00	-1.00	2.09
Straddle	-2.21	1.00	0.88	1.65
Front Kick	-0.52	1.00	-0.26	0.45
Side Dive	-1.83	-2.78	-0.32	1.18
Side Jump	-1.99	-1.44	0.44	0.70

Table 4.3: Representative take-off state features for discovered high jumps.

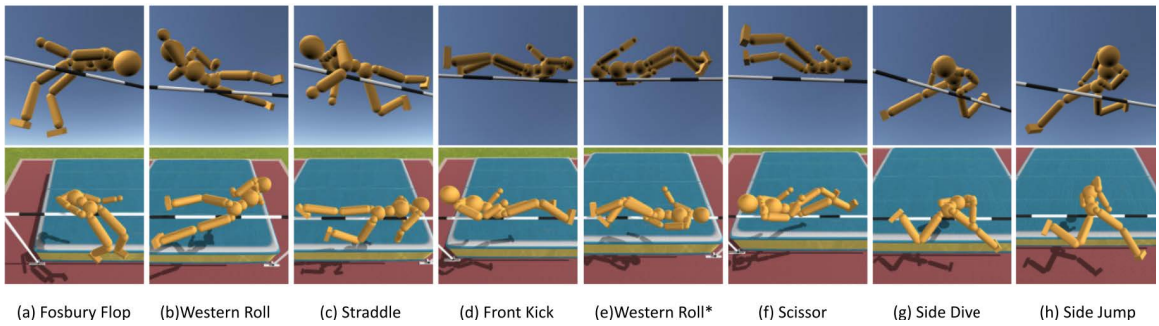


Figure 4.4: Peak poses of discovered high jump strategies, ordered by their maximum cleared height. First row: look-up views; Second row: look-down views.

(facing sideways) shares the same initial state with the Western Roll (up). Scissor Kick shares the same initial state with the Front Kick. The strategies discovered in each Stage are summarized in Figure 4.7. We visualize all eight distinct strategies in Figure 4.1 and Figure 4.3. We also visualize their peak poses in Figure 4.4.

Obstacle Jumps

Figure 4.5 shows the six different obstacle jump strategies discovered in Stage 1 within the first 17 BDS samples: Front Kick, Side Kick, Twist Jump (clockwise), Twist Jump (counterclockwise), Straddle and Dive Turn. More strategies are discovered in Stage 2, but only two of them are novel as shown in Figure 4.6: Western Roll and Twist Turn. Western Roll shares the initial state with Twist Jump (clockwise). Twist Turn shares the initial state with Dive Turn. The two stages together take about 180 hours.

Although our obstacle jump task is not an Olympic event, it is analogous to a long jump in that it seeks to jump a maximum-length jumped. Setting the obstacle height to zeros yields a standard long jump task. The framework discovers several strategies, including one similar to the standard long jump adopted in competitive events, with the strong caveat that the distance achieved is limited by the speed of the run-up.

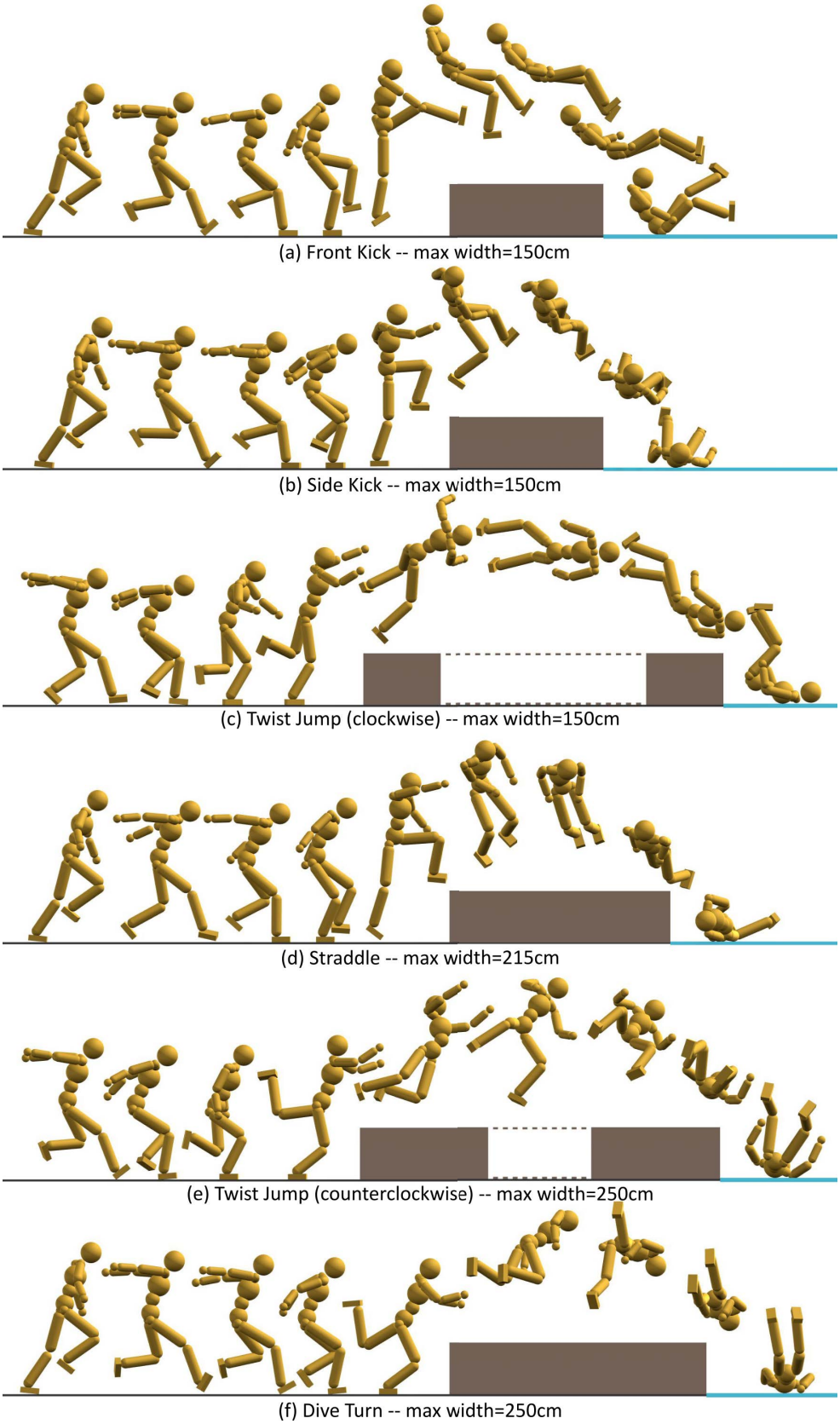
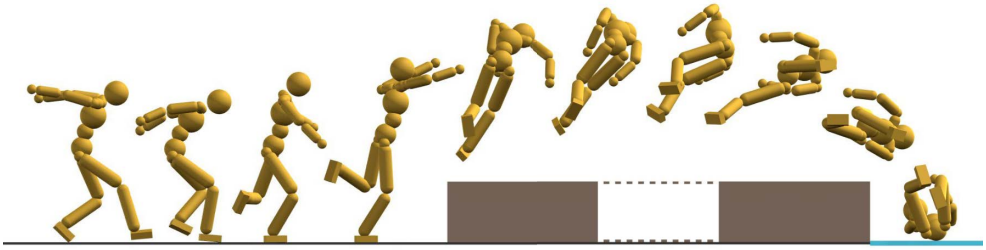
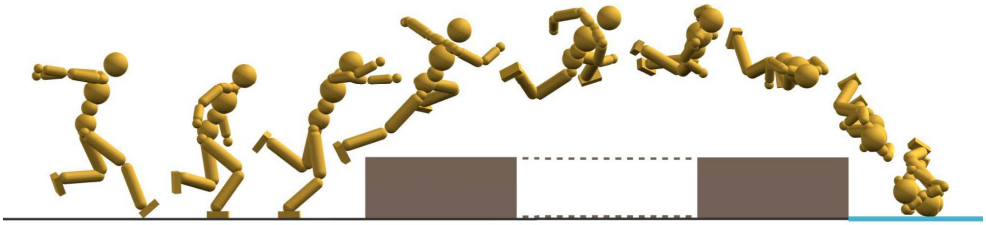


Figure 4.5: Six obstacle jump strategies discovered by our learning framework in Stage 1.



(a) Twist Turn -- max width=250cm



(b) Western Roll -- max width=250cm

Figure 4.6: Two obstacle jump strategies discovered in Stage 2 of our learning framework.

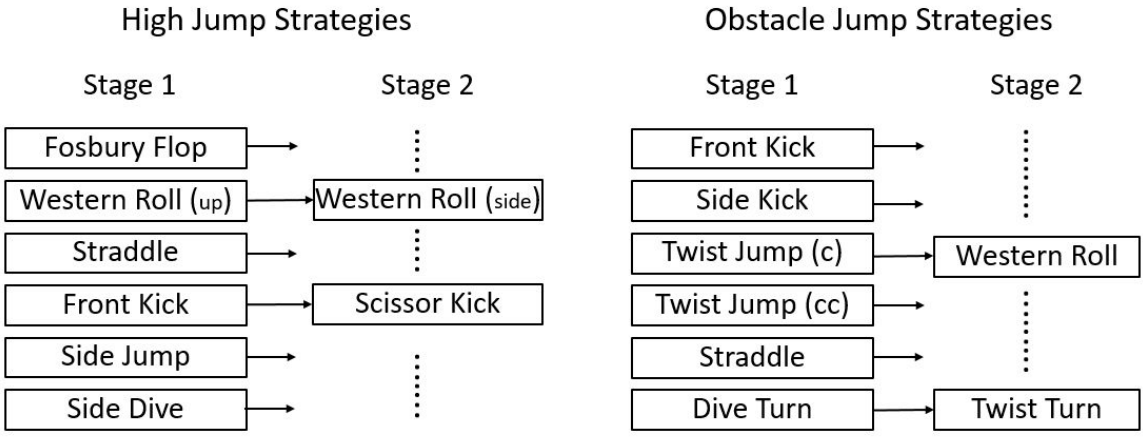


Figure 4.7: Diverse strategies discovered in each stage of our framework.

Strategy	ω_x	ω_y	ω_z
Front Kick	1.15	-1.11	3.89
Side Kick	3.00	3.00	-2.00
Twist Jump (c)	-1.50	1.50	-2.00
Straddle	0.00	0.00	1.00
Twist Jump (cc)	-2.67	0.00	-1.44
Dive Turn	-0.74	-2.15	-0.41

Table 4.4: Representative take-off state features for discovered obstacle jumps.

4.7.1 Validation and Ablation Study

BDS vs. Random Search

We validate the sample efficiency of BDS compared with a random search baseline. Within the first ten samples of initial states exploration in the high jump task, BDS discovered six distinct strategies as discussed in Section 4.7. Given the same computational budget, random search only discovered three distinct strategies: Straddle, Side Jump, and one strategy similar to Scissor Kick. Most samples result in repetitions of these three strategies, due to the presence of large flat regions in the strategy space where different initial states lead to the same strategy. In contrast, BDS largely avoids sampling the flat regions thanks to the acquisition function for diversity optimization and guided exploration of the surrogate model.

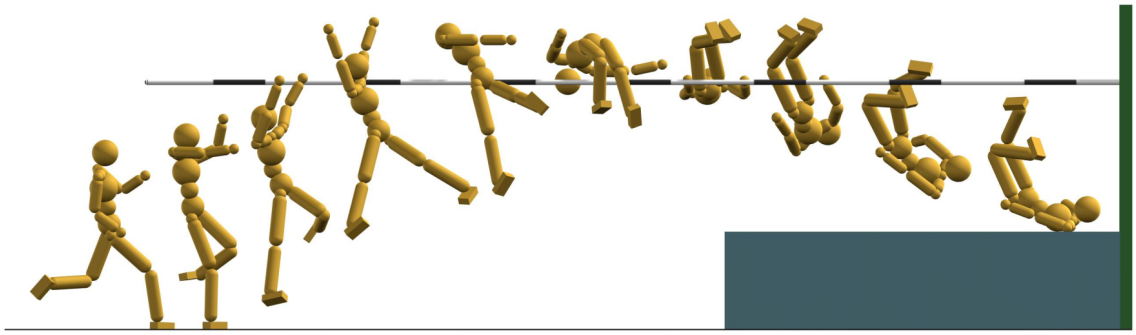
Motion Quality with/without P-VAE

We justify the usage of P-VAE for improving motion naturalness with results shown in Figure 4.8. Without P-VAE, the character can still learn physically valid skills to complete the tasks successfully, but the resulting motions usually exhibit unnatural behaviors. In the absence of a natural action space constrained by the P-VAE, the character can freely explore any arbitrary pose during the course of the motion to complete the task, which is unlikely to be within the natural pose manifold all the time.

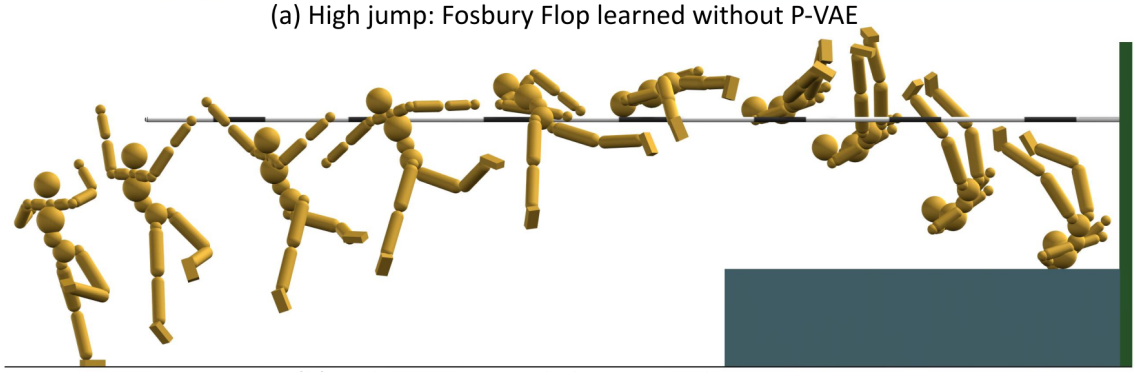
4.7.2 Comparison and Variations

Synthesized High Jumps vs. Motion Capture

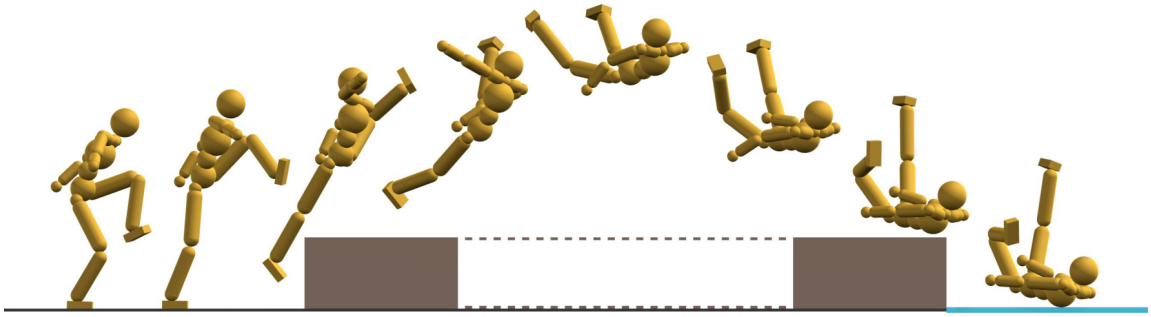
We capture motion capture examples from a university athlete in a commercial motion capture studio for three well-known high jump strategies: Scissor Kick, Straddle, and Fosbury Flop. We retarget the mocap examples onto our simulated characters, which is shorter than the real athlete as shown in Table 4.2. We visualize keyframes sampled from our simulated jumps and the retargted jumps in Figure 4.9 and Figure 4.10. Note that the bar heights are set to the maximal heights achieved by our trained control policies, while the bar heights for the mocap examples are just the bar heights used at the actual capture session.We did not



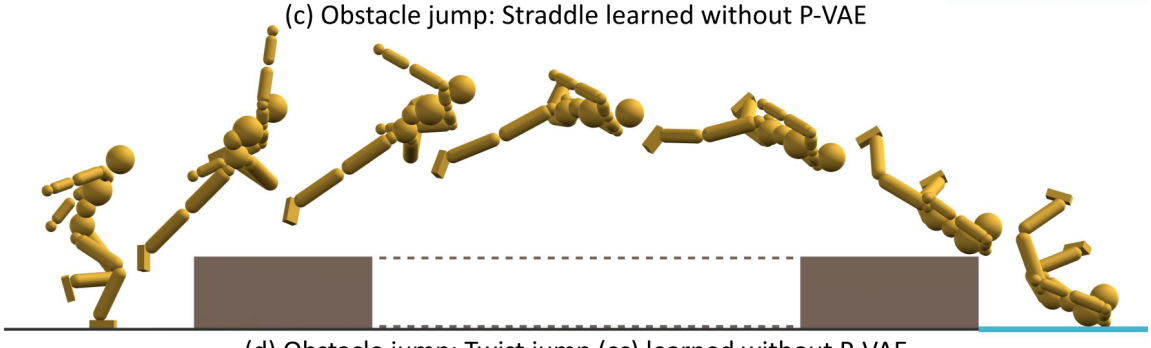
(a) High jump: Fosbury Flop learned without P-VAE



(b) High jump: Straddle learned without P-VAE



(c) Obstacle jump: Straddle learned without P-VAE



(d) Obstacle jump: Twist jump (cc) learned without P-VAE

Figure 4.8: Jumping strategies learned without P-VAE. Although the character can still complete the tasks, the poses are less natural.

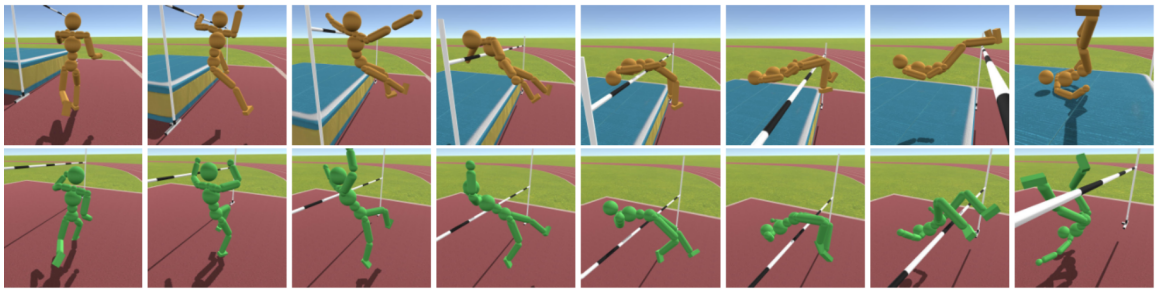


Figure 4.9: Fosbury Flop. First row: synthesized – max height=200cm; Second row: motion capture – capture height=130cm.

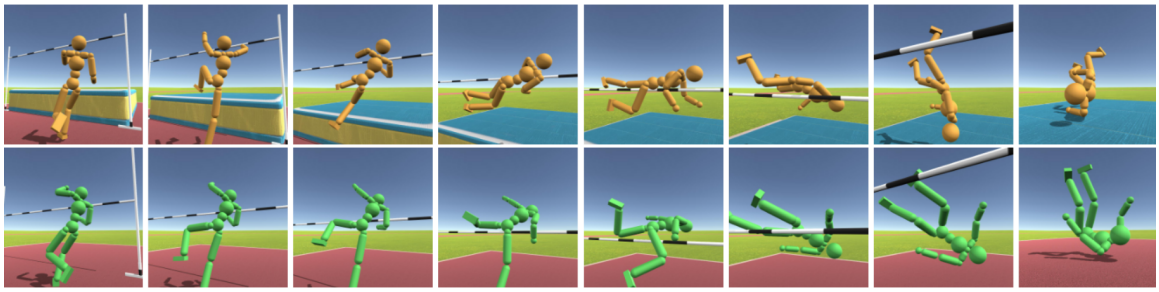


Figure 4.10: Straddle. First row: synthesized – max height=195cm; Second row: motion capture – capture height=130cm.

set the mocap bar heights at the athlete's personal record height, as we wanted to ensure his safety and comfort while jumping in a tight mocap suit with a lot of markers on.

High Jump Variations

In addition to discovering multiple motion strategies, our framework can easily support physically valid motion variations. We show four high jump variations generated from our framework in Figure 4.11, 4.12, 4.13 and 4.14. We generate the first three variations by taking the initial state of the Fosbury Flop strategy discovered in Stage 1, and retrain the jumping policy with additional constraints starting from a random initial policy. Figure 4.11 shows a jump with a weaker take-off leg, where the torque limits are reduced to 60% of its original values. Figure 4.12 shows a character jumping with a spine that does not permit backward arching. Figure 4.13 shows a character jumping with a fixed knee joint. All these variations clear lower maximum heights, and are visually different from the original Fosbury Flop in Figure 4.1 (a). For the jump in Figure 4.14, we take the initial state of the Front Kick, and train with an additional constraint that requires landing on feet. In Figure 4.15 we also show a high jump trained on Mars, where the gravity $g = 3.711m/s^2$ is lower, from the initial state of the Fosbury flop discovered on Earth.

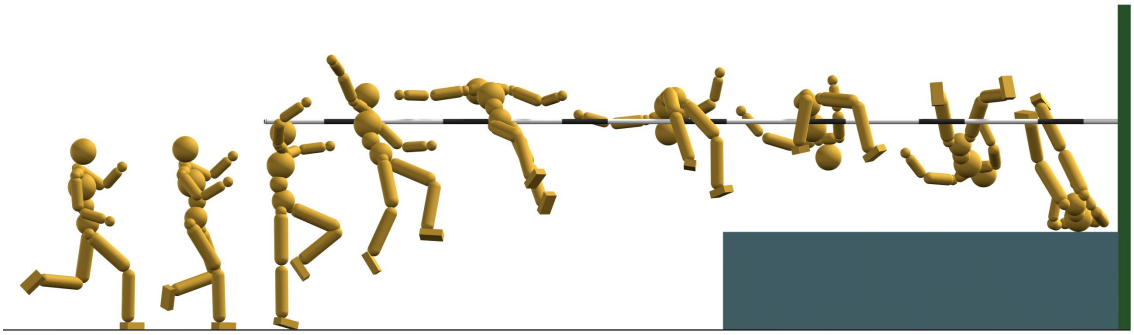


Figure 4.11: Fosbury Flop – max height=160cm, performed by a character with a weaker take-off leg, whose take-off hip, knee and ankle torque limits are set to 60% of the normal values.

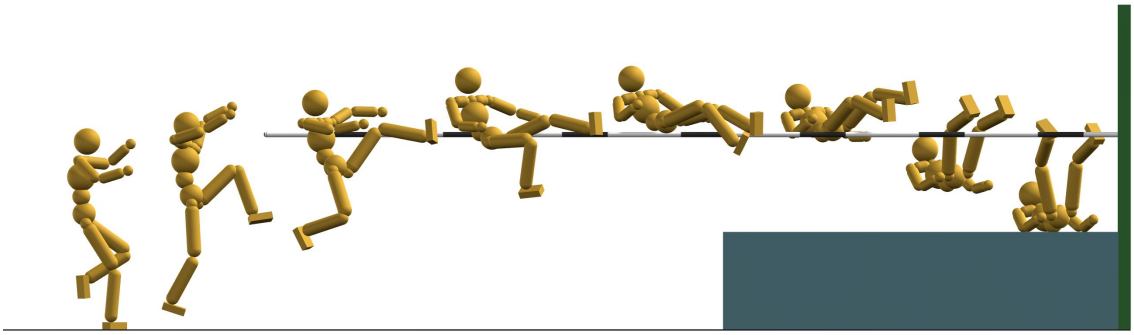


Figure 4.12: Fosbury Flop – max height=150cm, performed by a character with an inflexible spine that does not permit arching backwards.

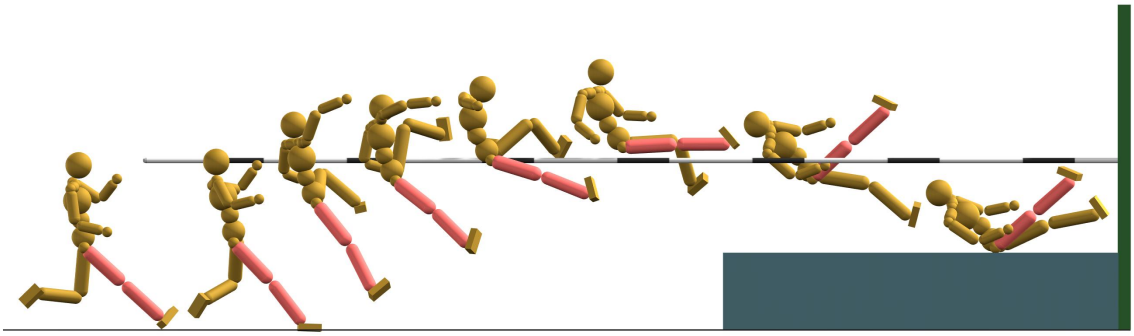


Figure 4.13: Front Kick – max height=120cm, performed with an additional constraint requiring landing on feet.

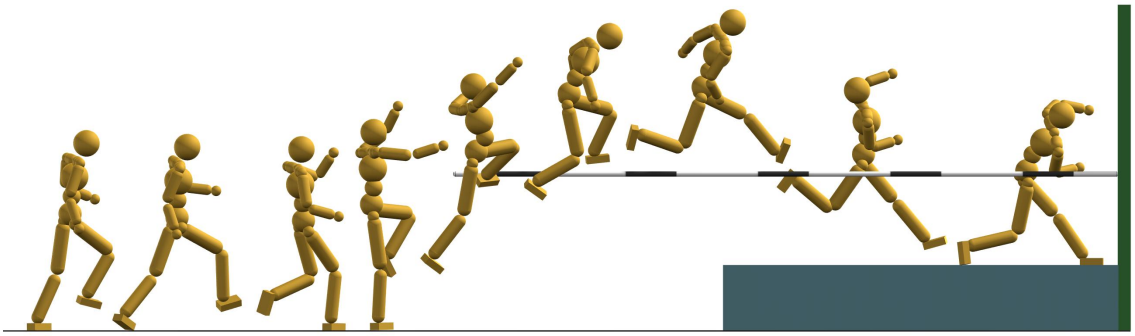


Figure 4.14: High jump variations. The first three policies are trained from the initial state of the Fosbury Flop discovered in Stage 1, and the last policy is trained from the initial state of the Front Kick discovered in Stage 1.

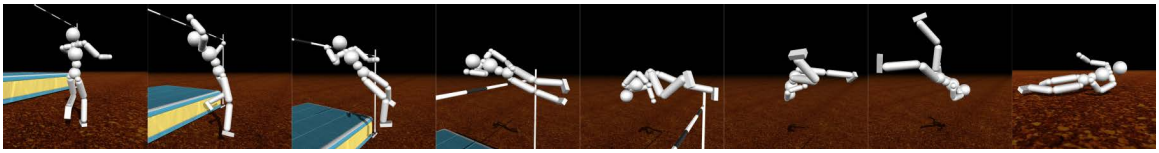


Figure 4.15: High jump policy trained on Mars with a lower gravity ($g = 3.711 m/s^2$), given the initial state of the Fosbury Flop discovered on Earth.

4.8 Conclusion

We have presented a framework for discovering and learning multiple natural and distinct strategies for highly challenging athletic jumping motions. A key insight is to explore the take-off state, which is a strong determinant of the jumping strategy that follows once airborne. In a second phase, we additionally use explicit rewards for novel motions. Another crucial aspect is to constrain the action space inside the natural human pose manifold. With the proposed two-stage framework and the pose variational autoencoder, natural and physically-nuanced jumping skills emerge automatically without any reference to human demonstrations. Within the proposed framework, the take-off state exploration is specific to jumping tasks, while the diversity search algorithm in both stages and the P-VAE are task agnostic. We leave further adaption of the proposed framework to additional motor skills as future work. We believe this work demonstrate a significant advance by being able to learn a highly-technical skill such as high-jumping.

We note that the current world record for men's high jump as of year 2022 is $245cm$, set in year 1993 by an athlete $170cm$ tall. The performance of realism of our simulated jumps are bounded by many simplifications in our modeling and simulations. We simplify the athlete's feet and specialized high jump shoes as rigid rectangle boxes, which reduces he maximum heights the virtual athlete can clear. We model the high jump crossbar as a wall at training time and as a rigid bar at run time, while real bars are made from more elastic materials such as fiberglasses. We use a rigid box as the landing surface, while real-world

landing cushions protect the athlete from breaking his neck and back, and also help him roll and get up in a fluid fashion.

The run-up phase of both jump tasks imitates reference motions, one single curved run for all the high jumps and one single straight run for all the obstacle jumps. The quality of the two reference runs affects the quality of not only the run-up controllers, but also the learned jump controller. This is because the jump controllers couple with the run-up controller through the take-off states, for which we only explore a low-dimensional feature space. The remaining dimensions of the take-off states stay the same as in the original reference run, As a result, the run-up controllers for our obstacle jumps remain in medium speed, and the swing leg has to kick backward sometimes in order for the body to dive forward. If we were to use a faster sprint with more forward leaning as the reference run, the discovered jumps could potentially be more natural and more capable to clear wider obstacles. Similarly, we did not find the Hurdle strategy for the high jumping, likely due to the reference run being curved rather then straight. In both references, there is a low in-place-jump after the last running step, which helped both jump controllers to jump up naturally. Using reference runs that anticipate the intended skills is definitely recommended, although retraining the run-up controller and the jump controller together in a post-processing stage may be helpful as well.

We were able to discover most well-known high-jump strategies, and some lesser-known variations. There remains a rich space of further parameters to consider for optimization, with our current choices being a good fit for our available computational budget. It would be exciting to discover a better strategy than the Fosbury flop, but a better strategy may not exist. We note that Stage 1 can discover most of the strategies shown in Figure 4.7. Stage 2 is only used to search for additional unique strategies and not to finetune the strategies already learned in Stage 1. We also experimented with simply running Stage 1 longer with three times more samples for the BDS. However, this could not discover any new strategies that can be discovered by Stage 2. In summary, Stage 2 is not absolutely necessary for our framework to work, but it complements Stage 1 in terms of discovering additional visually distinctive strategies. We also note that our Stage 2 search for novel policies is similar in spirit to the algorithm proposed in [269]. An advantage of our approach is its simplicity and the demonstration of its scalability to the discovery of visually distinct strategies for athletic skills.

There are many exciting directions for future investigations. First, we have only focused on strategy discovery for the take-off and airborne parts of jumping tasks. For landing, we only required not to land on head first. We did not model get-ups at all. How to seamlessly incoperate landing and get-ups into our framework is a worthy problem for future studies. Integrating the learning method in [212] with our framework might be a solution to this problem. Second, there is still room to further improve the motion quality of our synthesized motions. The P-VAE only constrains naturalness at a pose level, while ideally we need a

mechanism to guarantee naturalness on a motion level. This is especially helpful for under-constrained motor tasks such as crawling, where feasible regions of the tasks are larger and system dynamics cannot help prune a large portion of the state spaces as for the jumping tasks. Recently [170, 168] used a discriminator that grades the naturalness of two consecutive simulated motion frames to learn motion skills. It is possible to combine their framework and ours to further improve the motion quality. Lastly, our strategy discovery is computationally expensive. We are only able to explore initial states in a four-dimensional space, limited by our computational resources. If more dimensions could be explored, more strategies might be discovered. Parallel implementations is trivial for Stage 2 since searches for novel policies for different initial states are independent. For Stage 1, batched BDS where multiple points are queried together, similar to the idea of [13], might worth exploring. The key challenge of such an approach is how to find a set of good candidates to query simultaneously.

Chapter 5

Diverse Motor Skills Discovery for Hand Tool Manipulations

In chapter 4 we present a learning and control system capable of discovering diverse jumping skills for simulated humanoid characters. Simulated characters equipped with excellent locomotion and jumping skills can traverse and navigate in complex environments. However, in order to create artificial agents that replicate diverse human-like motor behaviors, we need to further equip simulated characters with hand object manipulation skills to help them interact with the surrounding environments, such as picking up different kinds of objects. In this chapter, we focus on one type of manipulation skill: hand tool manipulation skills, where simulated agents are asked to manipulate tools using their hands. Learning dexterous hand tool manipulation skills for simulated agents is a long-standing challenge in computer graphics and robotics. From computational perspectives, hand tool manipulation tasks are challenging for three main reasons: first, the high DoFs of anthropomorphic hands, which leads to high-dimensional sampling space for policy learning. Second, the tools are under-actuated, which means that the hands must apply delicate and accurate contact forces to the tools to manipulate them successfully. Lastly, the highly non-linear dynamics of manipulations involve the complex interplay between hands, tools and objects and make the objective function for policy learning highly discontinuous. Compared with locomotion skills learning tasks, hand tool manipulation tasks are less investigated in computer graphics, which might also reflect the intrinsic difficulty of the problem. This task can be more challenging when motion capture data or human demonstrations are not available.

In this chapter, we describe how we help simulated characters learn to use one common but challenging tool: chopsticks. More specifically, we design a learning system that combines Bayesian Optimization and deep reinforcement learning to discover multiple ways to hold and use chopsticks in the absence of motion capture data. We demonstrate our framework by relocating objects using chopsticks of different sizes and shapes, in different gripping

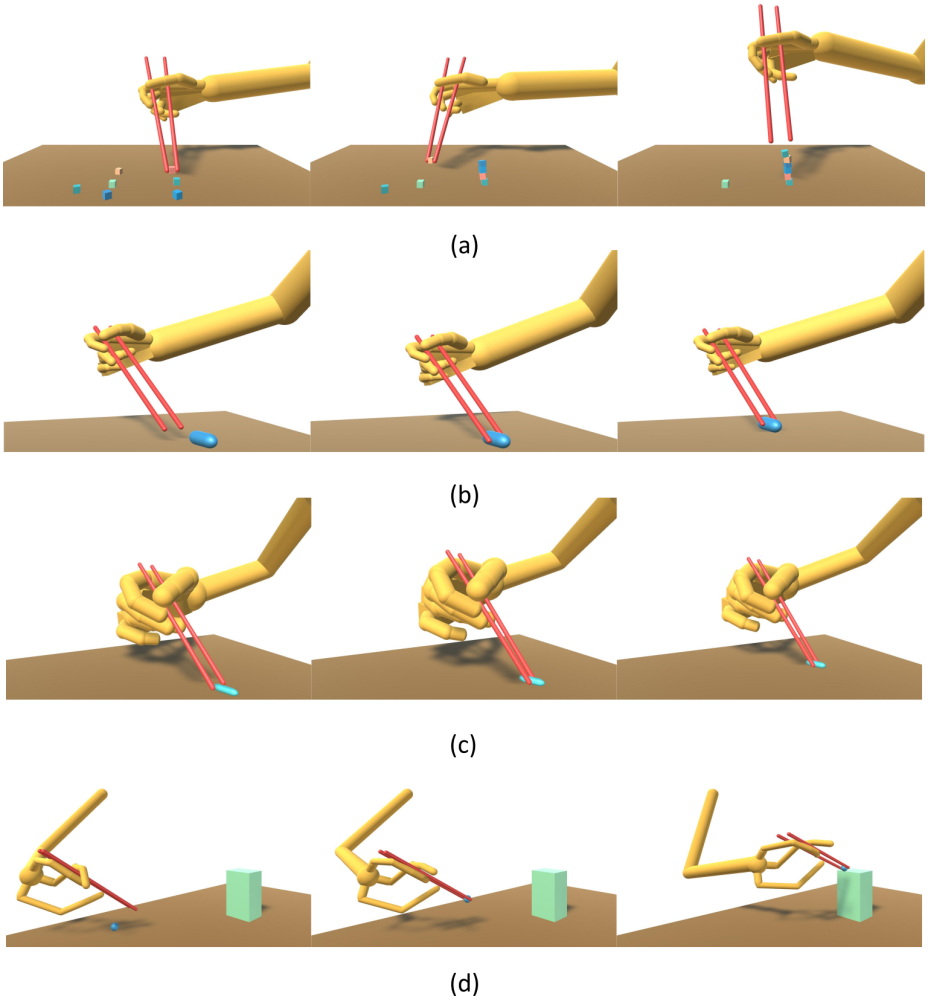


Figure 5.1: Our system learns how to use chopsticks in diverse gripping styles for multiple hand morphologies. The trained physics-based hand controllers can pick up and relocate objects of various shapes and sizes in realtime.

styles, holding positions, and for multiple hand morphologies, as shown in Figure 5.1. The accompanied video of this work is provided in the link [1].

5.1 Introduction

Dexterous hand manipulation and tool usage has been a long-standing challenge in computer animation and robotics. The main difficulties of tool use include the high degrees of freedom of the hands; the underactuation of the tools; and the complex interplay between the hands, tools and objects. The difficulty level also depends on the type of tools involved. Some tools only need to be grasped firmly in hand, such as hammers. Some tools need to be grasped

[1] `https://www.youtube.com/watch?v=rQHzwnSdsP8`

and manipulated by hand, such as scissors. In this chapter, we consider one of the most challenging tools: chopsticks.

Chopsticks are pairs of equal-length sticks used in Asian dinning for millenniums. Their simple design, or lack of design, poses several challenges in terms of control. First, the hand needs to grip and manipulate two independent sticks at the same time. Second, there are no obvious holding structures on the chopsticks, such as finger rings for a pair of scissors, to stabilize hand-tool contacts. Lastly, the chopstick-object contacts lie at the tip of the chopsticks, which are usually far away from the chopstick-hand contact points near the rear end of the chopsticks. Children usually need years of practice to master chopstick maneuvers. Even adults may find learning chopsticks challenging, if they did not grow up using them. The steep learning curve associated with chopsticks usage has spurred many video tutorials on YouTube, and the invention of training chopsticks that provide position retainer loops to stabilize finger-stick contacts.

The simple form of chopsticks, however, does enable their popularity and versatility. An estimated 33% of the world's population use chopsticks on a daily basis. Chopsticks are available everywhere, on dining tables as well as in the woods. They can pick up and move all kinds of foods: rice, meat, or noodles. They can manipulate in many different ways, so that spatulas, whisks, or pasta ladles are not necessary in Asian cooking. In robotics, research has been carried out to adopt chopsticks for eating assistance [21, 250], micro-manipulation [178], and medical surgery [185, 84, 175].

The practicality and generality of chopsticks come at the cost of control complexity. In robotics, the chopsticks are usually rigidly attached to robot arms with reduced DoFs. In graphics, research on using chopsticks is nonexistent so far, to the best of our knowledge. Chopsticks usage is emblematic of a wider category of difficult-to-solve multi-contact manipulation-and-control problems. We focus on solving the problem of using truly underactuated chopsticks, with reasonable robustness in diverse gripping styles and holding positions for multiple morphologies. Inspired by how parents teach children chopsticks skills, we tackle this challenging control problem by decomposing it into two sub-problems: how to hold chopsticks properly? And then how to manipulate objects using chopsticks?

To use chopsticks effectively, users need to firstly hold them firmly. Although there is a consensus on a so-called standard grip being the most efficient way to use chopsticks [249], many people find their own ways to grip chopsticks during learning [145, 153, 248], such as the various grips shown in Figure 5.2. We characterize a gripping style by the contact relationships between each finger and either chopstick. We optimize the gripping pose in a particular style by combining deep reinforcement learning and Bayesian Optimization. Such an approach enables automatic discovery of diverse gripping poses for unusual hand morphologies. Using the output grips of our BO optimization, a moving virtual hand can hold the chopsticks firmly in physics simulation, and achieve some basic open-and-close chopsticks maneuvers.

To use chopsticks proficiently, users also need to control finger movements precisely in order to relocate objects via the tips of the chopsticks. Such fine motor controls are most likely impossible to design manually, which were possible for locomotion tasks. We design a two-level control system that first plans the chopsticks movements kinematically, and then trains physics-based hand controllers via model-free deep reinforcement learning. The high-level kinematic motion planner consists of a grasping model to select the best chopsticks configuration for grasping the objects, and then a trajectory generator to optimize for a collision-free chopsticks trajectory based on the start and goal transformations of the object. The hand and arm trajectories are then solved from the desired chopsticks trajectory using inverse kinematics. All the planned reference trajectories along with the optimized gripping pose in a desired style are then passed to the DRL system to train the low-level hand controllers using simple tracking rewards. Our learned low-level hand controllers are able to grasp and move or throw objects of various shapes and sizes in real time, and the high-level motion planner can plan or replan trajectories at interactive rates.

In summary, the contributions of this work are mainly twofold:

1. We present a learning and control framework for object relocation using chopsticks. The high-level motion planner synthesizes collision-free kinematic trajectories at interactive rates, and the low-level physics-based hand controllers track the planned trajectories in realtime once trained. No sophisticated reward tuning or motion capture of human demonstrations are needed.

2. We use Bayesian Optimization combined with deep reinforcement learning to discover physically valid gripping poses in multiple styles. The optimized grips correspond well to human experiences and no manual specification is needed. Such an imitation-free method can generalize easily to different hand morphologies and is thus applicable to a broad range of graphics and robotics applications.

5.2 Related Work

Human hands and tool usage are the special anatomy and function that helped driving the evolution of human brain, which ultimately differentiated humans from the rest of the animal kingdom. Significant amount of research endeavors have been invested into tackling the challenging problem of synthesizing dexterous manipulation in simulation or on robots. We classify the most relevant recent works into two categories: hand manipulation and tool usage. Hand manipulation is manipulation of objects directly by fingers and the palm, such as grasping and relocation of objects [274, 105, 119], and in-hand manipulation [267]. Tool usage is manipulation of objects by tools that are operated by hands or robot arms. We refer interested readers to other orthogonal dimensions of research in hand and finger animation to the excellent survey provided in [234].

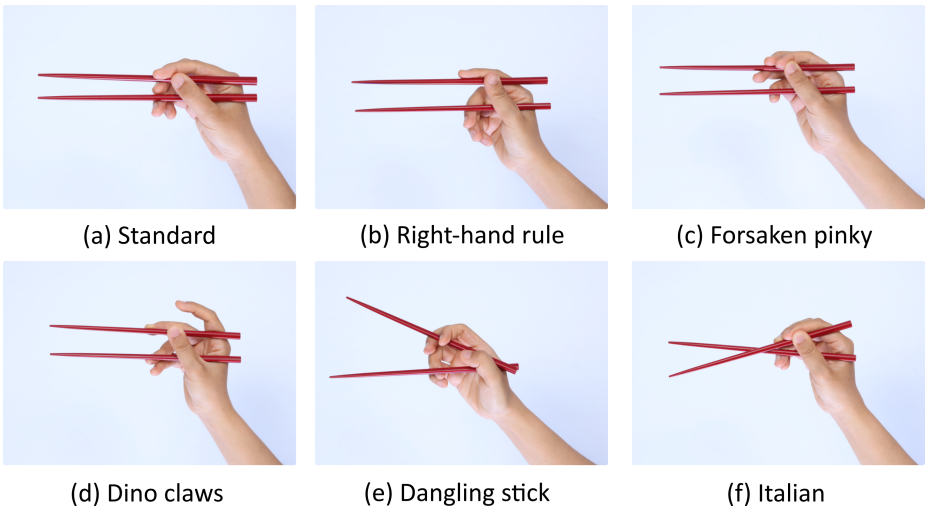


Figure 5.2: Multiple ways of holding chopsticks. Our system can discover similar gripping poses for seventeen styles, many of which correspond to the commonly used chopstick grips given in [134].

Kinematic Methods

Hand manipulations such as grasping or playing musical instruments can be synthesized through traditional inverse kinematic methods [77, 39, 94, 101, 12]. More recently, deep neural networks have also been utilized to synthesize hand manipulations [209, 91]. For example, [209] constructed a hand-object interaction database through motion capture, and trained a neural network to predict object grasping poses for human hands. [267] showcased impressive hand-object manipulation results such as turning a torus in hands. Kinematic methods, however, cannot generate motions responsive to a dynamic environment. The synthesized manipulations could also display artifacts such as penetrations into the objects. Moreover, data-driven kinematic methods usually require high-fidelity hand manipulation capture, which is hard and expensive in most cases.

5.2.1 Physics-based Methods

Physics-based control methods leverage physics simulation to generate motions with physical realism and environmental interactions. The key challenge, however, is to design or learn robust controllers to drive the simulated characters or hands. High-fidelity motion capture data has been utilized to help with synthesizing physically realistic hand manipulations [274, 105]. In the absence of motion capture data, trajectory optimization provides a viable approach to synthesize physics-based dexterous manipulations [118, 119, 255, 143, 230]. These methods usually need to model the dynamics to a great extent, such as incorporating friction constraints into the objective functions to avoid undesirable movements between hands and objects during grasping. Therefore, such methods usually simplify the dynamics to a certain degree to reduce the control complexity. For our case of dealing with five fingers

and two chopsticks, the contact and friction dynamics will probably be too overwhelming to handle, even with simplified physics.

We therefore opt for a model-free approach using deep reinforcement learning. DRL has been widely used in computer graphics and robotics to learn diverse motion skills, such as locomotion [165, 161, 15, 156, 211, 170], athletic skills [258, 122], and manipulations [177, 172, 146, 23, 10, 49, 130]. Challenging manipulation tasks, such as solving a Rubik's cube with a robot hand, have been demonstrated using DRL-based control learning methods [8]. It remains an open problem how to design suitable DRL reward functions to learn natural-looking skills for complex tasks, however. One idea is to leverage human demonstrations as reference skills for physical agents to imitate, which has been proven to improve both the learning efficiency and control robustness [177]. Multiple reference skills can be combined to help solve more challenging tasks using hierarchical deep reinforcement learning, such as dribbling a soccer ball or carrying objects to target locations [166, 136]. More recently, adversarial imitation learning DRL system has been quite successful at learning motion priors from large datasets of unstructured motion clips [170]. The motion priors obviate the need for manually designed imitation objectives or a high-level motion planner.

Capturing chopsticks skills, however, may be impractical as severe occlusions and subtle movements are involved. We also wish our solution to generalize well to graphics applications with monster-hand morphologies, and robotics applications with non human-hand-like manipulators. We thus rely on Bayesian Optimization coupled with DRL to discover diverse and physically valid chopsticks gripping poses. We then use a motion planner to generate chopsticks configurations and trajectories to satisfy kinematic task objectives. The discovered gripping poses and synthesized trajectories are then passed to our DRL-based training system to learn chopsticks skills using simple tracking rewards. Therefore, the advantages of our system include its simplicity in terms of system setup and tuning, and diversity in terms of gripping styles and hand morphologies.

5.2.2 Tool Usage

Tool usage has not been explored too much in the graphics community. The most relevant work is [268], which employed DRL to learn control policies to manipulate amorphous materials, such as gathering rice with scrapers or flipping pancakes with pans. Their tools are driven by a virtual proportional derivative controller. Then hand motions are reconstructed via inverse kinematics. There are more research activities on tool usage in the robotics community [242, 43, 218, 92, 93, 96], although most of them turn it into a simpler problem by attaching the tools directly onto the robot arm. For example, chopsticks are attached to a robot arm to grasp objects in [93], where human demonstrations are also used to help the learning of grasping policies. We study the problem of controlling underactuated chopsticks by hands, meaning that the chopsticks can actually move inside and fall out of the hand. We have not been able to find any prior work on exactly the same problem in the literature.

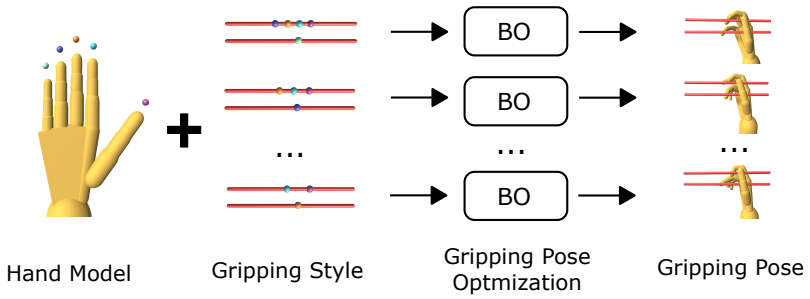


Figure 5.3: Gripping pose optimization: for a desired gripping style, we employ BO and DRL to optimize for a physically valid gripping pose of the hand.

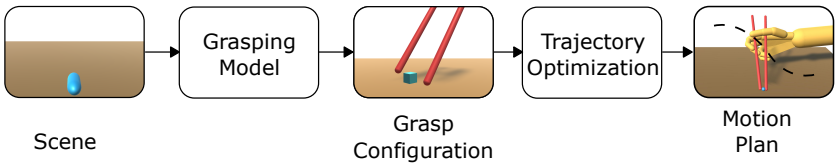


Figure 5.4: High-level motion planner: to achieve an object relocation task, the motion planner first selects a suitable chopsticks configuration for grasping, and then synthesizes collision-free trajectories for the chopsticks and hand.

5.3 Overview

Figure 5.3 5.4 and 5.5 provides an overview of our learning and control framework. Taken the models of one hand and two chopsticks as input, our goal is to learn robust hand controls to use the chopsticks for a variety of object grasping and relocation tasks. We achieve this goal by solving two sub-problems: first finding physically valid gripping poses in multiple styles in a *gripping pose optimization* step, and then learning *hand control policies* to hold the chopsticks in a specific gripping pose to pick up and relocate objects in simulation.

In the gripping pose optimization step as illustrated in Figure 5.3, we employ BO to find the optimal gripping pose for a desired style. A gripping style is characterized by a set of finger-chopstick contact relations, or specifically, which finger should be in contact with which stick. Gripping styles can be specified either manually or automatically as will be described later. Given a desired gripping style, our BO algorithm iteratively proposes a set of finger-stick contact positions, which are then converted into a hand pose using Inverse Kinematics (IK). We then employ DRL to evaluate the quality of the candidate gripping pose by training policies to perform simple open-and-close chopsticks maneuvers. The performance of the trained policy, characterized by its average reward in simulation, is then fed back to the BO to come up with the next proposal for contact positions.

After good gripping poses are found, we build hand controllers to relocate objects with chopsticks for each gripping pose. To this end, we design a two-level learning and control framework. The high-level kinematic motion planner as shown in Figure 5.4 first selects

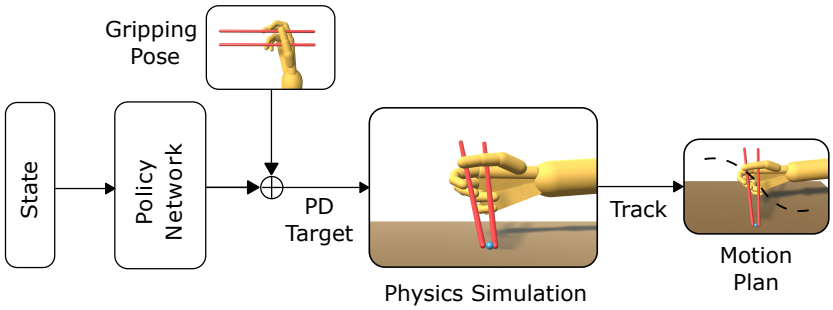


Figure 5.5: Low-level hand controller: then we train policy networks using DRL to track the planned trajectories for a chosen gripping pose.

the best chopsticks configuration in order to grasp the object of interest. We develop a neural-network based grasping model to recommend such chopsticks configurations based on the shape and transformation of the object. The planner then optimizes for collision-free trajectories for the chopsticks and hand, based on the relocation task required. Next the low-level hand control policy as shown in Figure 5.5 is trained through model-free DRL to track the generated motion plan. Once trained, the control policies can track novel reference trajectories generated by the planner in realtime to relocate objects of various shapes and sizes.

5.4 Physics-based Hand Tracking Control

We train tracking controllers for the hand to follow given trajectories of the object, the chopsticks, the hand, and the arm, while holding the chopsticks in a specific gripping pose. The trajectories to track are generated by a high-level motion planner in our system, but motion capture examples or manually defined keyframe animations could be used here if so wished. We use DRL to train these tracking controllers. The same DRL components are used for both the gripping pose optimization and the final control policy learning for actual object manipulations, although their reward terms are slightly different.

5.4.1 Simulation Setup

We simulate our hand models of potentially different morphologies together with a common two-link arm model as an articulated rigid-body system, allowing a moderate task space for object relocation tasks. The shoulder joint is fixed in position and only has three rotational DoFs. We note that hereafter whenever we refer to the state or pose of the hand, we really mean the state or pose of the whole hand-and-arm structure. We parameterize the state of the hand in generalized coordinates as $s_{\text{hand}} = (q, \dot{q})$, where q represents the joint angles and $\dot{q}$ the rotational speeds. All joints are actuated using a stable PD controller [210].

We model chopsticks as a pair of long rigid capsules of the same length and radius. The two sticks can move independently so the total DoFs of a pair of chopsticks is twelve. We denote the state of the chopsticks as s_{chop}=(p_i,o_i,v_i,ω_i), $i = 1, 2$, where p_i is the position of the Center of Mass (CoM) of Chopstick i, o_i, v_i and ω_i are the orientation, linear velocity and angular velocity of Chopstick i. Unless otherwise noted, the rotations are parameterized in quaternions in our system. The upper stick is indexed as Chopstick 1 and the lower stick is Chopstick 2, as shown in Figure 5.6.

The objects being manipulated by the chopsticks are modeled as rigid bodies from a predefined set of shapes and range of sizes as shown in Table 5.2. We use a tuple t_{obj} to indicate the shape and size of the corresponding object. The state of an object is denoted by a tuple $s_{\text{obj}} = (p_{\text{obj}}, o_{\text{obj}}, v_{\text{obj}}, \omega_{\text{obj}})$, which consists of the object's position, orientation, and linear and angular velocity.

5.4.2 Learning of Tracking Control

States and Actions

The state s is the input to the hand controllers. In our system, s consists of the simulation states of the chopsticks, the hand, and the object being manipulated, as well as the contact information between them. We compute the state of the chopsticks and the object with respect to the local coordinate frame of the palm, to facilitate learning and improve robustness. In addition, a short segment of the desired motion trajectory is also included as part of s to facilitate tracking. More specifically, the desired kinematic states of the hand, chopsticks, and objects of the next six frames from the planned trajectories are included in s. A complete list of all the components of s is provided in table 5.1. s_{hand}, s_{chop}, and s_{obj} are the simulation states of the hand and arm, the chopsticks, and the object, respectively. d_{hand} measures the distances between the fingertips and their desired contact locations on the chopsticks as specified in the gripping pose. f_{hand} and f_{chop} are the magnitude of the normal forces between the fingertips and the chopsticks, and between the chopsticks and the object, respectively. Quantities with a tilde on top represent desired states in the planned trajectory to track. In particular, $\tilde{s}_{\text{chop}} = (q_{\text{chop}}, \dot{q}_{\text{chop}})$, where q_{chop} is parameterized as the 7-DoF parallel gripper as described in Section 5.6.1. To encourage smooth tracking, we include six frames of these desired states sampled every $0.05s$ for the next $0.3s$.

The action a is the output of the hand controllers. In our system, a represents a corrective offset pose δq that will be added to a chosen gripping pose q^* to compose the final target pose required by a stable PD controller.

Rewards

The reward function is designed to encourage the hand to hold the chopsticks firmly in a chosen style and move the object following a desired trajectory. More specifically, it consists

of four reward terms:

$$r = e^{r_{\text{hand}} + r_{\text{chop}} + r_{\text{object}} + r_{\text{contact}}} \tag{5.1}$$

The hand control term r_{hand} encourages the hand and arm to match their planned trajectories:

$$r_{\text{hand}} = -10 \parallel q_{\text{hand}} - \tilde{q}_{\text{hand}} \parallel \tag{5.2}$$

where q_{hand} is the simulated hand pose, and $\tilde{q}_{\text{hand}}$ is the desired hand pose in the planned trajectory.

Similarly, the chopsticks term r_{chop} and the object term r_{obj} are defined as:

$$r_{\text{chop}} = -40 \sum_{i \in \{1,2\}} \|p_i - \tilde{p}_i\| - 10 \sum_{i \in \{1,2\}} \Theta(o_i, \tilde{o}_i) \tag{5.3}$$

$$r_{\text{obj}} = -40 \left\| p_{\text{obj}} - \tilde{p}_{\text{obj}} \right\| - 10 \cdot \Theta(o_{\text{obj}}, \tilde{o}_{\text{obj}}) \tag{5.4}$$

where p and o represent positions and orientations respectively. The scalar function $\Theta(o_1, o_2)$ computes the absolute angle between two quaternions o_1 and o_2.

Lastly, the term r_{contact} prevents fingertips from leaving or slipping on the chopsticks:

$$r_{\text{contact}} = -10 \cdot \sum_{i=1}^{i=N} d_i \tag{5.5}$$

where d_i is the minimal distance between the fingertip i (the part called distal phalanx in hand anatomy) and its desired contact position on the chopsticks as determined by the gripping style. N is the number of fingers that should remain in contact with the chopsticks in the corresponding gripping style, as will be described next.

symbol	description
s_{hand}	simulation state of the hand and arm
s_{chop}	simulation state of the chopsticks
s_{obj}	simulation state of the object
t_{obj}	shape parameters of the object
d_{hand}	distance between the fingertips and their desired contact locations on the chopsticks
f_{hand}	magnitude of contact forces on fingertips
f_{chop}	magnitude of contact forces on chopsticks tips
$\tilde{s}_{\text{hand}} \times 6$	planned states of the hand and arm
$\tilde{s}_{\text{chop}} \times 6$	planned states of the chopsticks
$\tilde{s}_{\text{obj}} \times 6$	planned states of the object

Table 5.1: Components of the state s of our hand controllers.

Algorithm 2: Gripping Pose Optimization with BO and DRL

Input: A gripping style c; maximal BO iterations *maxIter*.
Output: The optimal gripping pose q^* for c.
$i = 0$; $r^* = -\inf$
while $i <$ *maxIter* **do** //BO iterations
    $x \leftarrow$ contact position proposal by BO from c;
    $q \leftarrow$ solving gripping pose by IK from x;
    $r \leftarrow$ average reward of DRL-trained policy for q;
    update BO record with x, r;
    update Gaussian surrogate model;
    if $r > r^*$ **then** $(q^*, r^*) \leftarrow (q, r)$;
    i++;
end
return q^*

5.4.3 Training

We model our hand tracking control policy π as a fully-connected neural network with two 256-unit hidden layers and ReLU activations. We train these controllers using the PPO algorithm, where the value network shares the same structure as the policy network except that its last layer is a single linear unit. We employ multi-step returns $TD(\lambda)$ [207] and the generalized advantage estimator $GAE(\lambda)$ [189] to facilitate training of the neural networks.

5.5 Gripping Pose Optimization

A good chopstick gripping pose is a strong determinant of successful manipulations using chopsticks. Some gripping poses make it easy and efficient to use chopsticks, while others may feel awkward or even make it infeasible to manipulate with chopsticks. The goal of the gripping pose optimization component is to find the optimal pose with which the hand can hold the chopsticks steadily while still can move in a coordinated and flexible fashion to accomplish object manipulation using the tips of the chopsticks. Such an objective is difficult to formulate in closed form, thus we opt for a learning-based evaluation scheme coupled with Bayesian Optimization.

More specifically, we learn a control policy using reinforcement learning to track basic chopsticks maneuvers for a candidate gripping pose. The performance of the learned policy is then used as an assessment of the gripping pose. As the DRL-based evaluation is relatively expensive and not differentiable, we employ the sample-efficient and gradient-free Bayesian Optimization to suggest candidate poses. Our gripping pose optimization algorithm is summarized in Algorithm 2. Note that the arm is controlled to maintain a static pose at this stage, and dynamic arm movements will be synthesized later in Section 5.6.

5.5.1 Bayesian Optimization

Bayesian Optimization (BO) is one of the ideal choices for optimizing expensive black-box functions with no gradients. It is designed to minimize the number of function evaluations by querying the most promising and informative data points. For a given objective function, BO searches for its optimum through a series of evaluations, where the history of those evaluations are recorded to fit an *acquisition function*, which will determine the next candidate data point. Our gripping pose optimization is based on the GP-UCB algorithm [199] implemented with the Gaussian process framework [51]. Interested readers can refer [199] for more details. We set the maximal allowed number of function evaluations to ten in our implementation.

5.5.2 Chopsticks Gripping Style

In our system, a chopsticks gripping style is characterized by the contact relationships between each finger and either chopstick. For a human hand with five fingers, namely the thumb, index, middle, ring, and little finger, we can represent a gripping style with a 5-tuple $c = (c_1, c_2, c_3, c_4, c_5)$, where $c_i = j, j \in \{0, 1, 2\}$ indicates that finger i should be in contact with chopstick j. 0 denotes no contact. Following this notation, the standard gripping style shown in Figure 5.2 can be represented by the tuple $(1, 1, 1, 2, 0)$, for example. In general, for a hand model with N fingers, its chopsticks gripping style can be defined with an N-tuple.

For hands with a small number of fingers, we could simply enumerate all values of the gripping style tuple and optimize a gripping pose for each of them. A human hand, for example, has potentially a maximal of $3^5 = 243$ gripping styles. However, many of them may be infeasible, inefficient, or unnatural. We thus employ two heuristics to prune the style space to eliminate bad styles, as well as to reduce the optimization workload. First, the thumb plays a crucial role in performing precise tool use, and therefore we do not allow c_1 to be zero. We also observe that the thumb is usually used to support the upper Chopstick 1 as shown in Figure 5.2. This is because the lower stick has some default support from the valley between the thumb and the index finger, while the upper stick needs the thumb more for support and movement. Consequently, we set $c_1 = 1$ and shrink the style space by 2/3. Second, finger crossing usually leads to awkward or infeasible grasping poses, thus the gripping styles with finger crossings are excluded from further pose optimization. After applying these two heuristics, there are seventeen gripping styles left, for each of which we run BO to obtain an optimized gripping pose.

5.5.3 Gripping Pose Generation

When given a chopsticks gripping style as input, represented by an N-tuple, we use BO to search for the optimal gripping pose for that style. To reduce the degrees of freedom of the problem, we first optimize the finger-stick contact positions according to the contact

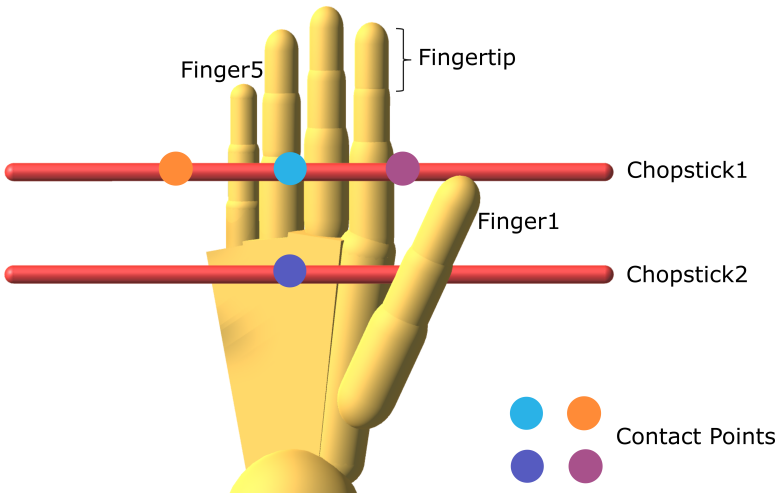


Figure 5.6: The default T-Pose used by our IK solver. The upper stick is indexed as Chopstick 1 and the lower stick is Chopstick 2. Fingers are indexed from the thumb to pinky as Finger 1 to 5. A fingertip is the first segment of a finger.

patterns specified in the style tuple. Specifically, we parameterize each contact position using a single scalar x representing the contact location along the chopstick. Then the BO algorithm just needs to optimize a vector x up to N dimensions. Once the contact positions are determined, the gripping pose can be obtained by an Inverse Kinematics (IK) solver. The quality of the gripping pose is then evaluated by the performance of its trained policy using deep reinforcement learning as described in Section 5.4.2.

Inverse kinematics

We implement an optimization-based IK solver. The objective function is formulated for a position vector x as follows:

$$\min_{q} \sum_{i=1}^{N} \|f_i(q) - p_i(x)\|_2^2 + \text{clog}(\delta_i, 0.001) \tag{5.6}$$

where $p_i(x)$ represents the 3D position of the contact point on the chopstick that finger i should touch, $f_i(q)$ calculates the 3D position of the point on fingertip i that is closest to $p_i(x)$, and δ_i represents the depth of penetration between fingertip i and its contacting chopstick. clog$(\cdot)$ is a modified clamped log-barrier function defined as:

$$\text{clog}(z, z_0) = \begin{cases} -\frac{(z-z_0)^2}{z}\ln(\frac{z}{z_0}), & 0 < z < z_0, \\ 0, & z \geq z_0 \end{cases} \tag{5.7}$$

The first term in Equation 5.6 encourages fingers to touch chopsticks at the positions proposed by BO, and the second term penalizes potential penetrations between the fingers and

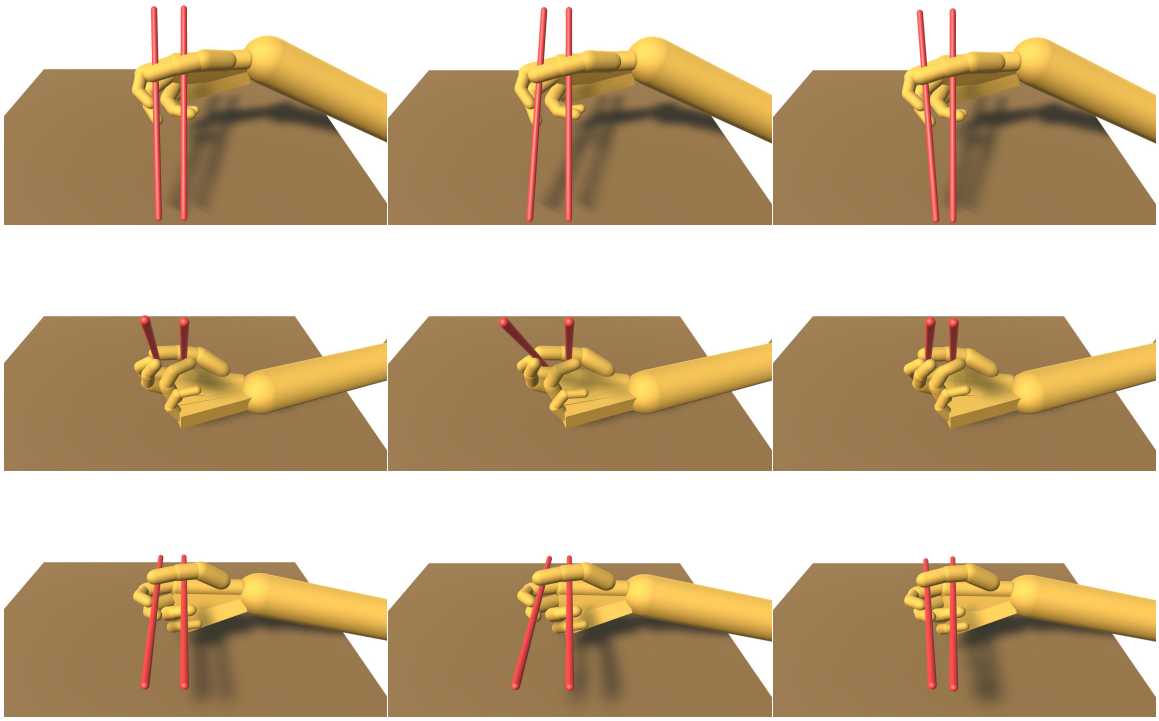


Figure 5.7: For gripping pose evaluation, three one-second long motions are used to train the hand controller to open and close chopsticks while pointing to different directions.

the chopsticks. Note that we only explicitly control contacts between the chopsticks and the fingertips. Chopsticks contacts on other parts of the fingers or the hand, such as contacts with the valley between the index finger and the thumb, emerge naturally during our DRL policy learning.

We solve this IK problem using L-BFGS. As the objective function in Equation 5.6 is highly non-convex, we use the default pose shown in Figure 5.6 to initialize the IK solver. This effectively eliminates poor local minima, such as fingers touching the wrong side of the chopsticks.

5.5.4 Gripping Pose Evaluation

We evaluate each candidate gripping pose proposed by BO via deep reinforcement learning of a tracking control policy to accomplish basic chopsticks maneuvers without manipulating any object. The details of the DRL training can be found in Section 5.4. Since no objects are involved in these maneuvers, the corresponding reward term of Equation 5.4 is excluded from the reward function. Three one-second long motions as shown in Figure 5.7 are used for training, where the chopsticks open and close several times while pointing to different directions. We train each tracking policy for 500 epochs, then run the learned controller to perform all the test motions again. The average reward of the simulated motions, i.e., the undiscounted cumulative reward divided by the episode length, is sent back to BO as the

quality score of the input candidate pose. More complicated maneuvers could be used here for gripping pose evaluation, but the cost of computation will go up as well.

5.6 High-level Motion Planning

Given a relocation task characterized by an object to be moved and its target location, we propose a hierarchical control framework where a high-level motion planner is responsible for generating feasible kinematic motion trajectories for the hand, the chopsticks, and the object to accomplish the task. A low-level hand tracking controller, as described in Section 5.4, then tries to follow these trajectories to drive the simulated hand to move towards the object, pick it up and then drop it at the target location using the chopsticks.

The motion planner is queried once for each object to be relocated. It synthesizes feasible trajectories in two steps. First, it proposes a chopsticks configuration from a grasping model to ensure quality of the grasp, as will be detailed shortly in Section 5.6.1. The grasping model consists of pretrained neural-network based models independent of the gripping styles and hand morphologies. Then a trajectory generation module computes the actual trajectories for the hand and arm, through trajectory optimization and inverse kinematics as will be described in Section 5.6.2. The trajectory generation algorithm takes as input the chopsticks configuration proposed by the grasping model, the object start and goal locations, as well as the hand morphology and desired gripping style.

The trajectories generated by the high-level motion planner are then passed to the low-level hand control policy for tracking. We train one hand controller for each chopstick gripping pose in the desired style by tracking a large set of trajectories created by the motion planner for moving objects between random start and goal locations. Once trained, the policy is robust enough to generalize to new trajectories generated by the same motion planner for new tasks not present in the training set.

5.6.1 Grasping Model

Determining how and where to grasp an object using chopsticks is a core mission of the motion planner, which is particularly challenging when dealing with objects of various shapes. We therefore simplify the planning problem in two ways. First, we reduce the DoFs of the chopsticks from twelve to seven, similar to a parallel gripper as illustrated in Figure 5.8. This is based on the observation that humans usually hold the lower chopstick firmly and rotate the upper chopstick around the object to prepare for grasping. The configuration of the 7-DoF chopsticks can thus be described by a tuple $q_{\text{chop}} = (p_{\text{chop}}, o_{\text{chop}}, \varphi_{\text{chop}})$, where p_{chop} and o_{chop} represent the position and orientation of the lower chopstick, and the scalar φ_{chop} represents the relative rotation of the upper chopstick with respect to the lower stick around the axis perpendicular to both sticks. Note that this simplified chopsticks model is only used in motion planning, and the full 12-DoF chopsticks are still used in simulation.

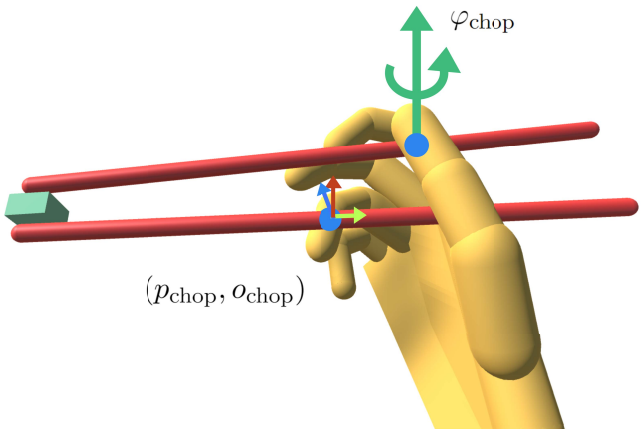


Figure 5.8: The 7-DoF chopsticks model used in motion planning. The hand controllers still use the 12-DoF chopsticks model in simulation.

We further assume that the line connecting the two tips of the chopsticks go through the CoM of the object when grasped, which further reduces the planning problem to three DoFs, i.e., the orientation of the lower chopstick o_{chop}. Once o_{chop} is known, we can compute p_{chop} from the object's position and φ_{chop} from the object's shape information.

We develop a neural-network based grasping model to predict the optimal grasping configuration for the chopsticks, from the shape and configuration of the object to be grasped. Our grasping model supports efficient planning and replanning at runtime, and is partially inspired by the GraspNet model proposed in [42]. More specifically, our grasping model consists of two neural networks. Given the shape parameter of an input object, the *configuration network* nominates a number of candidate chopsticks configurations for grasping, together with their probabilities of success. The chopsticks configurations are specified in the local coordinate frame of the object in the configuration network. We then transform them into the global coordinate frame and pass them into a *reachability network* to estimate how likely each candidate configuration is within the reachable space of the simulated hand and arm. We also compute a continuity score for a configuration by measuring how close it is to the current chopsticks configuration in simulation. Closer chopstick configurations help produce natural hand and arm motions in sequential relocation tasks. We then multiply the probability from the configuration network, the score from the reachability network, and the continuity score all together as the final quality score of a candidate grasping configuration. The motion planner then selects the candidate configuration with the highest score for further trajectory planning.

Configuration network

There are usually multiple ways to grasp an object, especially for symmetric objects such as a ball. We thus design the configuration network to evaluate multiple candidate configura-

	Width/Radius	Length	Height
Sphere	[0.5cm,1cm]		
Capsule	[0.5cm,1cm]	[2cm,4cm]	
Box	[1cm,2cm]	[1cm,2cm]	[1cm,2cm]

Table 5.2: The range of size of our tested geometry primitives.

tions simultaneously. Specifically, we uniformly discretize the three dimensional chopsticks configuration space into a set of N_c candidate configurations denoted as $\mathcal{C}_{\text{chop}}$. As the configuration space corresponds to the orientation of the lower chopstick, this discretization can be performed easily using Euler angles. Then the configuration network takes the shape parameters of an object as input, and computes an N_c-dimensional vector representing the success probability for each configuration.

The configuration network is implemented as a fully-connected neural network with two 512-unit hidden layers and tanh as the activation function. A softmax layer is appended after the last linear layer to turn its output into probabilities. The network is trained as a multi-class classification problem, matching a given object to configurations in $\mathcal{C}_{\text{chop}}$ with estimated success probabilities. The top n_c configurations will be passed to the reachability network for further assessment, as will be described shortly. In our implementation, we choose $N_c = 2000$ and $n_c = 10$.

We use synthetic data for our supervised learning problem. We first generate 100 objects from three primitive shapes and uniformly sample their sizes in the range as shown in Table 5.2. For each object, we then optimize for the grasping configuration using the particle swarm optimization algorithm with n_c random initial solutions [139]. The optimization objective follows the grasp quality metric employed by [274], which encourages the center of the line connecting the chopsticks tips to stay close to the object CoM, and the direction of the connecting line to align with contact surface normals. Each of the optimized configurations is then mapped to its nearest neighbour in $\mathcal{C}_{\text{chop}}$, for which the success probability in its corresponding one-hot vector is labeled as one.

Reachability network

The reachability network evaluates each candidate chopsticks configuration proposed by the configuration network, in terms of reachability in the global frame. We implement the reachability network as a fully-connected neural network with two 256-unit hidden layers, tanh activation functions, and a sigmoid output layer. Different from the configuration network, the reachability network is trained as a binary classification problem, which outputs the probability that whether a configuration is reachable by the hand and arm.

We use synthetic data for training as well. We first randomly sample 10000 chopsticks configurations in an operating cuboid of $0.5m \times 0.5m \times 0.25m$ over the table on which objects are placed. We then solve their corresponding arm poses using the analytical arm

IK algorithm proposed in [217]. More specifically, from the chopsticks configuration and the chosen gripping pose and hand morphology, we can compute the desired rigid transformation for the hand, from which the IK algorithm then solves for a pose for the 7-DoF arm. If the hand transformation is reachable, the IK solver returns a solution, otherwise the IK solver returns no solution. For training, chopsticks configurations with an IK solution are labeled with probability one while configurations with no IK solutions are labeled with zero.

The reachability network could simply be replaced by the arm IK solver, which is relatively fast and generates solutions of acceptable quality. We choose the neural network based model for reachability tests mainly for the potential of switching to a more expensive IK algorithm in the future that not only tests the reachability of the desired end-effector transformations, but also assesses the naturalness of the solved arm poses. We note that in robotics, machine learning models such as Gaussian mixture models are also used to rapidly predict a feasible catching configuration for object catching tasks [95].

Configuration Continuity

For a sequence of object relocation tasks, the hand and arm need to manipulate the chopsticks from one configuration to the next. The closer the chopsticks configurations are, the smoother the hand and arm trajectories will be, which improves the overall continuity and naturalness of the animation for consecutive relocation tasks. We thus calculate a continuity score between the candidate chopstick orientation and the current simulated chopstick orientation $\xi = \exp(-5 \cdot \Theta(o_{\text{chop}}, o^0_{\text{chop}}))$, where o^0_{chop} is the simulated lower chopstick orientation when the motion planner is activated for a new object relocation task. We then multiply the continuity score with the grasping success probability and the reachability score as the final quality score estimated for a candidate configuration.

5.6.2 Trajectory Generation

Once the optimal chopsticks configuration is selected by the grasping model, the motion planner needs to generate trajectories for the chopsticks, the hand and arm, and the object. This is done in three steps. First, a collision-free trajectory for the chopsticks is computed through trajectory optimization. Then the arm trajectory is solved by the arm IK algorithm, which also considers continuity when applied to a sequence of IK problems [217]. Lastly, the object trajectory is easily synthesized from the tips of the chopsticks. As the latter two steps are straightforward, we will only discuss our trajectory optimization method for the chopsticks.

Following the conventions in hand manipulation research, we segment an object relocation task into three phases: *approaching* the object, *relocating* the object, and then *releasing* the object. In the *releasing* phase, we simply keep the transformation of the chopsticks fixed and set φ_{chop} to 0. We utilize trajectory optimization for the first two phases. Two additional control points are inserted between the start and end chopsticks configurations, then

the chopsticks positions are computed by a degree-three Béizer curve from the four control points, and the chopsticks orientations are computed with spherical linear interpolation between two control points. We then solve for the optimal control points that generate the shortest collision-free trajectory, which can be formulated as

$$\min_{q_1,q_2} \int_t \|\dot{m}(t)\| \, dt + \text{clog}\left(\delta(m(t)), 0.001\right) dt \tag{5.8}$$

where $m(t) = m(t; q_1, q_2)$ represent the trajectory defined by the control points q_1, q_2, and δ computes the penetration depth between the chopsticks and the environment in trajectory $m(t)$, and $\text{clog}(\cdot)$ is the same barrier function as defined in Equation 5.7. The duration of this trajectory is computed as $d_{\text{chop}}/\tilde{v}_{\text{chop}}$, where d_{chop} is the length of the displacement vector of the chopsticks in the current phase, and $\tilde{v}_{\text{chop}}$ is a hyperparameter indicating the desired speed of the chopsticks. We use $\tilde{v}_{\text{chop}} = 0.25\, m/s$ for our demo results. We use numerical integration with a discretized time step $10ms$ to compute the integral in Equation 5.8. We use the L-BFGS algorithm with multi-start points to solve the optimization, similar to [113].

5.7 Results

We implement our system with Pytorch [159] and the MuJoCo physics simulator [216]. We use two 6-DoF capsules to model the chopsticks. The length and radius of the chopsticks are 26 cm and 0.4 cm respectively. Our hand model as shown in Figure 5.9 has 30 DoFs in total, and shares the same joint hierarchy as the hand in [255]. The only difference is that the palm part connecting the thumb and index finger, the so-called union valley in acupuncture, is modeled with a capsule instead of a box. We found that the chopsticks need stable contacts with the valley to perform well, which aligns with human experiences. We mount the hand on a 7-DoF robot arm consisting of a shoulder, an elbow, and a wrist joint. We run the simulation and control at the same frequency at 100 Hz. Torque limits for hand joints are taken from values of the shadow hand robot used in [177].

For low-level DRL policy training, we set λ to 0.95 for both TD(λ) and GAE(λ). The discount factor γ is set to 0.99 and the PPO policy clip ratio is 0.2. The learning rates of the actor and critic network are set to 3×10^{-5} and 3×10^{-4}, respectively. We sample 1×10^4 state-action tuples with 32 parallel simulation environments for each training epoch, as all experiments are performed on a workstation with 32 Intel-i9-9980XE CPUs. Both networks are updated 10 times in each epoch with a mini-batch size of 256. During training the action noise linearly decreases from 0.1 to 0.01 in the first 5×10^7 simulation steps to encourage exploration in early training and exploitation in later training. For learning basic chopsticks maneuvers of fixed length in the stage of gripping pose optimization, we run PPO for 500 epochs. For learning object relocation, we gradually increase the episode length to facilitate policy learning similar to [161]. More specifically, the episode length is 100 control steps

for the first 5000 epochs, and then increases by 100 steps every 500 epochs. The tracking trajectories are generated by the motion planner for 100 multi-object relocation tasks and last for about 30 minutes in duration. Each multi-object relocation task contain eight objects to be moved between random locations. We run PPO for 2×10^4 epochs for each gripping pose in a particular style, which takes roughly two days on our workstation.

For high-level motion planning, the planner is queried whenever a new object is required to be moved. The planning takes roughly $3 \sim 5$ seconds using a non-optimized single-thread implementation on our workstation.

We show optimized chopsticks gripping poses in different styles in Section 5.7.1. Various learned chopsticks skills are demonstrated with multiple hand morphologies in Section 5.7.2, 5.7.3, and 5.7.5. We demonstrate the ability to learn to use other tools with the same framework in Section 5.7.4. Lastly in Section 5.7.6 we conduct ablation studies and comparisons to validate each component of our framework.

5.7.1 Diverse Chopsticks Gripping Styles

The Bayesian Optimization takes roughly six days in total to optimize gripping poses for the seventeen valid gripping styles. In Figure 5.10 we show the most distinctive five styles: $(1, 1, 1, 2, 0)$, $(1, 1, 1, 1, 2)$, $(1, 1, 2, 0, 0)$, $(1, 0, 1, 2, 0)$ and $(1, 0, 0, 1, 2)$. Four out of the five styles are documented in [134], following which we name the gripping styles. The standard style with $c = (1, 1, 1, 2, 0)$ is the most common way to use chopsticks, and also considered the best way by common belief. Our BO results verify that it is indeed the most efficient way to use chopsticks, as policy learning for the standard style converges the fastest with the highest normalized return, as shown in Figure 5.20. The optimized gripping poses for the remaining twelve styles are shown in Figure 5.12.

5.7.2 Chopsticks Skills for Object Relocation

We evaluate our framework with object relocation tasks where the simulated hand uses the chopsticks to grasp objects of various shapes and sizes, and then move or throw them to some desired target locations.

Grasp and Move

We train hand control policies using sequential grasp-and-move object relocation tasks. Figure 5.11 shows the learned chopsticks skills using different gripping styles. We also test the generalization ability of the learned controllers with a more challenging stacking task, as shown in Figure 5.1 (a). This task is not included during policy training, yet the learned controllers can directly finish the stacking task without any fine tuning.

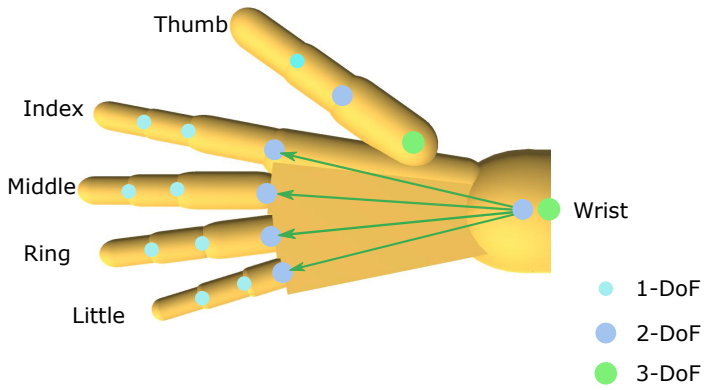


Figure 5.9: Our simulated hand has 30 DoFs in total. The thumb has 6 DoFs, and the other fingers each has 4 DoFs. The four bones connecting the root of the hand to the base of the fingers, indicated as green arrows, each has 2 DoFs to model small deformations of the palm.

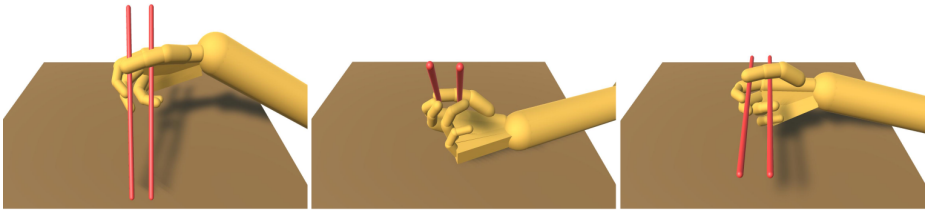
Grasp and Throw

Throwing is an efficient means to relocate objects when the target position is out of reach by the arm and hand. Our motion planner generates kinematic chopsticks trajectories for throwing tasks following the method described in [264]. More specifically, we first estimate the needed position and velocity of the chopsticks at the end of the relocating phase, given the final target position of the object. Then we use trajectory optimization for the relocating phase to first move the chopsticks to the estimated releasing position and then rotate them to achieve the estimated releasing velocity. Trajectory generation for the approaching and releasing phases are the same as described in Section 5.6.2. The trajectory of the object after being thrown is assumed to be projectile, and aerodynamic drags are ignored. We train the hand control policy by tracking the above planned trajectories, the same as for the grasp-and-move task. Figure 5.13 shows one of our throwing results where a box is grasped and thrown to hit another stack of boxes far away.

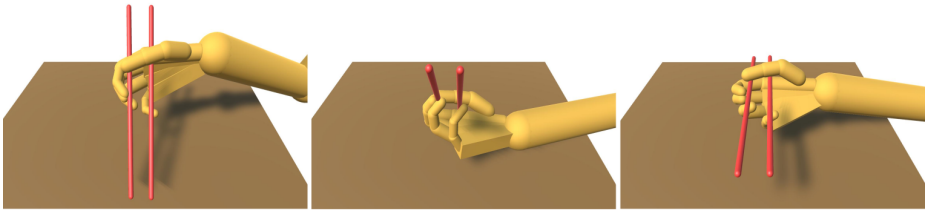
Parameterized Controllers for Different Holding Positions

Humans can hold chopsticks at different positions in similar poses. For example, children and beginners tend to hold chopsticks near the tips for easier control. Expert users hold chopsticks near the rear ends to increase the reachability and avoid potential harmful injury from hot food. Our system can learn such parameterized controllers with ease.

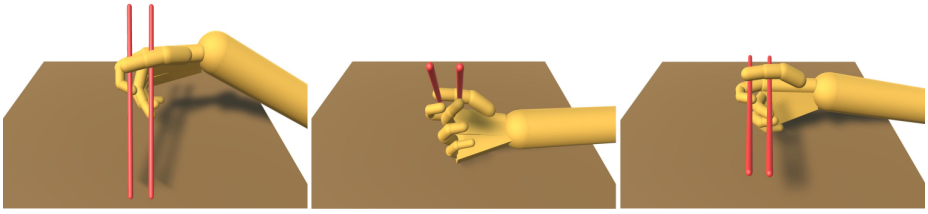
We parameterize the holding position of the chopsticks by translating the chopsticks along the axial direction of the lower chopstick. Such translations do not change the relative rotation between the chopsticks and the palm. The holding position is also added into the input state representation of the hand control policy to enable learning of parameterized



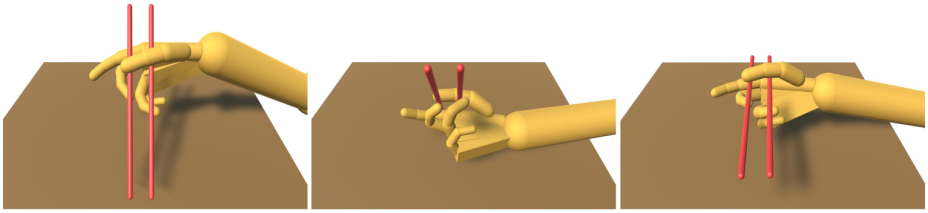
(a) c=(1, 1, 1, 2, 0) Standard style



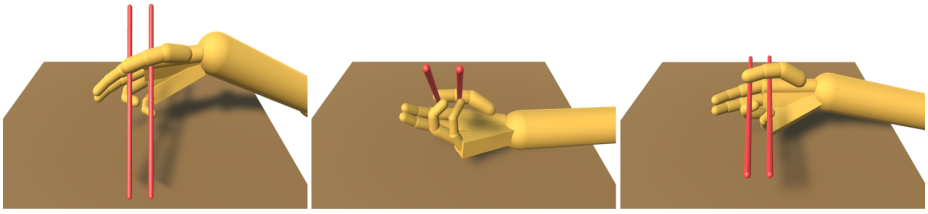
(b) c=(1, 1, 1, 1, 2) Forsaken pinky style



(c) c=(1, 1, 2, 0, 0) Right-hand rule style

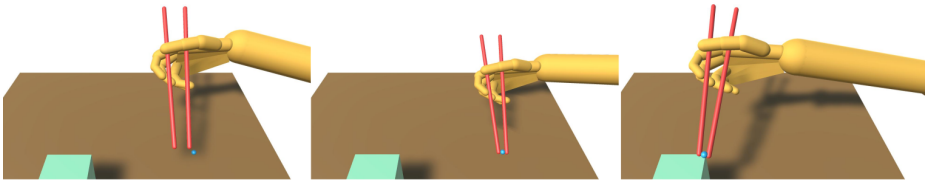


(d) c=(1, 0, 1, 2, 0) Dino claw style

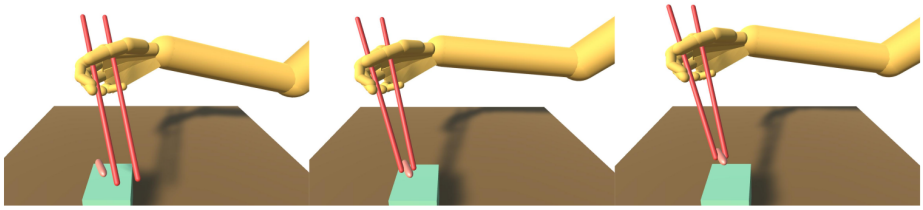


(e) c=(1, 0, 0, 1, 2) Style with no convention name

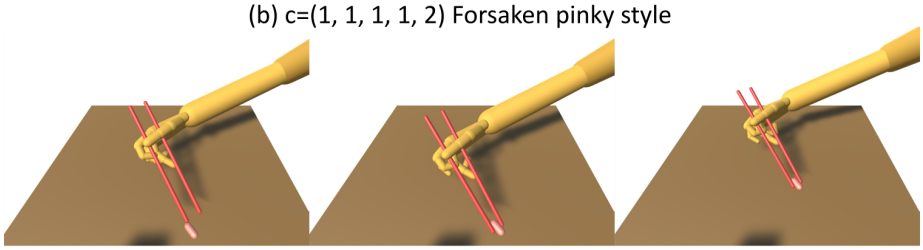
Figure 5.10: Visualization of the optimized chopsticks gripping poses in five styles, performing the basic open-and-close chopsticks maneuver.



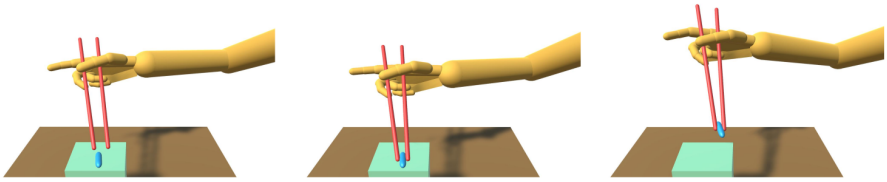
(a) c=(1, 1, 1, 2, 0) Standard style



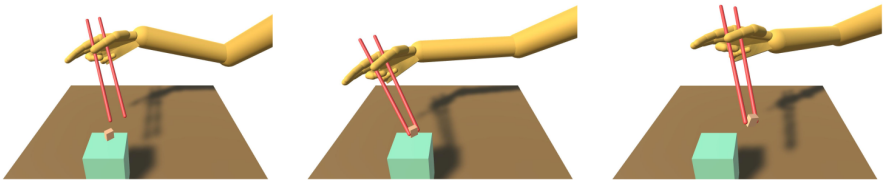
(b) c=(1, 1, 1, 1, 2) Forsaken pinky style



(c) c=(1, 1, 2, 0, 0) Right-hand rule style



(d) c=(1, 0, 1, 2, 0) Dino claw style



(e) c=(1, 0, 0, 1, 2) Style with no convention name

Figure 5.11: The hand controls chopsticks to grasp and move various objects with different gripping poses.

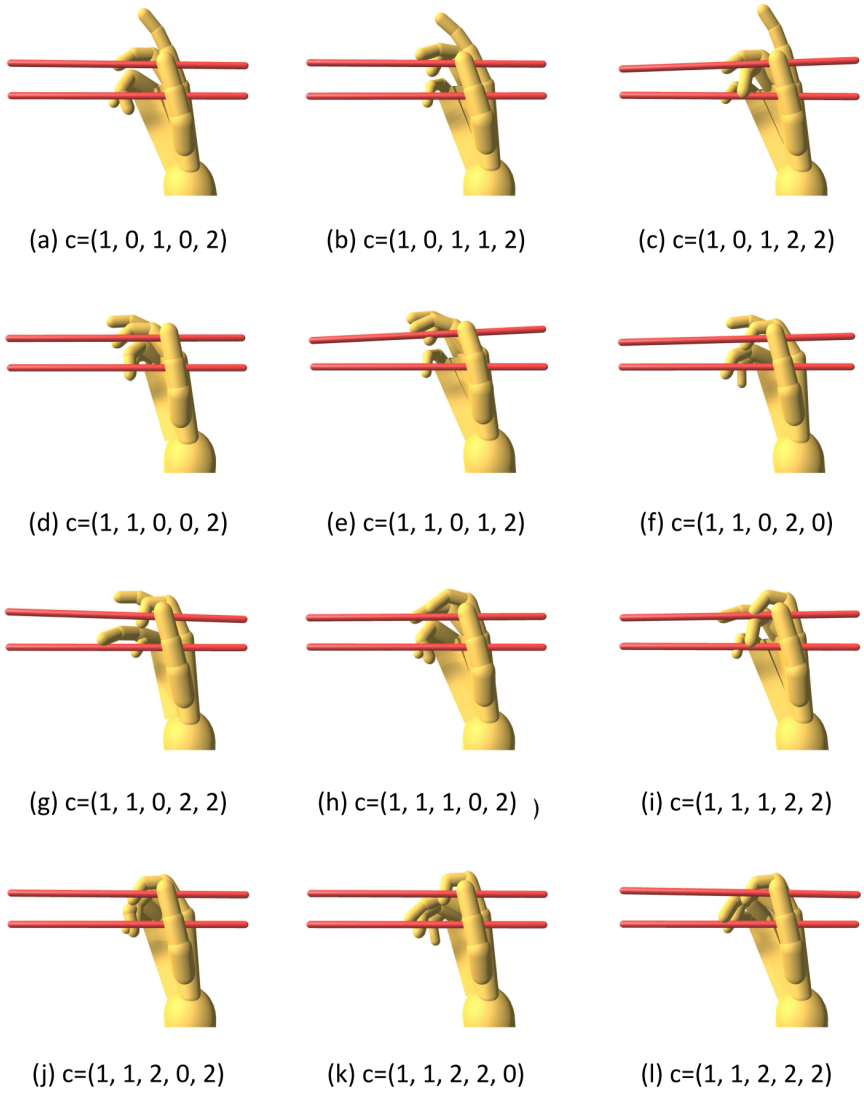


Figure 5.12: Visualization of twelve additional optimized gripping poses.

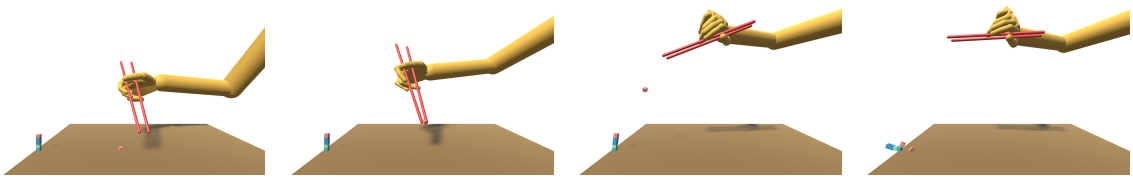


Figure 5.13: Grasping and throwing a box to hit another stack of boxes.

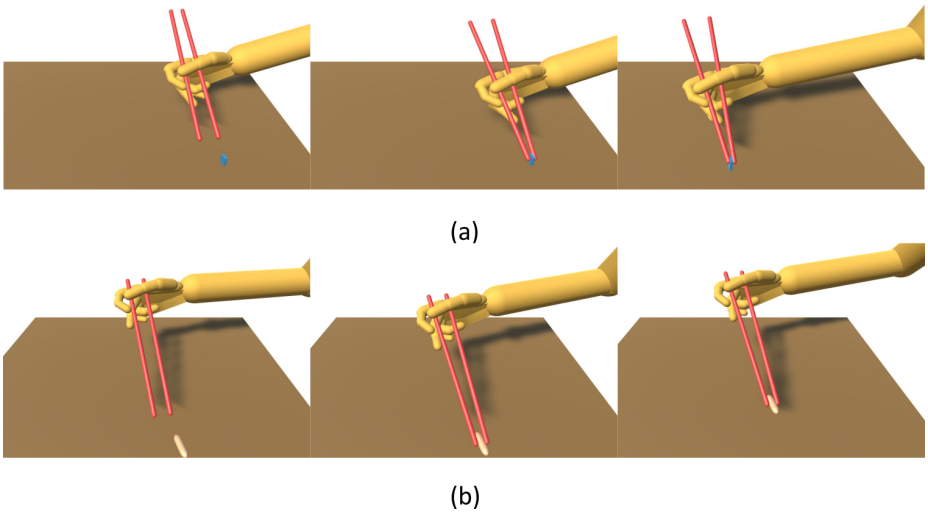


Figure 5.14: Our controllers are parameterized so that the hand can hold the chopsticks high or low.

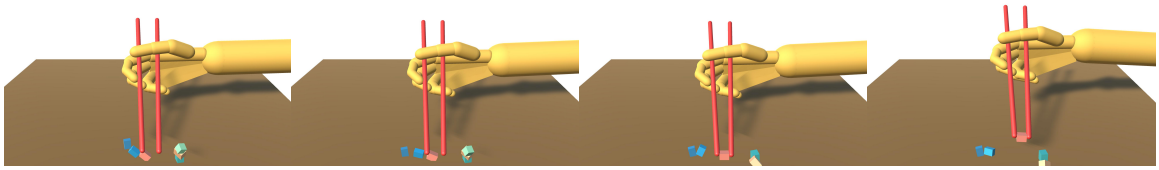


Figure 5.15: Grasping a moving object without replanning.

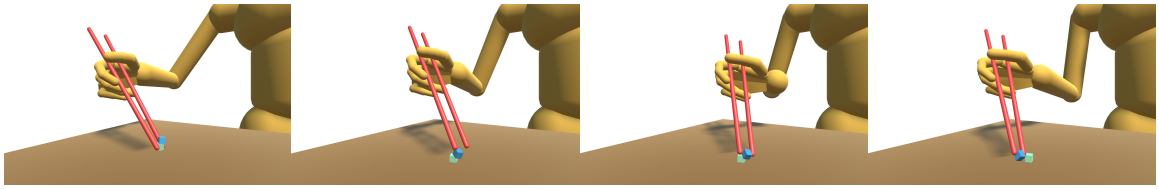


Figure 5.16: Relocating two boxes together. The top blue box falls down at the end of the first move. With replanning, the chopsticks are able to grasp the top box during falling to continue the second move.

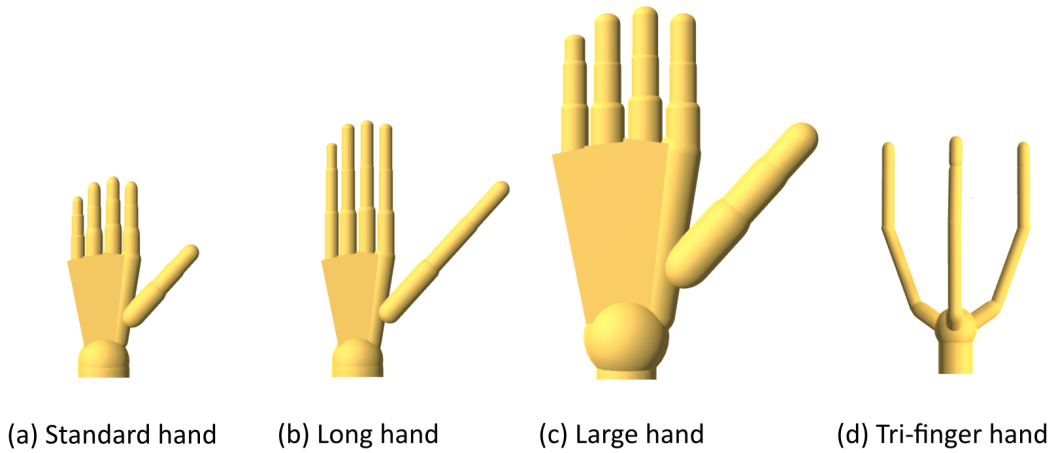


Figure 5.17: Our framework works well for drastically different hand morphologies.

controllers. Figure 5.14 shows that the learned controllers can hold the chopsticks at multiple positions to relocate objects. We allow a range of ± 5 cm for the chopsticks translation in our tests. Note that policies using some holding positions are harder to train than others, which leads to biased sampling during DRL training. We adopt the adaptive sampling strategy similar to [239] to increase samples for harder tasks.

5.7.3 Diverse Hand Morphologies

Our framework can learn chopstick skills for drastically different hand morphologies. In Figure 5.17 we visualize three additional hand models that we have tested: a hand with long fingers, a large hand, and a tri-finger hand. The fingers of the long hand double the lengths of the fingers of the standard hand. The large hand doubles the size of the standard hand in all dimensions for both the fingers and the palm. The tri-finger hand is designed by ourselves following claw toy grabbers commonly seen in arcade machines. We show some of the learned skills for each hand in the teaser Figures 5.1(b), (c) and (d). We are happy to find that the tri-finger hand can learn to use chopsticks successfully as well. Note that we still use the same arm model for these different hands, although we scale the radius of the arm to match the dimension of the hands for better visualization.

5.7.4 Using Other Tools

Our framework can be applied to learn physics-based hand controllers to use other types of tools. In Figure 5.19 we demonstrate relocating a ball with a pair of tongs, and tracing a curve using a brush. The tongs are modeled as a 7-DoF gripper, with its two bars modeled as $26cm \times 0.8cm \times 2cm$ cuboids. We model the pivot of the tongs with an angular spring to provide restoring torques. More specifically, the torques are computed as $-k(\theta - \theta_{\text{rest}})$,

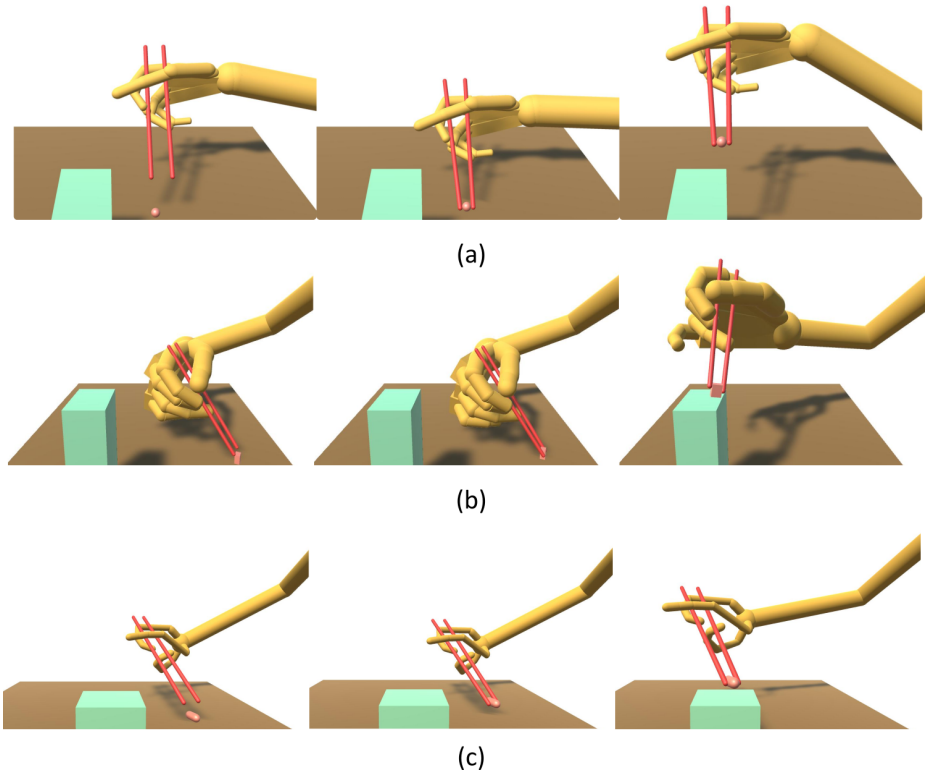


Figure 5.18: Chopsticks skills learned for hands of different morphologies.

where $k = 20Nm$ is the stiffness parameter, and $\theta_{\text{rest}} = 0.2\ rad$ is the resting angle between the two bars. The brush is modeled as a long capsule of $26cm$ in length and $0.4cm$ in radius.

We follow the same pipeline designed to learn chopsticks skills to learn tongs skills. The only difference is that we only allow the thumb, index, and middle fingers to contact the tongs. For curve tracing with the brush, we use a much simplified version of the motion planner. As there is no tool-object interactions involved, the grasping model to predict a configuration for the tool to grasp the object is not needed. Trajectory generation for the tool is also much easier, by simply tracing the curves using the tool tip, and the hand and arm trajectories can be solved directly from IK.

5.7.5 Robustness

Our learned physics-based hand controllers can handle noises, perturbations, and uncertainties to certain extent. Figure 5.15 shows such an example. The target object rotates due to unexpected collisions when the chopsticks approach the object. The controller can still grasp the rotated box successfully according to the plan computed before the collision. Figure 5.16 shows another example of relocating two stacked boxes together. The chopsticks only grasp the bottom box during the first move and the top box starts to fall when the

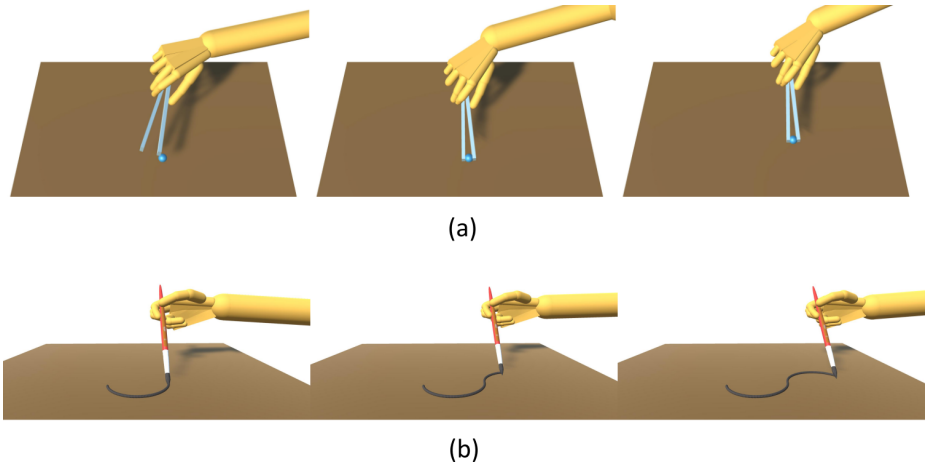


Figure 5.19: Top: relocating a ball using a pair of tongs. Bottom: tracing a curve using a brush.

first move ends. The controller can grasp the top box during falling according to the plan made at the beginning of the second move.

Generally speaking, the success rate of chopsticks-based grasps depends on many factors, such as the weight of the objects, the shape of the chopsticks, the hand gripping styles etc. Here we report the grasping success rates with respect to the object-chopsticks friction coefficients for consecutive relocation tasks. More specifically, we use our motion planner to generate 50 sequential grasp-and-move tasks, each containing eight random objects to be relocated. The friction coefficients are set to one of three values around the MuJoCo default 1.0. The success rates are plotted in Figure 5.21. Interestingly enough, we find our hand controllers always succeed for the first few grasps. Failures only occur at the latter stage. Objects with smaller friction coefficients are indeed more difficult to be grasped. Another contributing factor to the failures is likely to be the soft contact model implemented by MuJoCo as discussed in Section 5.7.6, as we observe failure cases where the chopsticks gradually slip out of the hand after a few moves.

To handle failure cases and bigger perturbations, replanning by the motion planner according to the latest scene configuration is needed. We note that human-level performance is not 100% for chopsticks-based grasping tasks either, especially for tiny or slippery objects, so replanning is also seen in real-life scenarios. We can simulate such behavior by deliberately adding Gaussian noises $\mathcal{N}(0, \sigma^2 I)$ to p_{chop} of the chopsticks configurations computed by the grasping model. $\sigma = 3mm$ in our experiments. The hand controller tracks the noisy motion plans, which eventually leads to failures in sequential relocation tasks. Upon such failures, we replan again with no added noise. Then the object can be grasped successfully.

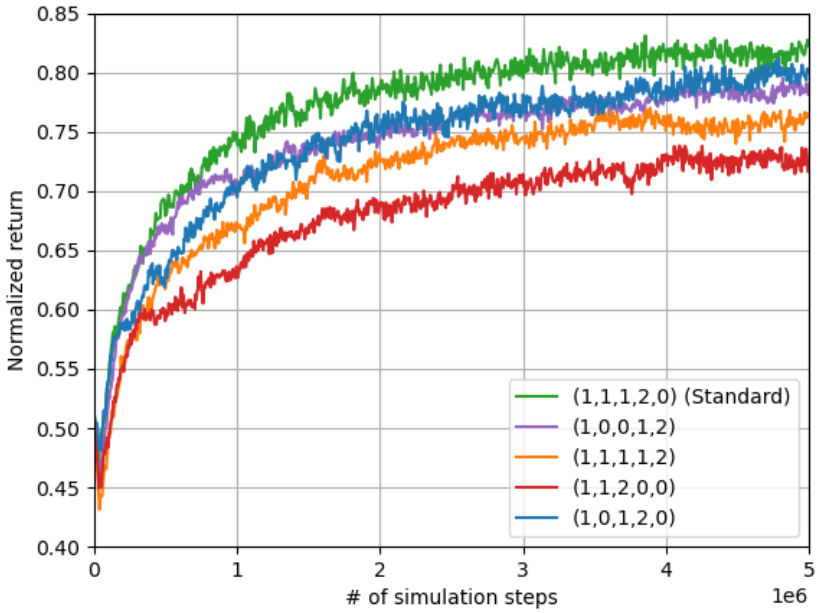


Figure 5.20: Training curves of learning basic chopsticks maneuvers using different gripping styles. The standard style is indeed the most efficient way of using chopsticks.

5.7.6 Ablation and Comparison

We conduct ablation and comparative studies to justify and validate two major components of our control and learning framework: the hierarchical control structure, and the gripping pose optimization module. Additionally, we report the effect of related MuJoCo contact dynamics parameters.

Hierarchical Control

Our high-level motion planner computes kinematic motion trajectories for the low-level hand controllers to track, which greatly simplifies the DRL reward design and improves the overall motion quality without using pre-captured example data. For system ablation without using the motion planner, we use a sparse reward to directly measure the success of a task at the very end of the task. More specifically, we return a reward 1.0 if the object can reach the target position within $0.5cm$ in distance, and a reward 0.0 otherwise. We use exactly the same PPO algorithm with the same parameter and hyperparameter settings. As shown in Figure 5.22, this scheme does not learn at all, with close to zero returns after 100 million simulation steps. The learned hand controller just randomly moves the fingers and cannot even hold the chopsticks firmly, let alone using chopsticks to grasp objects.

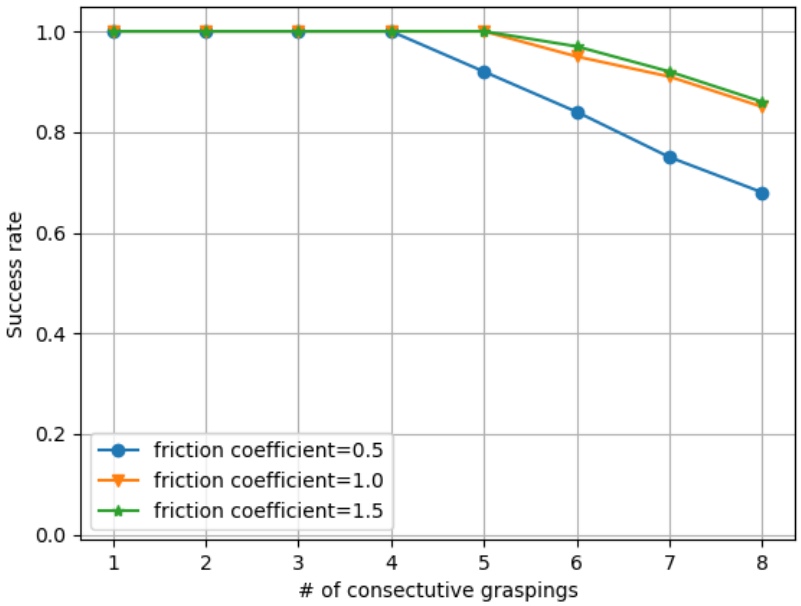


Figure 5.21: The task success rate with respect to the number of consecutive object relocations performed. We test three object-chopsticks friction coefficients, around the default MuJoCo friction coefficient 1.0.

Gripping Pose Optimization

We ablate the gripping pose optimization with two alternatives: policy learning with the default T-pose and a handcrafted gripping pose. The T-pose has no preferred finger-chopsticks contact relationships, so the reward term r_{contact} is not used in DRL training. The handcrafted gripping pose, as shown in Figure 5.23 left, is tuned to follow reference pictures in [134]. As shown in Figure 5.22, policy training with T-pose progresses poorly. The final policy cannot even hold the chopsticks steadily. Policy learning with the handcrafted pose shows better performance, but still converges slower and has inferior final policy, compared to learning with our BO optimized pose as shown in Figure 5.23 right.

Contact Dynamics

Chopsticks skills are contact-rich control problems for which contact dynamics plays an important role in the success, robustness, and quality of the learned controllers. We train our hand controllers and synthesize the demo animations using the MuJoCo default simulation parameters. However, the synthesized motions contain visible artifacts including penetrations of the chopsticks into the fingers, and objects being moved and stacked looking too soft or sticky. We therefore explore more settings for a few contact dynamics parameters in MuJoCo.

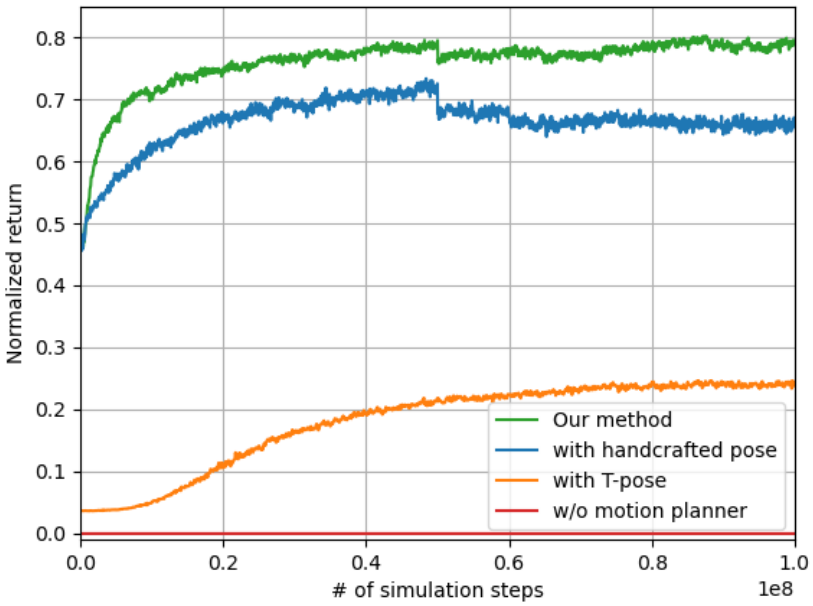


Figure 5.22: Learning curves for policies trained with different ablations. Without the high-level motion planner, the policy does not learn at all. Using the default T-pose shown in Figure 5.6, the policy learns very poorly. Our system can also learn using a handcrafted gripping pose, but using the BO-optimized pose performs better. Note that the performance drop around the middle of the training is due to lengthening the training episodes gradually from then on.

Contacts in MuJoCo are inherently soft and modeled as a convex optimization rather than the conventional LCP (Linear Complementarity Problem) [215]. The default parameters are also set to prefer soft contacts to encourage stable simulations. In particular, the parameter *solimp* parameterizes the impedance, which controls how "hard" the contact constraints are. The default setting of *solimp* is $(0.90, 0.95, 0.001)$. The valid range of the first two values of *solimp* is $[0.0001, 0.9999]$. The larger they are, the harder the contact constraints will be. In Figure 5.24 we show the policy learning curves for different settings of *solimp*. Policy training with the default setting does perform the best, although at the cost of visible penetrations. However, it does not mean that the more penetrations allowed, the better the policy will learn. Policy learning with (0.85,0.9,0.001) actually fails due to large penetrations between the fingers and chopsticks.

A related issue caused by MuJoCo's soft contact model is that gradual contact slip cannot be avoided, even with large friction coefficients. There is a parameter *impratio* that determines the ratio of frictional-to-normal constraint impedance for friction cones. Settings larger than 1 cause friction forces to be "harder" than normal forces, having the general effect of reducing slip. We train the low-level control policy for the standard gripping style

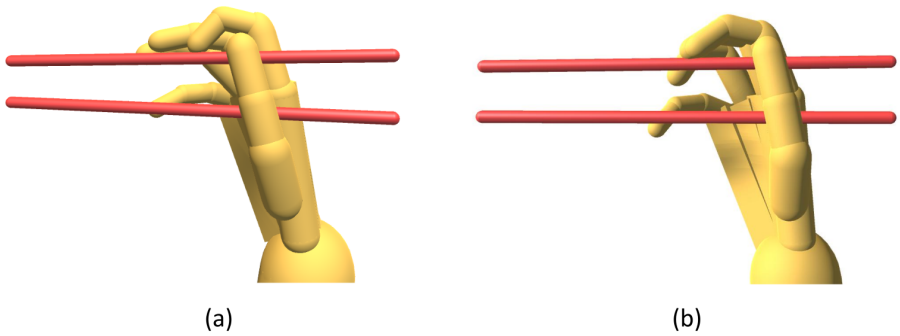


Figure 5.23: A handcrafted pose (left) vs. our optimized gripping pose (right) for the standard gripping style.

with different *impratio* values centered around its default value 1.0. As shown in Figure 5.25, policy learning with mid-range values have similar final performance, while policy learning with too small or too large values shows worse performance.

5.8 Conclusion and Discussion

We have presented a physics-based learning and control system for object manipulation using chopsticks. This is a challenging task that involves complex interactions between the hand, chopsticks, and objects. Our key insight is to tackle the problem by solving two subproblems: finding good grips to hold the chopsticks stably first, and then using them to grasp and relocate objects. More specifically, the gripping pose is a strong determinant of successful chopstick use later, and we use Bayesian optimization to efficiently explore the gripping pose space. Candidate gripping styles are evaluated through deep reinforcement learning on basic chopsticks maneuvers. After good gripping poses are found through BO and DRL, the actual object relocation using chopsticks is learned in two stages. A high-level motion planner first generates kinematic trajectories for the chopsticks and hand to satisfy task requirements. Then low-level hand control policies are trained to track the generated motion plans using a chosen gripping pose through DRL again. Whenever possible, we adopt common design choices, such as PPO for DRL learning, and MuJoCo with default parameters for simulation.

We have demonstrated physics-based object relocation using chopsticks in diverse gripping styles for multiple hand morphologies, for the first time in the literature. Our framework does not need any motion capture data or human demonstrations, and is easily applicable to virtual creatures and robotic manipulators. The framework is also transferable to learning other tools such as tongs, tweezers, pens and brushes, with minor tool-specific changes for style pruning and trajectory generation. Manipulating scissors without actually cutting things is also doable, although real cutting with scissors requires additional components such as a thin-shell simulator and a deformable object motion planner.

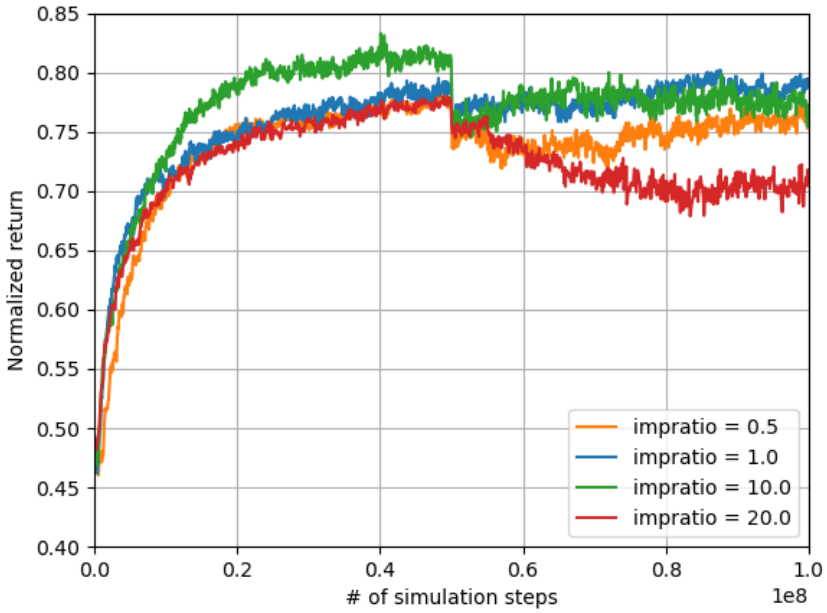


Figure 5.24: *solimp* controls the contact constraint impedance. The default value performs the best in terms of policy learning.

We have focused on learning visually robust policies with a limited number of quantitative tests. We hope our work will stimulate future work on more systematic investigations on factors and variations that can affect the success rates of chopsticks-based object manipulations tasks, such as object shapes and sizes, object weights, material and friction properties of the chopsticks and objects, chopsticks geometries, and gripping styles. Different robustness metrics other than the task success rates are also worth exploring.

There are some limitations of our current solution that we would like to further investigate in the near future. We use a 7-DoF parallel gripper to simplify the kinematic planning of chopsticks movements. However, some gripping styles allow chopsticks to move more freely, such as the "dangling stick" and the "Italian" style shown in Figure 5.2. To reproduce such styles, we need to model the chopsticks as a more versatile gripper with higher DoFs in the motion planner. Another possible solution is to learn a low-dimensional chopsticks motion manifold, similar to the approach described in [70] for learning a full-body human motion manifold, and train the grasping model in the latent space. Such a data-driven approach may produce more natural and diverse chopsticks manipulations, at the cost of pre-capturing all the chopsticks skills beforehand.

Our learned controllers cannot use chopsticks to successfully manipulate objects for an infinitely long time. The chopsticks gradually slip in hand after relocating objects repeatedly, and this destroys the good grips. Most likely this is caused by the MuJoCo contact dynamics models. As far as we know, no rigid body contact dynamics solver can completely avoid

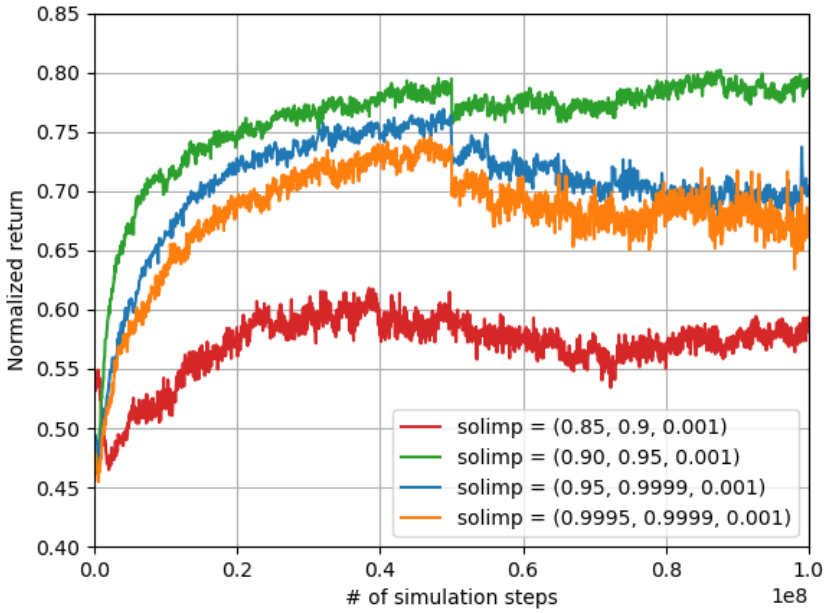


Figure 5.25: *impratio* controls the frictional-to-normal constraint impedance. Mid-range values perform the best in terms of policy learning.

unstable contacts or contact slips. Modelling fingers and palms as truly soft bodies as in [80] may lead to more robust skills. In the long run, we would like to learn to adjust chopsticks grips with dexterous finger gating maneuvers, such as picking up chopsticks from a table top, changing grips, and moving chopsticks up and down in hand.

Currently our system runs at interactive rates but not realtime, bottlenecked at the motion planner. An idea for improvement is to replace the trajectory optimization component with a neural network model, which may be trained jointly with the low-level control policies, similar to the ideas of [165] and [265]. We would also like to explore the possibility of combining gripping pose optimization and control policy learning into one iterative learning pipeline, which alternates the training between optimizing gripping poses and improving low-level control policies, evaluated on the same set of object relocation tasks. Such a scheme may potentially achieve better learning efficiency and control performance.

We have only touched the tip of the iceberg for physics-based chopsticks skills. How to eat and cut noodles with chopsticks? How to flip a piece of meat using chopsticks? How to beat an egg with chopsticks? Is it easier to use Chinese, Japanese, or Korean chopsticks for a certain type of food? Modeling and planning with soft or amorphous materials as in [267] is a must. Last but not least, we would like to transfer our chopsticks controllers to real anthropomorphic robot hands using "Sim2Real" techniques such as domain randomization and adaptation similar to [162, 167].

Chapter 6

Efficient Hyperparameter Optimization for Physics-based Character Animation

As we discussed and validated in chapter 4 and 5, the emergence of stylized motor skills often depends on some low-dimensional structures or patterns utilized in the motion tasks, such as take-off states for high jumps and gripping poses for chopsticks manipulations. These low-dimensional structures can be viewed as hyperparameters of skill learning systems and are optimized by some hyperparameter optimization algorithms like Bayesian Optimization. Comparing to these uncommon hyperparameters, people are more familiar with hyperparameters like learning rate and batch size for neural network training. These hyperparameters have a large effect on deep learning systems' performance. How to find suitable hyperparameters for deep learning systems is an important topic that has received more and more attention from machine learning communities. Many hyperparameter optimization algorithms have been proposed for supervised learning models and achieved great success in accelerating the training process or improving the final performance of models. However, hyperparamter optimization for deep reinforcement learning is less explored, especially for complex humanoid control and animation tasks. A common practice in physics-based character animation is to tune hyperparameters of DRL systems manually using expert knowledge. However, tuning hyperparameters for DRL systems usually requires repetitive training of control policies and can be extremely computationally expensive. As a result, an efficient and powerful hyperparameter optimization algorithm for DRL-based animation systems is in great need. In this chapter, we introduce a novel multi-fidelity Bayesian Optimization algorithm to optimize hyperparameter of DRL-based motor skill learning systems. Our algorithm significantly outperforms state-of-the-art hyperparameter optimization methods applicable for physics-based character animation. In particular, we show that hyperparameters optimized through our algorithm result in at least 5x efficiency

gain comparing to author-released settings in DeepMimic. The accompanied video of this work is available at the link [1].

6.1 Introduction

Physics-based character animation has seen many progresses in recent years, especially with the application of deep reinforcement learning, as we mentioned in chapter 2. Despite the impressive performance those DRL-based animation systems have demonstrate, it is difficult for a novice graduate student to reproduce the performance of such systems, without knowing the exact value of the hyperparameters involved. As a result, more and more researchers have released their codes for better reproducibility. However, a single change of one hyperparameter may totally ruin the performance, and sometimes even the convergence, of such learning algorithms. How the original authors found the working set of hyperparameters remains as an art and mystery. Improving upon prior work, even with released code, thus remains challenging, as any modification to the training algorithm may require a new set of hyperparameters to work well.

Automatic hyperparameter optimization thus is in great need. Hyperparameters, in the narrow sense, refers to parameters used to control the learning process in machine learning algorithms. They are usually determined before the learning starts. In contrast to hyperparameters, regular parameters are derived or optimized during the learning process. In this work, we refer parameters external to the learning algorithm that are determined prior to the learning as hyperparameters. For example, the learning rate, batch size or the choice of optimizer are hyperparameters of DRL algorithms. The morphology of a virtual character in motor skill learning tasks are also hyperparameters. To date, a common practice in this field is to manually select hyperparameters for various character control algorithms. Behind the scenes, maybe random search or simple grid search are used for semi-automatically choosing hyperparameters. However, better hyperparameter search algorithms are needed in physics-based character control to handle two main challenges: first, the evaluation of new hyperparameters usually involves re-learning of a controller, which is a expensive black box function. Second, the number of hyperparameters is usually large and result in the curse of dimensionality.

Bayesian Optimization (BO) is a promising candidate for hyperparameter optimization. It is a sequential design strategy for global optimization of expensive-to-evaluate black-box functions that do not assume any functional forms [83, 199, 88]. BO is often used to search for hyperparameters of deep neural networks for supervised learning tasks. We also use it to optimize the gripping pose for chopsticks manipulation skills. Traditional BO algorithms optimize objective functions by directly evaluating the expensive black box objective func-

[1] `https://www.youtube.com/watch?v=tU0xnVPe7Gg`

tions [83, 199]. We refer to these methods as single-fidelity BO methods (SFBO). However, for many physics-based character control applications, SFBO is inefficient due to the heavy computation cost of function evaluation including physics-based simulation and motor skill learning. Multi-fidelity Bayesian Optimization (MFBO) accelerate the optimization process by employing cheap approximations of objective functions in early optimization stages [88, 100, 197, 208]. Such MFBO algorithms typically use fewer training iterations on smaller datasets as cheap approximations to the original objective functions, and work well for supervised learning models.

For physics-based character control, however, existing MFBO methods do not work well. As we analyze in section 6.3.1 and 6.3.2, training with fewer iterations is not a good fidelity criterion for physics-based character control, and the multi-fidelity objectives do not have the desired properties across the fidelity dimension required by existing MFBO methods, such as BOCA [88] and FABOLOUS [100]. Motivated by our hand tool manipulation project where we use simple chopsticks maneuvers to evaluate the quality of input gripping poses, we propose a novel and efficient Bayesian Optimization algorithm: Curriculum-based Multi-Fidelity Bayesian Optimization (CMFBO) to optimize hyperparameters for DRL-based character control or animation systems. CMFBO employs easier motor skill learning tasks as low-fidelity optimization objectives. Task difficulties are organized and scheduled by a curriculum. For example, a curriculum that gradually increases the episode length can help characters to perform longer and longer skills [161], and a curriculum that gradually reduces the hand-of-God assisting forces to help simulated characters to learn to locomote [262]. Easier tasks are faster to evaluate thus enabling efficient search space pruning. Control policies learned at easier tasks can be transferred to more difficult ones to further reduce evaluation cost. Hyperparameters optimized by CMFBO may significantly outperform those tuned by experts and optimized by state-of-the-art methods. For instance, at least 5x efficiency gain is obtained with our optimized hyperparameters on DeepMimic [161], compared to author-released settings.

To summarize, we

- introduce Bayesian Optimization into physics-based character control for general hyperparameter optimization;
- propose an efficient algorithm CMFBO for hyperparameter optimization of challenging motor learning tasks based on deep reinforcement learning;
- systematically compare and evaluate SFBO, state-of-the-art MFBO, and CMFBO on two physics-based character animation problems: morphology optimization, and automatic tuning of DeepMimic training hyperparameters

6.2 Related Work

In this section we review some related works on applications of parameter optimization on computer graphics and hyperparameter optimization for deep learning. Since in our experiments, we also use hyperparameter optimization algorithms to optimize morphology parameters of a bipedal robot, we also investigate some previous methods on morphology optimization for robot control. Lastly we provide a brief review of curriculum learning, which is the core component of our proposed method.

6.2.1 Parameter Optimization in Computer Graphics

Parameter optimization or tweaking is a common task in computer graphics, such as tuning weights of SIMBICON-type feedback locomotion controller [257, 226]. Multiple derivative-free optimization algorithms, such as Covariance Matrix Adaptation (CMA) [59, 226] and Bayesian optimization, have been used to optimize such parameters where the objective functions are black box functions and no derivative information is available. Some methods further combine BO with user evaluations to achieve desired visual effects, such as bidirectional reflectance distribution function (BRDF) design [18] and smoke animation tuning [17]. More recently, BO is applied to low-dimensional sub problems of various graphics applications [104], such as color enhancement, and human body geometric modelling [127].

We also adopt a Bayesian Optimization framework, but for hyperparameter optimization. To the best of our knowledge, the only previous work on hyperparameter optimization in graphics is [219], where parameters of image processing hardware, such as threshold values in the denoising module, are optimized with a non-Bayesian approach. Hereafter we will focus on hyperparameter optimization literature from machine learning and robotics.

6.2.2 Hyperparameter Optimization for Deep Learning

Recently with the development of deep learning, deep neural network models have demonstrated their effectiveness on various tasks, such as image and speech recognition, content generation, etc. A set of hyperparameters of deep neural networks need to be determined before the training process, such as the architecture of neural networks and learning rates of training algorithms. These hyperparameters affect these deep neural networks' performance a lot. Manually tuning hyperparameters of deep neural networks is non-trivial and could be time-consuming, especially when the neural network is huge and requires heavy computational resources to train. To ease and accelerate the development of deep learning models, efficient and automatic hyperparameter optimization algorithms for deep learning models are in great demand. To this end, many research efforts have been invested in automatic hyperparameter optimization for deep neural networks. These methods have successfully accelerated training processes and improved the final performance of deep learning models. For example, [100] proposed a novel BO framework to find high-quality settings of deep

neural networks $10 - 100$ times faster than other hyperparameter selection methods. [89] combined optimal transport theory with BO to achieve efficient searches of neural architectures. In addition to these single-fidelity BO methods, a lot of Multi-Fidelity Bayesian Optimization (MFBO) algorithms have been designed to improve the data efficiency further. MFBO utilizes cheap approximations of the original objective functions to guide the hyperparameter optimization process. These cheap approximations are less expensive to compute and can still provide useful information about the candidate hyperparameters. For example, [87] used deep neural networks' performance on smaller datasets as a low-fidelity approximation of the network's performance on the original dataset to filter out unpromising hyperparameters. Despite these recent BO methods' effectiveness on hyperparameter optimization, they mainly focus on supervised learning tasks. Hyperparameter optimization for deep reinforcement learning using BO is less investigated. Compared with supervised learning, the training process of deep reinforcement learning is noisy and not stable, making choosing suitable hyperparameters even more difficult. Hyperparameter optimization for simple DRL benchmark systems, such as a cartpole model in OpenAI gym [19], with standard DRL algorithms, such as A2C [140], has been mostly investigated [147, 148, 99, 64]. However, in physics-based character animation, we study the control of high dimensional character models, which is a much more expensive and challenging task requiring specialized DRL algorithms such as DeepMimic to achieve natural-looking skills. Hyperparameter optimization for such complex systems is more challenging and could be computationally prohibitive for methods like [147, 148].

6.2.3 Morphology Design

Morphology design and optimization is a problem in both computer animation and robotics [155, 193, 171, 117, 160, 16, 224, 50, 7, 198, 54, 229, 239, 114, 76, 75, 271, 132, 97, 247], which aims to find suitable morphology parameters for robots to optimize their motor abilities. Traditional model-based methods require accurate dynamic models of movements, and only work for specific types of control algorithms such as trajectory optimization [198, 54]. Some recent morphology design methods work with DRL-based control [186, 53, 128]. These methods guide optimization in morphology space based on gradient estimation [186, 53], or performance prediction of unseen morphology through a learned value network [128]. We note that none of these methods address the issue of expensive evaluation per design. With effective search space pruning through cheap evaluation on easier tasks, our method significantly outperforms some of these methods in terms of sample efficiency.

6.2.4 Curriculum Learning

Curriculum learning (CL) is a learning method where task difficulty gradually increases during training [14]. Easier tasks are cheaper to accomplish and serve as good initial solutions for more difficult tasks. CL has been widely used in character animation to improve the

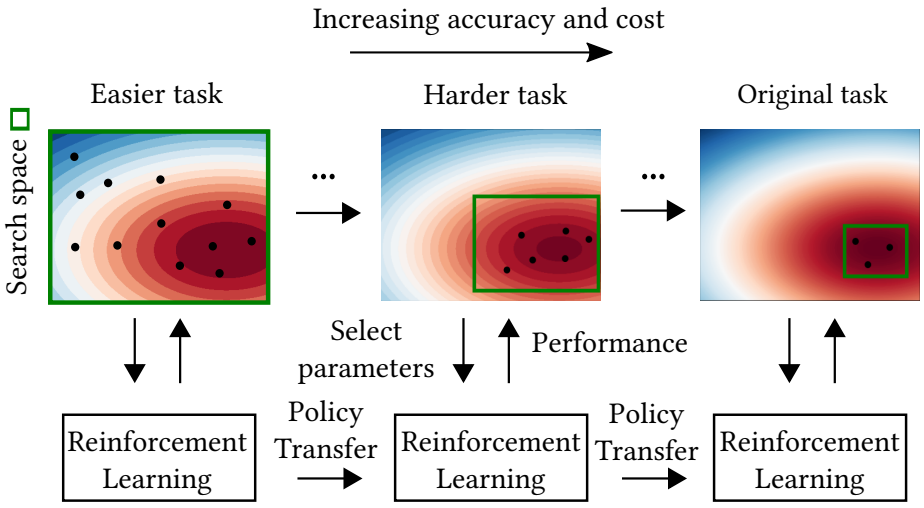


Figure 6.1: Conceptual illustration of CMFBO

performance and efficiency of motor learning tasks [222, 256, 241, 90]. [161, 102] employed a time-based curriculum with increasing episode length to help the character perform longer and longer skills. [262] applied hand-of-God assistance forces at robot torso and decreases the force gradually during training to encourage the emergence of natural gaits. More recently, [245] used CL to demonstrate successful training of stepping-stone locomotion controllers.

6.3 Method

We formulate hyperparameter optimization in a physics-based animation system as a black-box optimization problem:

$$\arg\max_{x \in \mathcal{A}} f(x) \tag{6.1}$$

where $\mathcal{A}$ is the hyperparameter space. $f(\cdot) : \mathcal{A} \to \mathcal{R}$ is the objective function, i.e. performance of control policy given hyperparameter x. The evaluation of $f(\cdot)$ involves DRL training, thus is costly, noisy and cannot be computed in closed forms.

Figure 6.1 illustrates a conceptual overview of our approach based on MFBO. We first define our multi-fidelity function based on a curriculum (section 6.3.1). During optimization, we use lower-fidelity cheap approximations to locate the promising region of hyperparameters through the proposed progressive acquisition function (section 6.3.2). Policies learned at easier tasks are transferred to difficult tasks for more efficient evaluation (section 6.3.3).

6.3.1 Curriculum-based Multi-Fidelity Functions

The main challenge of hyperparameter optimization for physics-based character control tasks comes from the fact that the performance evaluation function $f(x)$ given hyperparameter x is highly computationally expensive. Even if BO is designed for black-box function

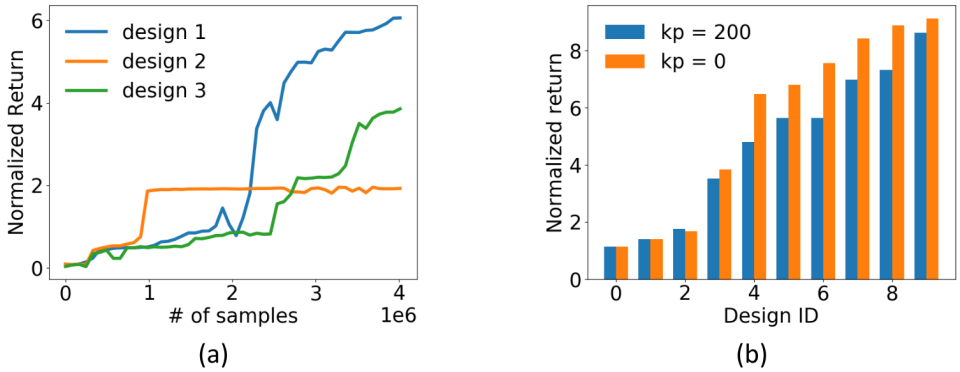


Figure 6.2: (a): Learning curves of locomotion controllers with different morphology designs. (b): Performance of ten morphology designs on two difficulty levels. Task difficulty is determined by the strength of hand-of-God assistance forces parameterized by proportional gain *kp* of the assistance stable PD controller.

optimization with minimum samples, the entire optimization procedure still requires a prohibitive amount of computational resources.

We propose to solve this problem with a multi-fidelity approach on top of Bayesian Optimization, where low-fidelity cheap evaluations are utilized for efficient search space pruning. This procedure crucially depends on an accurate fidelity criterion being able to distinguish good hyperparameters from bad ones with lower-fidelity approximations. To the best of our knowledge, number of training iterations is the only fidelity criterion from existing literature applicable to our problem. Though number of training iterations shows its effectiveness for supervised learning tasks [88, 100, 195], performance estimated through early stopping can be deceptive for challenging DRL character control tasks. Early stages for DRL training can be noisy [57, 47], and good hyperparameters do not always guarantee quick convergence [148]. We illustrate this problem in the context of finding optimal morphology hyperparameters for character locomotion. We show the performance of three morphology designs in Figure 6.2 (a), where some good morphology designs show worse performance in early training stages.

To mitigate this problem, we propose the use of curriculum-based task difficulty as a new fidelity criterion tailored to physics-based character control settings. Task difficulty can be parameterized by a single continuous scalar variable z. Multi-fidelity objective function is therefore defined to be $f(x, z)$. Note that $f(x, z_{max})$ is the original objective function. $f(x, z)$ is more accurate and requires more computational resources as z gets closer to z_{max}. Similar to single fidelity BO, we use GP to model multi-fidelity function $f(x, z)$. The kernel function generalizes to:

$$k((x, z), (x', z')) = k_{SE}(x, x')\ k_{SE}(z, z') \tag{6.2}$$

The factorized kernel adopts the multiplication form based on the fact that: if x and z are close to x' and z' respectively, $f(x, z)$ correlates strongly with $f(x', z')$.

We validate the effectiveness of using task difficulty as fidelity criterion through an example in Figure 6.2 (b), where ten morphology parameters are evaluated on two different task difficulty parameterized by the strength of virtual hand-of-God assistance forces. With fidelity criterion defined with task difficulty, relative performance of hyperparameters across different fidelity levels are well preserved. We detail our specific curriculum settings for tasks in section 6.4.

6.3.2 Progressive Acquisition Function

Given a properly designed multi-fidelity objective function, an effective acquisition function is still in need to achieve efficient hyperparameter optimization in physics-based character control tasks. In each iteration, multi-fidelity acquisition function $a((x, z), D_t)$ determines which point x in hyperparameter space is going to be evaluated next, and which fidelity level z the evaluation is going to be performed on. An effective multi-fidelity acquisition function is expected to generate queries such that the original objective can be optimized with much fewer samples by inferring optimality information through low-fidelity cheap approximations.

Existing MFBO methods like BOCA [88] and FABOLAS [100] require a well-fitted surrogate model for accurate estimation of original objective value from lower-fidelity approximations. Mathematical assumptions on the shape of fidelity dimension are usually necessary to achieve this. For example, BOCA assumes the fidelity dimension is flat while FABOLAS assumes it to satisfy quadratic form. However, due to stochasticity and complexity of DRL training, performance difference of hyperparameters on different fidelity levels can be unpredictable, as shown in Figure 6.2 (b). A GP trained on such DRL performance data will learn a small length scale, which degrades the accuracy of high fidelity value prediction through a reasonable number of lower-fidelity samples.

We attempt this problem with a different approach. Instead of relying on low fidelity samples to infer high-fidelity function value, we propose to use them simply for locating promising regions. The motivation is based on the key observation that though performance difference of hyperparameters on different fidelity levels can be unpredictable, optimal regions of objective function on different fidelity levels highly overlap, as shown in Figure 6.2 (b). Search space for a high-fidelity objective can therefore be greatly pruned by simply exploring around optimal regions estimated at lower fidelity levels.

Specifically, we propose a novel progressive acquisition function, which progressively conducts optimization from low fidelity to high fidelity levels. At iteration t, target fidelity z_t is chosen to be the least expensive while most informative one until we reach the highest. This is performed through an iterative procedure. Initializing z_t with z_{min}, we repeatedly increase z_t by a small step α and re-compute best candidate point x_t for new z_t. The pro-

cedure terminates when our model shows insufficient knowledge at (x_t, z_t) with uncertainty $\sigma_t(x_t, z_t)$ exceeding threshold ϵ. In our implementation, we choose α to be equal to the length scale of the fidelity kernel l_z, since function values from GP surrogate model strongly correlate along fidelity dimension within l_z proximity. z_t should be increased only when we have sufficient knowledge at (x_t, z_t). ϵ can therefore be set conservatively to a small value negligible comparing to the objective function value range in promising regions. In our experiments we set ϵ to 0.02.

Given a specified fidelity level z_t, the best candidate point x_t is chosen by optimizing a variant of the upper confidence bound (UCB) utility function defined as:

$$a((x, z_t), D_t) = \mu_t(x, z_t) + \beta^{\frac{1}{2}}\sigma_t(x, z_{max}) \tag{6.3}$$

where $\mu_t(x, z_t)$ favors high-performing regions and $\sigma_t(x, z_{max})$ favors uncertain regions of the original objective. We set β to $0.2d\log(2t)$ following suggestions from [88, 199], where d is the dimension of the hyperparameters and t is the iteration index. The optimization starts from $z_t = z_{min}$ to find initial promising candidates across the hyperparameter space. When $z_t > z_{min}$, optimization of $a((x, z_t), D_t)$ is restricted to be close to known high-utility regions obtained from lower-fidelity levels, which enables efficient search space pruning. This can be achieved by simply applying a gradient-based local optimization method like L-BFGS [121] on $a((x, z_t), D_t)$, with solutions of $\arg\max a(\cdot)$ at lower-fidelity levels serving as initial guesses. The optimization of the acquisition function does not have to be perfect, since we only care about promising points instead of optimal ones. Our algorithm works well with our default choices for α, β and ϵ without manual tuning. We summarize this procedure in Algorithm 3 for ease of re-implementation.

Algorithm 3: Progressive Acquisition Function

Input: Iteration index t, fidelity kernel length scale l_z, observation dataset D_t and uncertainty threshold ϵ

Output: Best candidate pair (x_t, z_t)

$z_t \leftarrow z_{min}$

$x_t \leftarrow \arg\max_x a((x, z_t), D_t)$ using L-BFGS with multiple random start points

while $z_t < z_{max}$ *and* $\sigma_t(x_t, z_t) < \epsilon$ **do**

  1. $z_t \leftarrow \min(z_t + l_z, z_{max})$;

  2. $x_t \leftarrow \arg\max_x a((x, z_t), D_t)$ using L-BFGS with x_t as an initial solution ;

end

Return (x_t, z_t)

6.3.3 Policy Transfer

Progressive acquisition function enables the use of transfer learning to speed up evaluation on most (x_t, z_t) pairs where $z_t > z_{min}$. This comes from the fact that progressive acquisition

Algorithm 4: Curriculum-based MFBO

Input: Multi-fidelity function $f(x, z)$ and the maximum allowed computational cost N_{total}

Output: Observed function maximum $y^\star$ and corresponding $x^\star$

$N_{\text{current}} \leftarrow 0, D_0 \leftarrow \varnothing$, the Gaussian Process is initialized randomly.

for $t = 1, 2, ...$ **do**

1.Choose (x_t, z_t) by progressive acquisition function optimization (see algorithm 3);

2.Evaluate $f(x_t, z_t)$ to get performance y_t and corresponding computational cost N_t;

3.$D_t \leftarrow D_{t-1} \cup ((x_t, z_t), y_t)$ and update parameters of the Gaussian Process using D_t;

4.$N_{\text{current}} \leftarrow N_{\text{current}} + N_t$;

5.**if** $N_{current} \geq N_{total}$ **then**

break;

end

$t^\star \leftarrow \arg\max_t y_t$, $y^\star \leftarrow y_{t^\star}$, $x^\star \leftarrow x_{t^\star}$;

Return $y^\star$ and $x^\star$

function queries z_t progressively, and only looks for x_t within a common promising region. Therefore, as long as $z_t > z_{min}$, a previously queried pair (x_i, z_i) with $i < t$ usually (if not always) exists such that (x_t, z_t) is close to (x_i, z_i). Pre-trained policy on (x_i, z_i) can therefore serve as a warm start for training on (x_t, z_t).

In practice, (x_i, z_i) can be acquired by traversing all previously visited (x, z) pairs to find the closest one to (x_t, z_t) measured by kernel function correlation $k((x_t, z_t), (x_i, z_i))$. Theoretically, negative transfer may happen but we do not observe any in our experiments. We show in our experiments that the use of transfer learning saves a significant number of samples during CMFBO optimization. Algorithm 4 summarizes the pipeline of CMFBO.

6.4 Experiments

We validate CMFBO on two physics-based character control tasks: morphology optimization and hyperparameter optimization of DeepMimic. For each task, we discuss its specific curriculum setting and objective function, followed by experiment results, comparisons and ablation studies. In our implementation, we use Gaussian process from GPy [51] and L-BFGS [121] from Scipy [225]. All physics simulation is performed on Pybullet [29]. We report the performance statistics on a desktop with i7-7800x CPU (12 threads). Each experiment is run 3 times independently with different random seeds.

6.4.1 Morphology Optimization

The locomotion capability of a creature is intimately coupled with its morphology. Elaborately designed morphology of simulated characters encourages the emergence of natural locomotion [50]. In this experiment, we optimize character morphology for learning fast and low-energy locomotion. Our simulated character has 15 torque-controlled revolute joints. Morphology hyperparameter x is defined to be a 12-dimensional vector that scales the length and radius of character links within a limited range, as shown in table 6.1. Link mass is uniformly scaled according to its volume. Sagittal symmetry is imposed on legs of the character.

We use a feed-forward policy network with three fully connected layers, each with 128 units using tanh activation. The critic network shares the same architecture with the policy. Policy is trained with the PPO algorithm [190]. We set discount factor $\gamma = 0.95$ and $\lambda = 0.95$ for both $TD(\lambda)$ and $GAE(\lambda)$. Learning rates of both policy and critic network are set to 3×10^{-4}. In each training iteration, we sample 4096 state-action tuples with 10 paralleled environments. Batch size for policy update is set to 256.

Curriculum and Multi-fidelity Objective

We construct our multi-fidelity function based on a curriculum learning setting proposed in [262], where a stable PD controller [210] is applied to the character root to provide hand-of-God balancing forces. Task difficulty is parameterized by stiffness *kp* and damping *kd* coefficients of the stable PD controller. High gain controllers make the task easier by providing large assistance forces. Given the normalized task difficulty scalar z ranging from 0 to 1, we set $kp(z) = 200 - 200z$. *kd* is set to be equal to *kp* as proposed in [262]. The policy training starts at $kp(0)$. *kp* gradually decreases to $kp(z)$ during training according to the curriculum schedule in [262]. We refer readers to section 4.2.2 in [262] for more details. The multi-fidelity objective is the normalized return defined as:

$$f(x, z) = \frac{\sum_{t=1}^{T} r(s_t, a_t)}{T} \tag{6.4}$$

where $r(s_t, a_t)$ is the reward function, and T is the episode length fixed to 500 simulation steps (8.25 seconds).

Comparison

We compare our method with several baseline and state-of-the-art algorithms: a single fidelity BO with GP-UCB acquisition [199], a multi-fidelity method BOCA [88] and a recent DRL-based algorithm [186] (Schaff *et al.* 2019) specialized for morphology optimization. GP-UCB and [186] can be directly applied to the problem. For multi-fidelity method BOCA, We implement it following [88] and construct the multi-fidelity function parameterized by both

Length scale	Range	Radius scale	Range
Torso	[0.3, 2]	Torso	[0.5, 1.5]
Waist	[0.3, 2]	Waist	[0.5, 1.5]
Pelvis	[0.3, 2]	Pelvis	[0.5, 1.5]
Thigh	[0.3, 2]	Thigh	[0.5, 1.5]
Knee	[0.3, 2]	Knee	[0.5, 1.5]
Foot	[0.3, 2]	Foot	[0.5, 1.5]

Table 6.1: Morphology hyperparameters.

number of training iterations and our proposed curriculum-based task difficulty. Maximum allowed iteration number at fidelity z for BOCA is set to $500 + 1500z$ since most training finishes within 2000 iterations. Maximum allowed simulation steps N_{total} is set to 10^8. The optimization process takes about 18 hours for all methods.

Results

We plot the best performance over number of total simulation steps in Figure 6.3 (a) for various methods. Note that for multi-fidelity methods, best performance is only recorded when the original objective function is evaluated. Our method finds high-performing morphology hyperparameters much faster than other methods. BOCA performs worse than single fidelity method GP-UCB due to a misleading fidelity criterion defined as number of training iterations, as explained in section 6.3.1. Figure 6.4 visualizes the best morphology and the associated gaits for each method obtained from 3 individual runs. Our method learns morphology resembling human and shows fast and stable gaits. Note that though our method learns different morphology hyperparameters at different runs, proper values of essential components key to stable locomotion are consistently learned, like thigh and foot lengths.

Figure 6.3 (b) shows results of our ablation study, where we further compare our method against BOCA with our proposed curriculum-based fidelity (BOCA + CL), and our method without policy transfer (w/o transfer). With an effective fidelity criterion defined by curriculum-based task difficulty, BOCA shows a large performance gain. However, our method still outperforms BOCA by a large margin due to effective search space pruning through our progressive acquisition function. The use of transfer learning further improves efficiency of our method, with a saving of roughly 2×10^7 samples before a high-performing set of hyperparameters is confirmed on the original fidelity.

6.4.2 DeepMimic Hyperparameter Optimization

DeepMimic is a recent DRL-based framework for diverse motor skill learning. Many recent physics-based character animation systems are built on DeepMimic, such as [156, 15, 239, 236, 129]. As far as we know, this is the first work towards hyperparameter optimization for

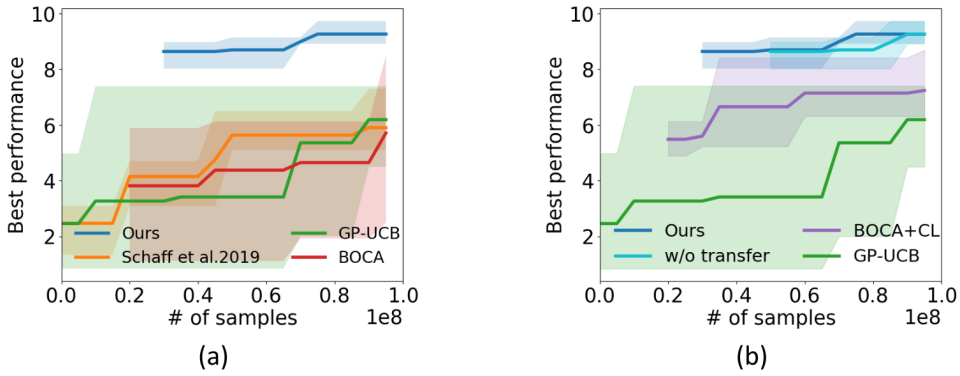


Figure 6.3: (a):Results on morphology optimization. We compare best performance over the number of training samples among CMFBO (Ours), GP-UCB, BOCA with number of iteration as fidelity and [186]. (b): Ablation study on morphology optimization. We further compare our method with BOCA with curriculum-based fidelity (BOCA+CL), and CMFBO without policy transfer (w/o transfer).

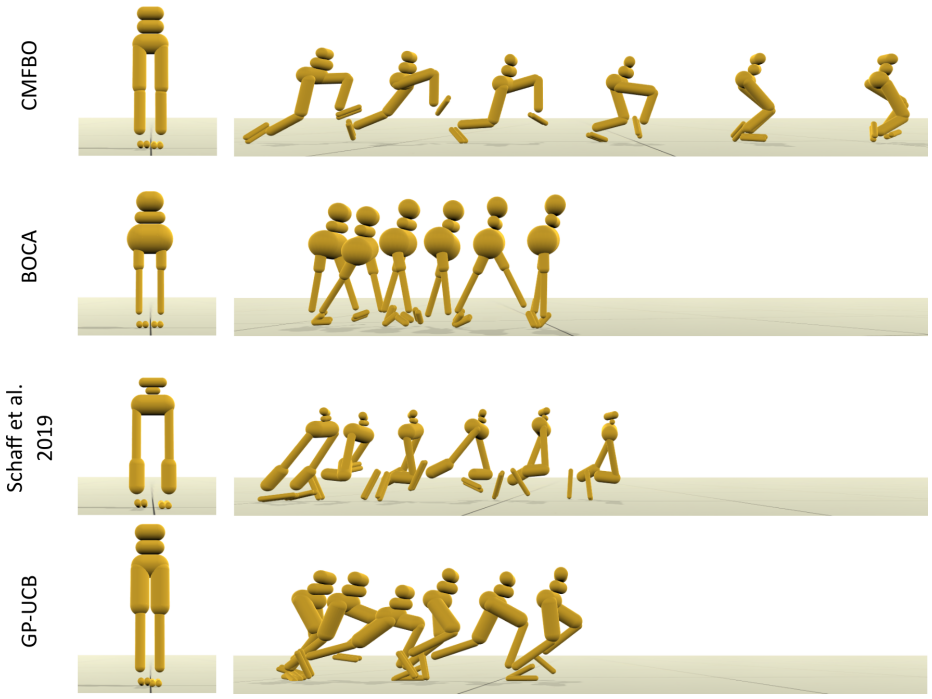


Figure 6.4: Visualization of morphology and gaits optimized by different methods.

Parameters	Range	Default
Batch size	[32,1024]	256
Policy updates per iteration	[1,10]	1
Learning rate of actor network	$[2.5 \times 10^{-6}, 2.5 \times 10^{-4}]$	2.5×10^{-6}
Learning rate of critic network	$[1 \times 10^{-4}, 1 \times 10^{-2}]$	1×10^{-2}
Weight decay	$[5 \times 10^{-4}, 5 \times 10^{-2}]$	5×10^{-4}
PPO clip rate	[0.02, 0.2]	0.2
Maximum of the gradient norm	[1, 100]	100

Table 6.2: Hyperparameters of DeepMimic

high-dimensional physics-based character control systems. In our experiments, we focus on the optimization of DRL training hyperparameters listed in table 6.2. The choices of these hyperparameters are critical for the learning performance where none of our randomly sampled 5 hyperparameters lead to successful policy training. These hyperparameters correlate with each other implicitly and are hard to be manually tuned. Some recent works [132, 239] therefore simply adopt the default DeepMimic hyperparameters directly into their systems with minor modifications. We evaluate our method on two motor skill learning tasks: walk and backflip. Our optimized hyperparameters improve DeepMimic learning efficiency by a large margin. Hyperparameters optimized for walking can be reused for other skills as well resulting in superior performance than original settings.

Curriculum and Multi-fidelity Objective

We construct our multi-fidelity function using a time-based curriculum proposed in [161, 102]. Given the normalized task difficulty scalar z, task difficulty is given by maximum episode length $T(z) = 30 + 570z$, which ranges from 1 to 20 seconds. Our multi-fidelity objective considers both policy performance and sample efficiency, which is defined as:

$$f(x, z) = \frac{\sum_{t=1}^{t=T(z)} r(s_t, a_t)}{T(z)} e^{-(\frac{N_{\text{sample}}}{C})}. \tag{6.5}$$

N_{sample} is the total number of samples consumed during training. Training is terminated when normalized expected return reaches a threshold pre-defined in original DeepMimic, or the normalized expected return does not increase for 500 iterations. C is a scaling factor set to 10^7 and 2×10^7 for walk and backflip respectively, based on the scale of usual samples required by DeepMimic on different tasks. Note that original DeepMimic has a fixed annealing-based schedule for maximum episode length during training. This schedule can be too slow on easy tasks. We switch to an adaptive schedule to avoid wasting samples.

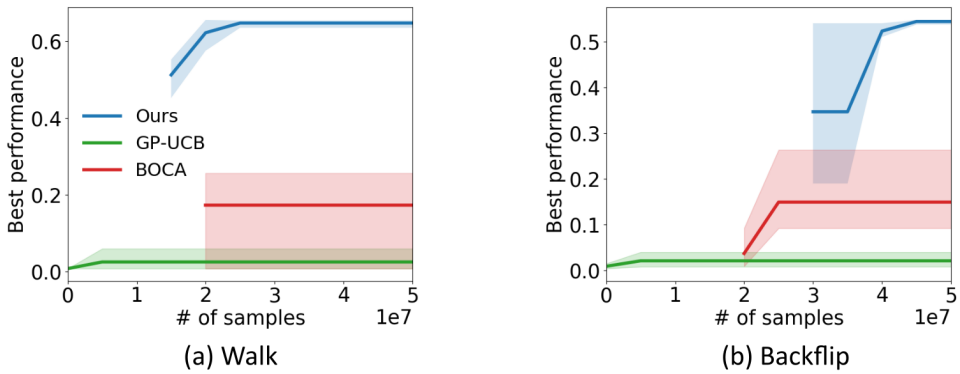


Figure 6.5: DeepMimic hyperparameter optimization results. We compare best performance over the number of training samples among CMFBO, GP-UCB and BOCA.

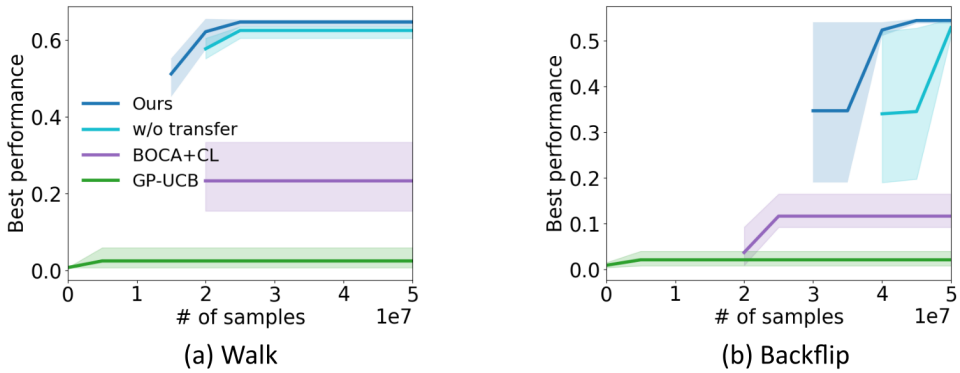


Figure 6.6: Ablation studies on DeepMimic hyperparameter optimization.

Starting from 1 second, episode length is increased by 0.5 seconds every iteration until it reaches $T(z)$, as long as the current fail rate is below 20%.

Comparison

We compare our method against GP-UCB and BOCA in our experiments. For BOCA based on number of iterations as its fidelity criterion, maximum iterations on each fidelity $N(z)$ is set to $500 + 6500z$ and $500 + 11500z$ respectively for walk and backflip. We also show comparison with BOCA using our curriculum-based fidelity in our ablation study. Maximum allowed simulation steps N_{total} is set to 5×10^7 for all methods. The optimization process takes about 16 hours for all methods.

Results

As shown in Figure 6.5, our method consistently outperforms other methods. CMFBO find different hyperparameters with different random seeds, but the performance is consistently

Parameters	Walk	Backflip
Batch size	172	48
Policy updates per iteration	8	10
Learning rate of actor network	1.68×10^{-5}	4.6×10^{-5}
Learning rate of critic network	2×10^{-3}	0.01
Weight decay	5×10^{-4}	5×10^{-4}
PPO clip rate	0.18	0.02
Maximum of the gradient norm	47.36	12.30

Table 6.3: Optimized hyperparameters for walk and backflip.

good. Table 6.3 shows the best optimized hyperparameters, which are quite different from the default ones listed in table 6.2. We select best hyperparameters optimized from each method and test their policy learning efficiency based on 3 individual runs. Results are shown in Figure 6.7. GP-UCB fails to find hyperparameters with which policies could be learned successfully. Since DeepMimic is computationally expensive, only few evaluations can be performed in such single fidelity methods. BOCA performs better than GP-UCB and find hyperparameters as good as the default ones for walk task. It also finds hyperparameters converging 1.7 times faster than default ones for backflip. Comparing to GP-UCB and BOCA, our method finds high-performing hyperparameters highly efficiently. Hyperparameters optimized by CMFBO requires $5 - 6$ times less simulation steps than default settings for successful policy learning. We note that by default DeepMimic takes about 8 and 16 hours respectively to train control policies until convergence for walk and backflip, while our method can find high-performing hyperparameters within 8 and 13 hours respectively for these tasks, even before the training of default DeepMimic finishes. We conduct the same set of ablation studies as in morphology optimization shown in Figure 6.6, where we compare CMFBO against BOCA with curriculum-based fidelity and CMFBO without policy transfer. As expected, each component is essential to achieve the impressive efficiency of CMFBO.

We show that hyperparameters optimized by CMFBO generalize to other tasks as well. Figure 6.8 illustrates the large performance gain of policy learning on run and cartwheel tasks using hyperparameters optimized by CMFBO on walk comparing with DeepMimic default settings.

CMFBO can also give us insights on the sensitivity of each hyperparameter. After optimization, the surrogate model shows small length scales along dimensions corresponding to policy updates per iteration, learning rate of critic network and weight decay. This implies that the performance is more sensitive to these hyperparameters within the given ranges.

We briefly analyze the computational cost distribution of CMFBO for DeepMimic hyperparameter optimization. The optimization of acquisition function usually finishes within three minutes, which is negligible comparing with the time required for DRL training. We

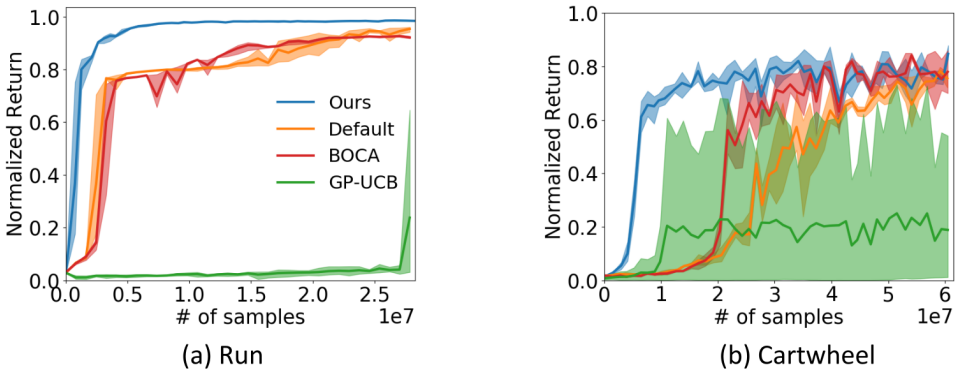


Figure 6.7: Learning curves of policies trained with hyperparameters from CMFBO, DeepMimic default settings, BOCA and GP-UCB.

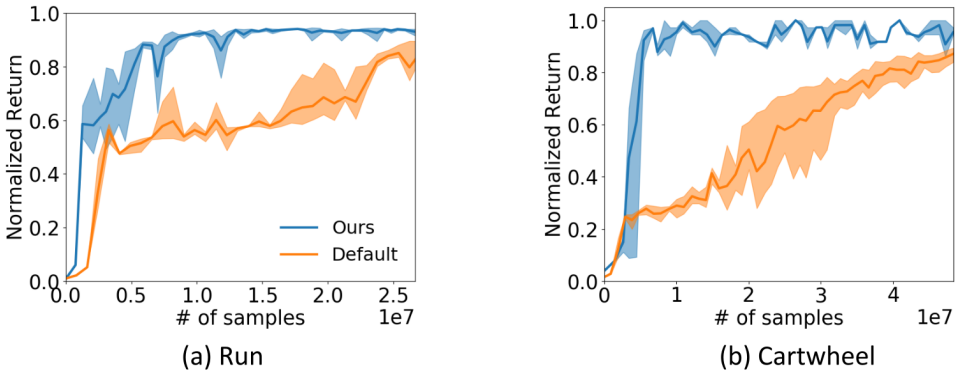


Figure 6.8: Learning curves of policy learning for other tasks with hyperparameters optimized by CMFBO on walk task comparing with DeepMimic default settings.

	Walk	Backflip
# of sampled hyperparameters	20	23
# of training samples on original tasks	6.1×10^6	1.23×10^7
# of training samples on low fidelities	4.39×10^7	3.77×10^7

Table 6.4: The distribution of training samples in CMFBO

summarize the number of training samples consumed on different fidelity levels in table 6.4, which indicates that most computational resources are allocated to low fidelity approximations. Easy tasks require less time to train than difficult ones. For instance, in walk task of DeepMimic, training at the easiest and the original difficulty level with our optimized hyperparameters take roughly 20 minutes and 80 minutes respectively.

6.5 Conclusion

In this work we introduce the hyperparameter optimization problem in physics-based character animation. We present CMFBO, a novel multi-fidelity Bayesian Optimization framework to achieve efficient optimization. Motivated by curriculum learning, we propose the use of task difficulty as an effective fidelity criterion, which enables relative performance of different hyperparameters to be accurately estimated even with low-fidelity cheap approximations. We enable efficient search space pruning through a progressive acquisition function focusing only on optimal regions. Transfer learning is further adopted to reduce evaluation cost of function queries. Through extensive experiments, we demonstrate that CMFBO is more efficient than state-of-the-art hyperparameter optimization algorithms. In particular, we show that hyperparameters optimized through CMFBO result in at least 5x performance gain comparing to original author-released settings in DeepMimic. We believe that our method could serve as stepping-stone for automatic tuning of physics-based character animation systems and free researchers or engineers from laborious and exhausting tuning works.

Our current method has some limitations that we would like to investigate in the near future. First, the search space grows exponentially with the dimension of hyperparameters. Bayesian Optimization usually copes with hyperparameters with less than 30 dimensions, thus is not suitable for problems with high-dimensional hyperparameters. To overcome this limitation, we could incorporate modern machine learning and optimization techniques, such as random embedding [232], to scale our method to even higher dimensions requiring much larger computation budgets, such as optimizing the complex routing of muscles for full-body characters.

Second, in this work we mainly focus on optimizing stationary hyperparameters. However, the optimal hyperparameters of deep reinforcement learning systems could be non-stationary and change during the learning process. Recently some population-based hyper-

parameter optimization methods [79, 158, 46] have been proposed to discover non-stationary good hyperparameters for some deep reinforcement learning task. These methods maintain a population of agents trained with different hyperparameters in parallel, and periodically replace hyperparameter settings with inferior performances with promising ones. In the replacing process, the hyperparameters can be changed to explore the non-stationary property of hyperparameters. This enables online adaption of hyperparameters and has demonstrated effectiveness in finding non-stationary hyperparameters settings that lead to better model performance. We could adapt our CMFBO method to population-based settings to enable online adaption of hyperparameters.

Third, our method is limited to Euclidean search space. The search space in some tasks are non-Euclidean, such as optimizing orientation, stiffness matrices in robotics tasks. It would be interesting to extend our method to general Non-Euclidean search space such as Riemannian manifolds in [82].

Last but not least, our method is currently evaluated on simulation platforms. In principle it could also be applied to real world automatic robotics design and learning tasks. We might combine our method with 'sim to real' transfer learning [211, 261] to tackle real-world robotics tasks.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we make several contributions towards learning diverse and stylized motion skills for physics-based characters. More specially we develop some computational methods to help simulated characters learn to locomote, jump and manipulate tools in different styles without using task-specific motion examples.

In Chapter 3, we propose a deep reinforcement learning framework named Spacetime bounds to constrain the allowed state space of physics-based characters. We found that many motor skills can emerge naturally by specifying a set of loose spacetime constraints without using any handcrafted reward functions. As a result, simulated characters can freely explore the skill space within our specified spacetime bounds. For some challenging motor tasks which are difficult to learn by tracking-based methods, we can use a tight spacetime bound to replicate these challenging skills faithfully. Spacetime bounds can be combined with style rewards to support stylized skill exploration and enrich the generated motion variety.

In Chapter 4 we investigate motion skill discovery for challenging tasks such as high jumps in the absence of task-specific motion examples. Our key insight is that the take-off state of physics-based character before jumping is a strong determinant of the resulting jumping skill learned through deep reinforcement learning. We therefore propose a data-efficient diversity optimization algorithm, Bayesian diversity search, to explore the take-off state space to discover as many stylized jumping skills as possible. We also propose to use an autoencoder trained on motion datasets to regularize the action space of simulated characters, leading to learning more natural and visually pleasing motion solutions.

In Chapter 5 we further study diverse motion skill discovery for hand manipulation tasks. We choose one common but challenging tool: chopsticks to demonstrate our method. Similar to high jump tasks, we observe that the chopsticks holding pose play an important role in learning tool manipulation skills, so we adopt Bayesian Optimization to optimize holding poses for finding natural and functional gripping poses with which to hold chopsticks. Given

the optimized holding pose, we design a two-level control system to solve object relocation tasks using chopsticks in a physics-based manner. A high-level motion planner first synthesizes a kinematic motion plan for the hand,tools and objects that satisfies task requirements. Then a low-level hand controller is trained using deep reinforcement learning to reproduce the planned trajectories in physics simulations. Our system can learn to use chopsticks in diverse styles, at different holding positions, and for multiple hand morphologies.

Finally in Chapter 6 we recap the problems we solved in the previous two chapters, and formulate them as a general hyperparameter optimization problem for deep reinforcement learning. Most of the research efforts of hyperparameter optimization are focused on supervised learning instead of reinforcement learning, and tuning hyperparameters for DRL-based character control systems is non-intuitive and time-consuming. To solve this problem, we systematically explore general hyperparameter optimization for DRL-based character animation systems. We propose a novel algorithm: Curriculum-based Multi-Fidelity Bayesian Optimization to help search for suitable hyperparameters for DRL-based character control systems in an efficient manner. Instead of evaluating the candidate hyperparameter directly on the target task, we use the hyperparameter's performance on easier motor learning tasks to prune bad-performing hyperparameters and allocate more computational resources to more promising hyperparameters. Additionally policies learned at easier tasks can be transferred to the original difficult task for more efficient evaluations. We compare our method with several state-of-the-art hyperparameter optimization methods and show our method outperforms them. We believe our method can serve as a powerful tool for researchers and engineers to ease the development of DRL-based character control systems.

7.2 Some Future Work

Our work presented in this thesis takes steps towards stylized motor skill acquisition for physics-based characters without task-specific motion examples. However, there remain plenty of challenges to be solved before realizing physics-based agents that can behave and move like humans and other animals. Here we summarize some interesting and important research directions worth exploring.

7.2.1 Deep Reinforcement Learning Efficiency

Deep reinforcement learning provides a powerful tool for learning complex motor skills for high-dimensional characters. However, the policy learning process is still very computationally expensive and inefficient. Some skills, such as regular locomotions, can be learned within hours using high-end computers. While learning some challenging skills like gymnastic flips or learning a large corpus of motion skills recorded in datasets can take several days [168] or even weeks [126]. In practice, we usually train control policies repetitively to adjust the performance of learned skills. The high computation cost of policy learning hinders the fast

development of physics-based motion control systems, and even makes policy learning with large datasets infeasible for researchers who do not have access to enough computational resources. In contrast to these artificial agents whose motor abilities grow slowly, humans and other animals can perform unseen motions in a few-shot or zero-shot manner. We need to solve the computation cost bottleneck to develop physics-based characters that can replicate motion behaviors like humans. A possible solution is to inject specific domain knowledge into the policy learning process. Currently most motor skill learning tasks are solved by using model-free DRL algorithms like PPO, which views the system dynamics as a black-box function. Integrating expert knowledge of human movements with deep reinforcement learning might help accelerate the learning process. For example, some traditional control methods can produce robust locomotion controllers fast with simplified dynamic models like an inverted pendulum, and the parameter tuning process does not need to take millions of simulation samples. However, it could take a much longer time to learn such handcrafted locomotion behaviors using DRL. Recently [244] used a centroidal dynamic model to approximate the dynamics of a quadruped robot and demonstrated effective and efficient DRL training by using this centroidal model. [181] used a simplified dynamic model to capture the characteristics of brachiation movements and learned a control policy for this simplified model. The learned policy for the simplified model facilitates policy learning for the full character model. It is interesting to explore data-driven simplified dynamic models using deep neural networks. Data-driven dynamic models are more expressive, flexible and could describe complex dynamics involved in various motion tasks. If this data-driven dynamic model is learnable and robust, we can use it to facilitate policy learning and take a big and important step towards efficient policy learning.

7.2.2 Generalized Motor Skills

Although deep reinforcement learning is effective at solving complex motor control tasks, the generalization ability of DRL is limited. Control policies learned using deep reinforcement learning could overfit the environment settings and cannot generalize to unseen configurations well. For example, if we train a simulated character using slow walking motions, it is unlikely that the learned control policies can be extended to running motions in a few-shot manner. Control policies trained for one specific character morphology cannot be retargeted to another different morphology directly. Instead of learning a narrow set of skills that are specialized for certain tasks, we hope artificial agents can extract meta motor abilities from training tasks and reuse them to facilitate new skill learning and finish unseen tasks. There are several promising research directions that might help us tackle this challenging problem. The first direction is to develop explainable motor skill learning algorithms. Most DRL methods are model-free and learn skills end to end, which makes it difficult for us to understand what the policy actually learns. As we mentioned in the last subsection, combining dynamic models requiring domain knowledge and DRL might be

a solution to learning generalized motor abilities. Another viable direction to improve the generalization ability of DRL for policy learning is to use large pre-trained models. Deep learning models in computer vision and natural language processing tasks have benefited a lot from the off-the-shelf pretrained models. These large pretrained models can provide good initial solutions to problems at hands, accelerate the training process and improve the final outputs. For example, many deep learning-based computer vision models use ResNet [60] pretrained at large image datasets as backbones. Recently generative models have seen remarkable breakthroughs with the adoption of large-pretrained models, such as diffusion models [36, 182] and ChatGPT [151] which can synthesize highly realistic images and texts, to the point where the generated content is often indistinguishable from content created by humans. These advancements have opened up new avenues for research in generative modeling and hold great promises for future applications. However, a common practice in physics-based character control is training policies from scratch. A limited number of works explore policy reuse in the form of hierarchical deep reinforcement learning. For example, [166, 168, 136, 126] used reference motions to construct low-level motion primitives and then learned high-level schedulers to schedule these motor primitives for downstream tasks. These methods successfully demonstrated more complex behaviours such as catching and moving boxes and soccer team play. However, the variety of motor primitives is limited and the generated motions contain some artifacts. Developing large pretrained motor models compatible with downstream tasks could be an exciting direction for future character control research.

7.2.3 Learning From Videos and Languages

Most imitation-based motor skill learning frameworks use motion capture data as a source of imitation objectives. High-quality motion capture data is still an indispensable component of current animation systems for synthesizing general and realistic character movements. However, unlike images or texts that can be collected easily, acquiring high-quality motion capture data requires carefully instrumented environments, professional actors and enormous human efforts for post-processing. It is infeasible to construct a huge human motion dataset whose magnitude of variations is comparable to that of widely used image datasets such as the ImageNet [35]. In contrast, videos provide a natural and realistic representation of human motion and are easier and cheaper to obtain, making them a perfect choice of imitation source data. Recent progress in vision-based pose estimators has made inferring 3D human motions from 2D videos possible. The 3D human motions extracted from videos by using these pose estimators can serve as noisy references for physics-based agents to imitate. [169] adopted this idea and has successfully demonstrated acquiring skills from videos. While the quality of generated skills strongly relies on the estimated 3D motions. Inaccurate 3D motions could lead to failed learning of the corresponding target skills. It is interesting to explore end-to-end skill learning from videos, which might reduce the dependence on

accurate pose estimators. For example, we could combine our Spacetime bounds, GAIL and some consistency projection reward functions to improve the visual quality of learned skills.

In addition to videos and motion capture, language can also be used to implicitly depict human motion. Prior works, such as those presented in [231, 205], have demonstrated systems capable of directing character motions using spoken inputs for kinematic animation. Texts can also serve as supervision signals to guide the motor skill learning process of physics-based agents. Traditional methods such as those described in [55] have used pre-defined finite sets of languages as instructions to improve learned skills in physics-based agents. Recently, there have been works focusing on constructing general mappings between texts and human motions. MotionCLIP [213] is an auto-encoder model that aligns the latent space of human motions to the language latent space. Similarly, [86] proposed an approach to map high-level language commands to low-level controls, enabling physics-based agents to perform motions specified by users in texts. Despite the promising results, current language motion models are still limited in expressing a diverse range of motions with detail. An interesting avenue for future research is to leverage ChatGPT, a powerful language model, to construct a powerful and robust language motion model to facilitate motor skill learning from languages. For example, we could use MotionCLIP to produce texts describing motor skills performed by physics-based agents, and then use ChatGPT as a virtual coach to generate instructions on how to improve the learned skill. This iterative and automatic refinement process could potentially enhance the motion quality of the learned skills.

In conclusion, this line of work can help us build a large and general motor skill model capable of reproducing a massive set of skills, which can be used as a powerful pretrained model for many downstream tasks. This pretrained model that embodies the rich human motor behaviors and experiences might serve as a key component for achieving general stylized motion skill discovery and acquisition.

Bibliography

[1] CMU motion capture database. `http://mocap.cs.cmu.edu`.

[2] SFU motion capture database. `http://mocap.cs.sfu.ca`.

[3] Kfir Aberman, Yijia Weng, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Unpaired motion style transfer from video to animation. *ACM Transactions on Graphics (TOG)*, 39(4):64, 2020.

[4] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[5] V.M. Adashevskiy, S.S. Iermakov, and A.A. Marchenko. Biomechanics aspects of technique of high jump. *Physical Education of Students*, 17(2):11–17, 2013.

[6] Shailen Agrawal, Shuo Shen, and Michiel van de Panne. Diverse motion variations for physics-based character animation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 37–44, 2013.

[7] Shailen Agrawal, Shuo Shen, and Michiel van de Panne. Diverse motions and character shapes for simulated skills. *IEEE transactions on visualization and computer graphics*, 20(10):1345–1355, 2014.

[8] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[9] Mazen Al Borno, Martin De Lasa, and Aaron Hertzmann. Trajectory optimization for full-body movements with complex contacts. *IEEE transactions on visualization and computer graphics*, 19(8):1405–1414, 2012.

[10] Sheldon Andrews and Paul G Kry. Goal directed multi-finger manipulation: Control policies and analysis. *Computers & Graphics*, 37(7):830–839, 2013.

[11] Andreas Aristidou, Qiong Zeng, Efstathios Stavrakis, KangKang Yin, Daniel Cohen-Or, Yiorgos Chrysanthou, and Baoquan Chen. Emotion control of unstructured dance movements. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 1–10, 2017.

[12] Yahya Aydin and Masayuki Nakajima. Database guided computer animation of human grasping using forward and inverse kinematics. *Computers & Graphics*, 23(1):145–154, 1999.

[13] Javad Azimi, Alan Fern, and Xiaoli Z Fern. Batch bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems*, pages 109–117. Citeseer, 2010.

[14] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM New York, NY, USA, 2009.

[15] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics (TOG)*, 38(6):206, 2019.

[16] Josh Bongard. Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, 108(4):1234–1239, 2011.

[17] Eric Brochu, Tyson Brochu, and Nando de Freitas. A bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 103–112, 2010.

[18] Eric Brochu, Abhijeet Ghosh, and Nando de Freitas. Preference galleries for material design. *SIGGRAPH Posters*, 105(10.1145):1280720–1280834, 2007.

[19] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[20] Jinxiang Chai and Jessica K Hodgins. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics (TOG)*, 24(3):686–696, 2005.

[21] Bao-Chi Chang, Biing-Shiun Huang, Ching-Kong Chen, and Shyh-Jen Wang. The pincer chopsticks: The investigation of a new utensil in pinching function. *Applied ergonomics*, 38(3):385–390, 2007.

[22] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[23] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*. PMLR, 2021.

[24] Matei Ciocarlie. *Low-Dimensional Robotic Grasping: Eigengrasp Subspaces and Optimized Underactuation*. PhD thesis, Columbia University, 2010.

[25] Alexandra Coman and Hector Munoz-Avila. Generating diverse plans using quantitative and qualitative plan distance metrics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2011.

[26] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in neural information processing systems*, pages 5027–5038, 2018.

[27] Stelian Coros, Philippe Beaudoin, and Michiel Van de Panne. Generalized biped walking control. *ACM Transactions on Graphics (TOG)*, 29(4):130, 2010.

[28] Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel Van De Panne. Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics (TOG)*, 30(4):59, 2011.

[29] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[30] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[31] Ana Lucia Cruz Ruiz, Charles Pontonnier, Jonathan Levy, and Georges Dumont. A synergy-based control solution for overactuated characters: Application to throwing. *Computer Animation and Virtual Worlds*, 28(6):e1743, 2017.

[32] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *The 16th International Conference on Artificial Intelligence and Statistics*, pages 207–215. PMLR, 2013.

[33] Jesus Dapena. The evolution of high jumping technique: Biomechanical analysis. In *20th Internat. Symp. Biomech. Sports*, 2002.

[34] Martin De Lasa, Igor Mordatch, and Aaron Hertzmann. Feature-based locomotion controllers. *ACM Transactions on Graphics (TOG)*, 29(4):131, 2010.

[35] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[36] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[37] Sean Donnelly. *An Introduction to the High Jump*. 2014.

[38] Han Du, Erik Herrmann, Janis Sprenger, Klaus Fischer, and Philipp Slusallek. Stylistic locomotion modeling and synthesis using variational generative models. In *Motion, Interaction and Games*, pages 1–10, 2019.

[39] George ElKoura and Karan Singh. Handrix: animating the human hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 110–119, 2003.

[40] Tom Erez, Kendall Lowrey, Yuval Tassa, Vikash Kumar, Svetoslav Kolev, and Emanuel Todorov. An integrated system for real-time model predictive control of humanoid robots. In *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*, pages 292–299. IEEE, 2013.

[41] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2019.

[42] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 11444–11453, 2020.

[43] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39(2-3):202–216, 2020.

[44] Martin L Felis and Katja Mombaur. Synthesis of full-body 3D human gait using optimal control methods. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1560–1566. IEEE, 2016.

[45] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE international conference on computer vision*, pages 4346–4354, 2015.

[46] Jörg KH Franke, Gregor Köhler, André Biedenkapp, and Frank Hutter. Sample-efficient automated deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

[47] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[48] Levi Fussell, Kevin Bergamin, and Daniel Holden. Supertrack: Motion tracking for physically simulated characters using supervised learning. *ACM Transactions on Graphics (TOG)*, 40(6):197, 2021.

[49] Guillermo Garcia-Hernando, Edward Johns, and Tae-Kyun Kim. Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9561–9568. IEEE, 2020.

[50] Thomas Geijtenbeek, Michiel Van De Panne, and A Frank Van Der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)*, 32(6):206, 2013.

[51] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[52] F Sebastian Grassia. Practical parameterization of rotations using the exponential map. *Journal of graphics tools*, 3(3):29–48, 1998.

[53] David Ha. Reinforcement learning for improving agent design. *Artificial life*, 25(4):352–365, 2019.

[54] Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. Joint optimization of robot design and motion parameters using the implicit function theorem. In *Robotics: Science and systems*, 2017.

[55] Sehoon Ha and C Karen Liu. Iterative training of dynamic skills inspired by human coaching techniques. *ACM Transactions on Graphics (TOG)*, 34(1):1, 2014.

[56] Sehoon Ha, Yuting Ye, and C Karen Liu. Falling and landing motion control for character animation. *ACM Transactions on Graphics (TOG)*, 31(6):155, 2012.

[57] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[58] Ikhsanul Habibie, Daniel Holden, Jonathan Schwarz, Joe Yearsley, and Taku Komura. A recurrent variational autoencoder for human motion synthesis. In *28th British Machine Vision Conference*, 2017.

[59] Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[61] Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 5, pages 372–377, 2005.

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

[63] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *The Journal of Machine Learning Research*, 13(1):1809–1837, 2012.

[64] Lars Hertel, Pierre Baldi, and Daniel L Gillen. Quantity vs. quality: On hyperparameter optimization for deep reinforcement learning. *arXiv preprint arXiv:2007.14604*, 2020.

[65] Todd Hester and Peter Stone. Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence*, 247:170–186, 2017.

[66] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017.

[67] Jessica K Hodgins, Wayne L Wooten, David C Brogan, and James F O'Brien. Animating human athletics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 71–78, 1995.

[68] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transctions on Graphics (TOG)*, 36(4):42, 2017.

[69] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Transctions on Graphics (TOG)*, 35(4):138, 2016.

[70] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, pages 1–4. 2015.

[71] Seokpyo Hong, Daseong Han, Kyungmin Cho, Joseph S Shin, and Junyong Noh. Physics-based full-body soccer motion control for dribbling and shooting. *ACM Transactions on Graphics (TOG)*, 38(4):74, 2019.

[72] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29:1109–1117, 2016.

[73] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. *ACM Transactions on Graphics (TOG)*, 24(3):1082–1089, 2005.

[74] Haoyuan Hu, Xiaodong Zhang, Xiaowei Yan, Longfei Wang, and Yinghui Xu. Solving a new 3D bin packing problem with deep reinforcement learning method. *arXiv preprint arXiv:1708.05930*, 2017.

[75] Sha Hu, Zeshi Yang, and Greg Mori. Neural fidelity warping for efficient robot morphology design. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7079–7086. IEEE, 2021.

[76] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

[77] Zhiyong Huang, Ronan Boulic, Nadia Magnenat Thalmann, and Daniel Thalmann. A multi-sensor approach for grasping and 3D interaction. In *Computer graphics*, pages 235–253. Elsevier, 1995.

[78] Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. Policies modulating trajectory generators. In *Conference on Robot Learning*, 2018.

[79] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[80] Sumit Jain and C Karen Liu. Controlling physics-based characters using soft contacts. *ACM Transactions on Graphics (TOG)*, 30(6):1–10, 2011.

[81] Sumit Jain, Yuting Ye, and C Karen Liu. Optimization-based interactive motion synthesis. *ACM Transactions on Graphics (TOG)*, 28(1):10, 2009.

[82] Noémie Jaquier, Leonel Rozo, Sylvain Calinon, and Mathias Bürger. Bayesian optimization meets riemannian manifolds in robot learning. In *Conference on Robot Learning*, pages 233–246. PMLR, 2020.

[83] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[84] Rohan A Joseph, Alvin C Goh, Sebastian P Cuevas, Michael A Donovan, Matthew G Kauffman, Nilson A Salas, Brian Miles, Barbara L Bass, and Brian J Dunkin.

"chopstick" surgery: A novel technique improves surgeon performance and eliminates arm collision in robotic single-incision laparoscopic surgery. *Surgical endoscopy*, 24(6):1331–1335, 2010.

[85] Teen Jumper. 7 classic high jump styles, 2020.

[86] Jordan Juravsky, Yunrong Guo, Sanja Fidler, and Xue Bin Peng. Padl: Language-directed physics-based character control. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[87] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems*, pages 992–1000, 2016.

[88] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity bayesian optimisation with continuous approximations. In *Advances in Neural Information Processing Systems*, pages 1799–1808, 2017.

[89] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in neural information processing systems*, pages 2016–2025, 2018.

[90] Andrej Karpathy and Michiel Van De Panne. Curriculum learning for motor skills. In *Canadian Conference on Artificial Intelligence*, pages 325–330. Springer, 2012.

[91] Korrawe Karunratanakul, Jinlong Yang, Yan Zhang, Michael J Black, Krikamol Muandet, and Siyu Tang. Grasping field: Learning implicit representations for human grasps. In *2020 International Conference on 3D Vision (3DV)*, pages 333–344. IEEE, 2020.

[92] Liyiming Ke, Ajinkya Kamat, Jingqiang Wang, Tapomayukh Bhattacharjee, Christoforos Mavrogiannis, and Siddhartha S Srinivasa. Telemanipulation with chopsticks: Analyzing human factors in user demonstrations. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11539–11546. IEEE, 2020.

[93] Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6185–6191. IEEE, 2021.

[94] Junhwan Kim, Frederic Cordier, and Nadia Magnenat-Thalmann. Neural network-based violinist's hand animation. In *Proceedings Computer Graphics International 2000*, pages 37–41. IEEE, 2000.

[95] Seungsu Kim, Ashwini Shukla, and Aude Billard. Catching objects in flight. *IEEE Transactions on Robotics*, 30(5):1049–1065, 2014.

[96] Uikyum Kim, Dawoon Jung, Heeyoen Jeong, Jongwoo Park, Hyun-Mok Jung, Joono Cheong, Hyouk Ryeol Choi, Hyunmin Do, and Chanhun Park. Integrated linkage-driven dexterous anthropomorphic robotic hand. *Nature Communications*, 12(1):1–13, 2021.

[97] Yeonju Kim, Zherong Pan, and Kris Hauser. Mo-bbo: Multi-objective bilevel bayesian optimization for robot and behavior co-design. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9877–9883. IEEE, 2021.

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

[99] Mariam Kiran and Melis Ozyildirim. Hyperparameter tuning for deep reinforcement learning applications. *arXiv preprint arXiv:2201.11182*, 2022.

[100] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *The 20th International Conference on Artificial Intelligence and Statistics*, pages 528–536. PMLR, 2017.

[101] Yoshihito Koga, Koichi Kondo, James Kuffner, and Jean-Claude Latombe. Planning motions with intentions. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 395–408, 1994.

[102] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

[103] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transctions on Graphics (TOG)*, 21(3):473–482, 2002.

[104] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. Sequential line search for efficient visual design optimization by crowds. *ACM Transactions on Graphics (TOG)*, 36(4):48, 2017.

[105] Paul G Kry and Dinesh K Pai. Interaction capture and synthesis. *ACM Transactions on Graphics (TOG)*, 25(3):872–880, 2006.

[106] Kyungho Lee, Seyoung Lee, and Jehee Lee. Interactive character animation by learning multi-objective control. *ACM Transactions on Graphics (TOG)*, 37(6):180, 2018.

[107] Kyungho Lee, Sehee Min, Sunmin Lee, and Jehee Lee. Learning time-critical responses for interactive character control. *ACM Transactions on Graphics (TOG)*, 40(4):147, 2021.

[108] Seunghwan Lee, Phil Sik Chang, and Jehee Lee. Deep compliant control. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, page 23. Association for Computing Machinery, 2022.

[109] Seyoung Lee, Sunmin Lee, Yongwoo Lee, and Jehee Lee. Learning a family of motor skills from a single motion clip. *ACM Transactions on Graphics (TOG)*, 40(4):93, 2021.

[110] Yoonsang Lee, Moon Seok Park, Taesoo Kwon, and Jehee Lee. Locomotion control for many-muscle humanoids. *ACM Transactions on Graphics (TOG)*, 33(6):218, 2014.

[111] Joel Lehman and Kenneth O Stanley. Novelty search and the problem with objectives. In *Genetic programming theory and practice IX*, pages 37–56. Springer, 2011.

[112] Sergey Levine, Jack M Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (TOG)*, 31(4):28, 2012.

[113] Shutao Li, Mingkui Tan, Ivor W Tsang, and James Tin-Yau Kwok. A hybrid pso-bfgs strategy for global optimization of multimodal functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(4):1003–1014, 2011.

[114] Thomas Liao, Grant Wang, Brian Yang, Rene Lee, Kristofer Pister, Sergey Levine, and Roberto Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2488–2494. IEEE, 2019.

[115] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2016.

[116] Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)*, 39(4):40, 2020.

[117] Hod Lipson and Jordan B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978, 2000.

[118] C Karen Liu. Synthesis of interactive hand manipulation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 163–171, 2008.

[119] C Karen Liu. Dextrous manipulation from a grasping pose. *ACM Transactions on Graphics (TOG)*, 28(3):59, 2009.

[120] C Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics (TOG)*, 21(3):94, 2002.

[121] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[122] Libin Liu and Jessica Hodgins. Learning to schedule control fragments for physics-based characters using deep Q-learning. *ACM Transactions on Graphics (TOG)*, 36(3):42a, 2017.

[123] Libin Liu, Michiel Van De Panne, and KangKang Yin. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)*, 35(3):29, 2016.

[124] Libin Liu, KangKang Yin, and Baining Guo. Improving sampling-based motion control. *Computer Graphics Forum*, 34(2):415–423, 2015.

[125] Libin Liu, KangKang Yin, Michiel Van de Panne, Tianjia Shao, and Weiwei Xu. Sampling-based contact-rich motion control. *ACM Transactions on Graphics (TOG)*, 29(4):128, 2010.

[126] Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, SM Ali Eslami, Daniel Hennes, Wojciech M Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, et al. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69):eabo0235, 2022.

[127] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):248, 2015.

[128] Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Conference on Robot Learning*, pages 854–869. PMLR, 2020.

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

[130] Yunhao Luo, Kaixiang Xie, Sheldon Andrews, and Paul G Kry. Catching and throwing control of a physically simulated hand. In *Motion, Interaction and Games*, pages 1–7, 2021.

[131] Li-Ke Ma, Zeshi Yang, Baining Guo, and KangKang Yin. Towards robust direction invariance in character animation. *Computer Graphics Forum*, 38(7):235–242, 2019.

[132] Li-Ke Ma, Zeshi Yang, Xin Tong, Baining Guo, and KangKang Yin. Learning and exploring motor skills with spacetime bounds. *Computer Graphics Forum*, 40(2):251–263, 2021.

[133] Wanli Ma, Shihong Xia, Jessica K Hodgins, Xiao Yang, Chunpeng Li, and Zhaoqi Wang. Modeling style and variation in human motion. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 21–30, 2010.

[134] Macro. Ten thousand ways to use chopsticks. `https://marcosticks.org/poster-ten-thousand-ways-to-use-chopsticks/`, since 2021.

[135] B Matérn. Spatial variation: Meddelanden fran statens skogsforskningsinstitut. *Lecture Notes in Statistics*, 36:21, 1960.

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

[137] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM transactions on graphics (TOG)*, 30(4):87, 2011.

[138] Jianyuan Min and Jinxiang Chai. Motion graphs++ a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (TOG)*, 31(6):153, 2012.

[139] Lester James V. Miranda. PySwarms, a research-toolkit for particle swarm optimization in Python. *Journal of Open Source Software*, 3, 2018.

[140] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1928–1937, 2016.

[141] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[142] Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, and Emanuel V Todorov. Interactive control of diverse complex characters with neural networks. *Advances in neural information processing systems*, 28:3132–3140, 2015.

[143] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 137–144, 2012.

[144] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):43, 2012.

[145] Yukiko Mukai and Keiko Hashimoto. A study on ways of holding chopsticks. *Journal of Home Economics of Japan*, 29(7):467–473, 1978.

[146] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[147] Vu Nguyen and Michael A Osborne. Knowing the what but not the where in bayesian optimization. In *Proceedings of the 37th International Conference on Machine Learning*, pages 7317–7326. PMLR, 2020.

[148] Vu Nguyen, Sebastian Schulze, and Michael Osborne. Bayesian optimization for iterative learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[149] Rubens F Nunes, Joaquim B Cavalcante-Neto, Creto A Vidal, Paul G Kry, and Victor B Zordan. Using natural vibrations to guide control for locomotion. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 87–94, 2012.

[150] K. Okuyama, M. Ae, and T. Yokozawa. Three dimensional joint torque of the takeoff leg in the fosbury flop style. In *Proceedings of the XIXth Congress of the International Society of the Biomechanics (CD-ROM)*, 2003.

[151] OpenAI. ChatGPT: Optimizing language models for dialogue. `https://openai.com/blog/chatgpt/`, 2022.

[152] Takayuki Osa, Jan Peters, and Gerhard Neumann. Hierarchical reinforcement learning of multiple grasping strategies with human instructions. *Advanced Robotics*, 32(18):955–968, 2018.

[153] Tomoko Osera, Chihiro Yamamoto, Rika Senke, Misako Kobayashi, Setsuko Tsutie, and Nobutaka Kurihara. Relationship between mothers and children on how to hold chopsticks and concerns about chopsticks in japanese kindergarten. *Journal of Japanese Society of Shokuiku*, 12(1):19–25, 2018.

[154] SA Overduin, A d'Avella, J. Roh, JM Carmena, and E. Bizzi. Representation of muscle synergies in the primate brain. *Journal of Neuroscience*, 37, 2015.

[155] Jahng-Hyon Park and Haruhiko Asada. Concurrent design optimization of mechanical structure and control for high speed robots. 1994.

[156] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)*, 38(6):205, 2019.

[157] Soomin Park, Deok-Kyeong Jang, and Sung-Hee Lee. Diverse motion stylization for multiple style domains via spatial-temporal graph-based generative model. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(3):1–17, 2021.

[158] Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. Provably efficient online hyperparameter optimization with population-based bandits. *Advances in Neural Information Processing Systems*, 33:17200–17211, 2020.

[159] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[160] Chandana Paul and Josh C Bongard. The road less travelled: Morphology in the optimization of biped robot locomotion. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 226–232. IEEE, 2001.

[161] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):143, 2018.

[162] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[163] Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. Dynamic terrain traversal skills using reinforcement learning. *ACM Transactions on Graphics (TOG)*, 34(4), 2015.

[164] Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4):80, 2016.

[165] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):41, 2017.

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

[167] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems*, 2020.

[168] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 41(4):94, 2022.

[169] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions on Graphics (TOG)*, 37(6):178, 2018.

[170] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):144, 2021.

[171] Anton C Pil and H Haruhiko Asada. Integrated structure/control design of mechatronic systems using a recursive experimental optimization method. *IEEE/ASME transactions on mechatronics*, 1(3):191–203, 1996.

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

[173] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[174] Amir H Rabbani, Michiel van de Panne, and Paul G Kry. Anticipatory balance control and dimension reduction. *Computer Animation and Virtual Worlds*, 29(6):e1726, 2018.

[175] Madhu Ragupathi, Diego I Ramos-Valadez, Rodrigo Pedraza, and Eric M Haas. Robotic-assisted single-incision laparoscopic partial cecectomy. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 6(3):362–367, 2010.

[176] Akshara Rai, Rika Antonova, Seungmoon Song, William Martin, Hartmut Geyer, and Christopher Atkeson. Bayesian optimization using domain knowledge on the atrias biped. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778. IEEE, 2018.

[177] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Robotics: Science and Systems*, 2018.

[178] Ahmed A Ramadan, Tomohito Takubo, Yasushi Mae, Kenichi Oohara, and Tatsuo Arai. Developmental process of a chopstick-like hybrid-structure two-fingered micromanipulator hand for 3-d manipulation of microscopic objects. *IEEE Transactions on Industrial Electronics*, 56(4):1121–1135, 2009.

[179] Avinash Ranganath, Pei Xu, Ioannis Karamouzas, and Victor Zordan. Low dimensional motor skill learning using coactivation. In *Motion, Interaction and Games*, pages 1–10, 2019.

[180] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[181] Daniele Reda, Hung Yu Ling, and Michiel van de Panne. Learning to brachiate via simplified model imitation. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, page 24. Association for Computing Machinery, 2022.

[182] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[183] Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Transctions on Graphics (TOG)*, 26(3):106–es, 2007.

[184] Alla Safonova, Jessica K Hodgins, and Nancy S Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transctions on Graphics (TOG)*, 23(3):514–521, 2004.

[185] Haruka Sakurai, Takahiro Kanno, and Kenji Kawashima. Thin-diameter chopsticks robot for laparoscopic surgery. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4122–4127. IEEE, 2016.

[186] Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9798–9805. IEEE, 2019.

[187] Jürgen Schmidhuber. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pages 1458–1463, 1991.

[188] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

[189] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

[191] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[192] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[193] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, 1994.

[194] Harrison Jesse Smith, Chen Cao, Michael Neff, and Yingying Wang. Efficient neural networks for real-time motion style transfer. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2(2):1–17, 2019.

[195] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[196] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pages 2171–2180, 2015.

[197] Jialin Song, Yuxin Chen, and Yisong Yue. A general framework for multi-fidelity bayesian optimization with gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3158–3167, 2019.

[198] Andrew Spielberg, Brandon Araki, Cynthia Sung, Russ Tedrake, and Daniela Rus. Functional co-optimization of articulated robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5042. IEEE, 2017.

[199] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, page 1015–1022, 2010.

[200] Biplav Srivastava, Tuan Anh Nguyen, Alfonso Gerevini, Subbarao Kambhampati, Minh Binh Do, and Ivan Serina. Domain independent approaches for finding diverse plans. In *IJCAI*, pages 2016–2022, 2007.

[201] Sebastian Starke, Ian Mason, and Taku Komura. Deepphase: periodic autoencoders for learning motion phase manifolds. *ACM Transactions on Graphics (TOG)*, 41(4):136, 2022.

[202] Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. Neural state machine for character-scene interactions. *ACM Transactions on Graphics (TOG)*, 38(6):209, 2019.

[203] Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)*, 39(4):54, 2020.

[204] Sebastian Starke, Yiwei Zhao, Fabio Zinno, and Taku Komura. Neural animation layering for synthesizing martial arts movements. *ACM Transactions on Graphics (TOG)*, 40(4):92, 2021.

[205] Hariharan Subramonyam, Wilmot Li, Eytan Adar, and Mira Dontcheva. Taketoons: Script-driven performance animation. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pages 663–674, 2018.

[206] Hao Sun, Zhenghao Peng, Bo Dai, Jian Guo, Dahua Lin, and Bolei Zhou. Novel policy seeking with constrained optimization, 2020.

[207] Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. 1998.

[208] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

[209] Omid Taheri, Nima Ghorbani, Michael J Black, and Dimitrios Tzionas. GRAB: A dataset of whole-body human grasping of objects. In *European conference on computer vision (ECCV)*, pages 581–600. Springer, 2020.

[210] Jie Tan, Karen Liu, and Greg Turk. Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications*, 31(4):34–44, 2011.

[211] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems*, 2018.

[212] Tianxin Tao, Matthew Wilson, Ruiyu Gou, and Michiel van de Panne. Learning to get up. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, page 47, New York, NY, USA, 2022. Association for Computing Machinery.

[213] Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermano, and Daniel Cohen-Or. Motionclip: Exposing human motion generation to clip space. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 358–374. Springer, 2022.

[214] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[215] Emanuel Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6054–6061. IEEE, 2014.

[216] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033. IEEE, 2012.

[217] Deepak Tolani and Norman I. Badler. Real-time inverse kinematics of the human arm. *Presence: Teleoperators and Virtual Environments*, 5(4):393–401, 1996.

[218] Marc A Toussaint, Kelsey Rebecca Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. 2018.

[219] Ethan Tseng, Felix Yu, Yuting Yang, Fahim Mannan, Karl ST Arnaud, Derek Nowrouzezahrai, Jean-François Lalonde, and Felix Heide. Hyperparameter optimization in black-box image processing using differentiable proxies. *ACM Transactions on Graphics (TOG)*, 38(4):27, 2019.

[220] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 91–96, 1995.

[221] Rasmus K Ursem. Diversity-guided evolutionary algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 462–471. Springer, 2002.

[222] Michiel Van de Panne and Alexis Lamouret. Guided optimization for balanced locomotion. In *Computer Animation and Simulation'95*, pages 165–177. Springer, 1995.

[223] Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, and James Hensman. A framework for interdomain and multioutput Gaussian processes. *arXiv:2003.01115*, 2020.

[224] Miguel G Villarreal-Cervantes, Carlos A Cruz-Villar, Jaime Alvarez-Gallegos, and Edgar A Portilla-Flores. Robust structure-control design approach for mechatronic systems. *IEEE/ASME Transactions on Mechatronics*, 18(5):1592–1601, 2012.

[225] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[226] Jack M Wang, David J Fleet, and Aaron Hertzmann. Optimizing walking controllers. *ACM Transactions on Graphics (TOG)*, 28(5):1–8, 2009.

[227] Jack M Wang, Samuel R Hamner, Scott L Delp, and Vladlen Koltun. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG)*, 31(4):25, 2012.

[228] Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. Unicon: Universal neural controller for physics-based character motion. *arXiv preprint arXiv:2011.15119*, 2020.

[229] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[230] Yangang Wang, Jianyuan Min, Jianjie Zhang, Yebin Liu, Feng Xu, Qionghai Dai, and Jinxiang Chai. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics (TOG)*, 32(4):43, 2013.

[231] Zhijin Wang and Michiel van de Panne. "walk to here": A voice driven animation system. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 243–251, 2006.

[232] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, Nando De Freitas, et al. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, pages 1778–1784, 2013.

[233] Yu-Hui Wen, Zhipeng Yang, Hongbo Fu, Lin Gao, Yanan Sun, and Yong-Jin Liu. Autoregressive stylized motion synthesis with generative flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13612–13621, 2021.

[234] Nkenge Wheatland, Yingying Wang, Huaguang Song, Michael Neff, Victor Zordan, and Sophie Jörg. State of the art in hand and finger modeling and animation. *Computer Graphics Forum*, 34(2):735–760, 2015.

[235] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

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

[237] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Transactions on Graphics (TOG)*, 40(4):146, 2021.

[238] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Physics-based character controllers using conditional vaes. *ACM Transactions on Graphics (TOG)*, 41(4):96, 2022.

[239] Jungdam Won and Jehee Lee. Learning body shape variation in physics-based characters. *ACM Transactions on Graphics (TOG)*, 38(6):207, 2019.

[240] Wayne Lewis Wooten. *Simulation of leaping, tumbling, landing, and balancing humans*. Georgia Institute of Technology, 1998.

[241] Jia-chi Wu and Zoran Popović. Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics (TOG)*, 29(4):72, 2010.

[242] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019.

[243] Shihong Xia, Congyi Wang, Jinxiang Chai, and Jessica Hodgins. Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics (TOG)*, 34(4):119, 2015.

[244] Zhaoming Xie, Xingye Da, Buck Babich, Animesh Garg, and Michiel Van de Panne. Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model. In *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2022.

[245] Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. Allsteps: curriculum-driven learning of stepping stone skills. *Computer Graphics Forum*, 39(8):213–224, 2020.

[246] Zhaoming Xie, Sebastian Starke, Hung Yu Ling, and Michiel van de Panne. Learning soccer juggling skills with layer-wise mixture-of-experts. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, page 25. Association for Computing Machinery, 2022.

[247] Jie Xu, Andrew Spielberg, Allan Zhao, Daniela Rus, and Wojciech Matusik. Multi-objective graph heuristic search for terrestrial robot design. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9863–9869. IEEE, 2021.

[248] Kazuki Yamakawa, Yuko Tashiro, Mizuki Nakajima, and Tsuyoshi Saitoh. Development of a support system for holding chopsticks correctly. In *2018 International Workshop on Advanced Image Technology (IWAIT)*, pages 1–2. IEEE, 2018.

[249] Tomoko Yamauchi, Atsumi Koide, Atsuko Yamamoto, and Kazuko Oba. Effect of parental training on table manners and the way of holding chopsticks. *Journal of Cookery Science of Japan*, 43(4):260–264, 2010.

[250] Akira Yamazaki and Ryosuke Masuda. Autonomous foods handling by chopsticks for meal assistant robot. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6. VDE, 2012.

[251] Zeshi Yang, Kangkang Yin, and Libin Liu. Learning to use chopsticks in diverse gripping styles. *ACM Transactions on Graphics (TOG)*, 41(4):95, 2022.

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

[253] Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. Controlvae: Model-based learning of generative controllers for physics-based characters. *arXiv preprint arXiv:2210.06063*, 2022.

[254] Yuting Ye and C Karen Liu. Synthesis of responsive motion using a dynamic model. *Computer Graphics Forum*, 29(2):555–562, 2010.

[255] Yuting Ye and C Karen Liu. Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (TOG)*, 31(4):41, 2012.

[256] KangKang Yin, Stelian Coros, Philippe Beaudoin, and Michiel Van de Panne. Continuation methods for adapting simulated skills. *ACM Transactions on Graphics (TOG)*, 27(3):1–7, 2008.

[257] KangKang Yin, Kevin Loken, and Michiel Van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)*, 26(3):105–es, 2007.

[258] Zhiqi Yin, Zeshi Yang, Michel Van de Panne, and Kangkang Yin. Discovering diverse athletic jumping strategies. *ACM Transactions on Graphics (TOG)*, 40(4):91, 2021.

[259] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

[260] Ri Yu, Hwangpil Park, and Jehee Lee. Human dynamics from monocular video with dynamic camera movements. *ACM Transactions on Graphics (TOG)*, 40(6):208, 2021.

[261] Wenhao Yu, Visak CV Kumar, Greg Turk, and C Karen Liu. Sim-to-real transfer for biped locomotion. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3503–3510. IEEE, 2019.

[262] Wenhao Yu, Greg Turk, and C Karen Liu. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)*, 37(4):144, 2018.

[263] M Ersin Yumer and Niloy J Mitra. Spectral style transfer for human motion between independent actions. *ACM Transactions on Graphics (TOG)*, 35(4):137, 2016.

[264] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[265] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8660–8669, 2019.

[266] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transctions on Graphics (TOG)*, 37(4):145, 2018.

[267] He Zhang, Yuting Ye, Takaaki Shiratori, and Taku Komura. Manipnet: Neural manipulation synthesis with a hand-object spatial representation. *ACM Transactions on Graphics (TOG)*, 40(4):121, 2021.

[268] Yunbo Zhang, Wenhao Yu, C Karen Liu, Charlie Kemp, and Greg Turk. Learning to manipulate amorphous materials. *ACM Transactions on Graphics (TOG)*, 39(6):189, 2020.

[269] Yunbo Zhang, Wenhao Yu, and Greg Turk. Learning novel policies for tasks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7483–7492. PMLR, 2019.

[270] Zhenyue Zhang and Jing Wang. MLLE: Modified locally linear embedding using multiple weights. In *Advances in neural information processing systems*, pages 1593–1600, 2007.

[271] Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):188, 2020.

[272] K. Zhao, Z. Zhang, H. Wen, Z. Wang, and J. Wu. Modular organization of muscle synergies to achieve movement behaviors. *Journal of Healthcare Engineering*, 2019.

[273] Peng Zhao and Michiel van de Panne. User interfaces for interactive control of physics-based 3D characters. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 87–94, 2005.

[274] Wenping Zhao, Jianjie Zhang, Jianyuan Min, and Jinxiang Chai. Robust realtime physics-based motion control for human grasping. *ACM Transactions on Graphics (TOG)*, 32(6):207, 2013.

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