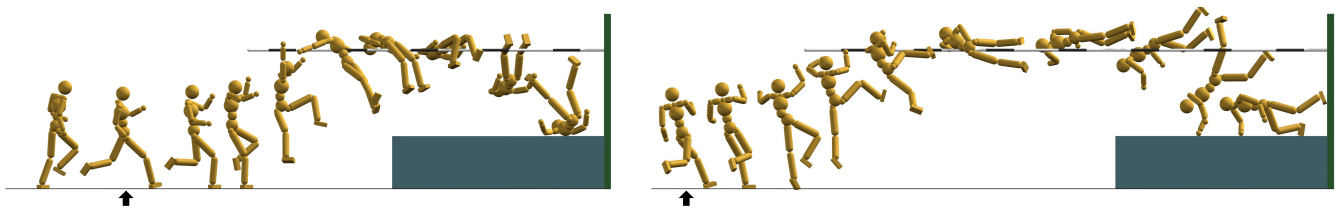
Discovering Diverse Athletic Jumping Strategies

ZHIQI YIN, Simon Fraser University, Canada
ZESHI YANG, Simon Fraser University, Canada
MICHIEL VAN DE PANNE, University of British Columbia, Canada
KANGKANG YIN, Simon Fraser University, Canada



(a) Fosbury Flop – max height=200cm

(b) Western Roll (facing sideways) – max height=195cm

Fig. 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.

We present a framework that enables the discovery of diverse and natural-looking motion strategies for athletic skills such as the high jump. The strategies are realized as control policies for physics-based characters. Given a task objective and an initial character configuration, the combination of physics simulation and deep reinforcement learning (DRL) provides a suitable starting point for automatic control policy training. 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 strategies 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.

CCS Concepts: • **Computing methodologies** → **Animation**; *Physical simulation*; • **Theory of computation** → *Reinforcement learning*.

Additional Key Words and Phrases: physics-based character animation, motion strategy, Bayesian optimization, control policy

ACM Reference Format:
Zhiqi Yin, Zeshi Yang, Michiel van de Panne, and KangKang Yin. 2021. Discovering Diverse Athletic Jumping Strategies. *ACM Trans. Graph.* 40, 4, Article 91 (August 2021), 17 pages. https://doi.org/10.1145/3450626.3459817

Authors' addresses: Zhiqi Yin, Simon Fraser University, Burnaby, Canada, zhiqi_yin@sfu.ca; Zeshi Yang, Simon Fraser University, Burnaby, Canada, zeshi_yang@sfu.ca; Michiel van de Panne, University of British Columbia, Vancouver, Canada, van@cs.ubc.ca; KangKang Yin, Simon Fraser University, Burnaby, Canada, kkyin@sfu.ca.

0730-0301/2021/8-ART91 $15.00
https://doi.org/10.1145/3450626.3459817

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 paper, 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.

In summary, our contributions include:

- A system which can discover common athletic high jump strategies, and execute them using learned controllers and physics-based simulation. 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.

2 RELATED WORK

We build on prior work from several areas, including character animation, diversity optimization, human pose modeling, and high-jump analysis from biomechanics and kinesiology.

2.1 Character Animation

Synthesizing natural human motion is a long-standing challenge in computer animation. We first briefly review kinematic methods, and then provide a more detailed review of physics-based methods. To the best of our knowledge, there are no previous attempts to synthesize athletic high jumps or obstacle jumps using either kinematic or physics-based approaches. Both tasks require precise coordination and exhibit multiple strategies.

Kinematic Methods. Data-driven kinematic methods have demonstrated their effectiveness for synthesizing high-quality human motions based on captured examples. Such kinematic models have evolved from graph structures [Kovar et al. 2002; Safonova and Hodgins 2007], to Gaussian Processes [Levine et al. 2012; Ye and Liu 2010b], and recently deep neural networks [Holden et al. 2017; Lee et al. 2018; Ling et al. 2020; Starke et al. 2019, 2020; Zhang et al. 2018]. Non-parametric models that store all example frames have limited capability of generalizing to new motions due to their inherent nature of data interpolation [Clavet 2016]. Compact parametric models learn an underlying low-dimensional motion manifold. Therefore they tend to generalize better as new motions not in the training dataset can be synthesized by sampling in the learned latent space [Holden et al. 2016]. Completely novel motions and strategies, however, are still beyond their reach. Most fundamentally, kinematic models do not take into account physical realism, which is important for athletic motions. We thus cannot directly apply kinematic methods to our problem of discovering unseen strategies for highly dynamic motions. However, we do adopt a variational autoencoder (VAE) similar to the one in [Ling et al. 2020] as a means to improve the naturalness of our learned motion strategies.

Physics-based Methods. Physics-based control and simulation methods generate motions with physical realism and environmental interactions. The key challenge is the design or learning of robust controllers. Conventional manually designed controllers have achieved significant success for locomotion, e.g., [Coros et al. 2010; Felis and Mombaur 2016; Geijtenbeek et al. 2013; Jain et al. 2009; Lee et al. 2014; Wang et al. 2009, 2012; Yin et al. 2007]. 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 directions, and platform dives [Hodgins et al. 1995; Wooten 1998]. 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. Zhao and van de Panne [2005] thus proposed an interface to ease such a design process, and demonstrated controllers for diving, skiing and snowboarding. Controls can also be designed using objectives and constraints adapted to each motion phase, e.g., [de Lasa et al. 2010; Jain et al. 2009], or developed using a methodology that mimics human coaching [Ha and Liu 2014]. In general, manually designed controllers remain hard to generalize to different strategies or tasks.

With the wide availability of motion capture data, many research endeavors have been focused on tracking-based controllers, which are capable of reproducing high-quality motions by imitating motion examples. Controllers for a wide range of skills have been demonstrated through trajectory optimization [da Silva et al. 2008; Lee et al. 2010, 2014; Muico et al. 2009; Sok et al. 2007; Ye and Liu 2010a], sampling-based algorithms [Liu et al. 2016, 2015, 2010], and deep reinforcement learning [Liu and Hodgins 2018; Ma et al. 2021; Peng et al. 2018a, 2017, 2018b; Seunghwan Lee and Lee 2019]. Tracking controllers have also been combined with kinematic motion generators to support interactive control of simulated characters [Bergamin et al. 2019; Park et al. 2019; Won et al. 2020]. Even though tracking-based methods have demonstrated their effectiveness on achieving task-related goals [Peng et al. 2018a], the imitation objective inherently restricts them from generalizing to novel motion strategies fundamentally different from the reference. Most recently, style exploration has also been demonstrated within a physics-based DRL framework using spacetime bounds [Ma et al. 2021]. However, these remain style variations rather than strategy variations. Moreover, high jumping motion capture examples are difficult to find. We obtained captures of three high jump strategies, which we use to compare our synthetic results to.

Our goal is to discover as many strategies as possible, so example-free methods are most suitable in our case. Various tracking-free methods have been proposed via trajectory optimization or deep reinforcement learning. Heess *et al.* [2017] demonstrate a rich set of locomotion behaviors emerging from just complex environment interactions. However, the resulting motions show limited realism in the absence of effective motion quality regularization. Better motion quality is achievable with sophisticated reward functions and domain knowledge, such as sagittal symmetry, which do not directly generalize beyond locomotion [Coumans and Bai 2019; Mordatch et al. 2015, 2013; Xie et al. 2020; Yu et al. 2018]. Synthesizing diverse physics-based skills without example motions generally requires optimization with detailed cost functions that are engineered specifically for each skill [Al Borno et al. 2013], and often only works for simplified physical models [Mordatch et al. 2012].

2.2 Diversity Optimization

Diversity Optimization is a problem of great interest in artificial intelligence [Coman and Munoz-Avila 2011; Hebrard et al. 2005;

Lehman and Stanley 2011; Pugh et al. 2016; Srivastava et al. 2007; Ursem 2002]. 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 [Agrawal et al. 2013; Merrell et al. 2011]. 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 [Agrawal et al. 2013]. 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 Bayesian Optimization, objective functions are optimized purely through function evaluations as no derivative information is available. A Bayesian statistical *surrogate model*, usually a Gaussian Process (GP) [Rasmussen 2003], is maintained to estimate the value of the objective function along with the uncertainty of the estimation. An *acquisition function* is then repeatedly maximized for fast decisions on where to sample next for the actual expensive 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 [Frazier et al. 2009; Jones et al. 1998; Srinivas et al. 2010]. BO has been widely adopted in machine learning for parameter and hyperparameter optimizations [Kandasamy et al. 2018, 2020; Klein et al. 2017; Korovina et al. 2020; Snoek et al. 2012, 2015]. Recently BO has also seen applications in computer graphics [Koyama et al. 2020, 2017], such as parameter tuning for fluid animation systems [Brochu et al. 2007].

We propose a novel acquisition function to encourage discovery of diverse motion strategies. 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 initial state space defined at the take-off moment. Initial states exploration has been applied to find appropriate initial conditions for desired landing controllers [Ha et al. 2012]. In the context of DRL learning, initial states are usually treated as hyperparameters rather than being explored.

Recently a variety of DRL-based learning methods have been proposed to discover diverse control policies in machine learning, e.g., [Achiam et al. 2018; Conti et al. 2018; Eysenbach et al. 2019; Haarnoja et al. 2018; Hester and Stone 2017; Houthooft et al. 2016; Schmidhuber 1991; Sharma et al. 2019; Sun et al. 2020; Zhang et al. 2019]. These methods mainly encourage exploration of unseen states or actions by jointly optimizing the task and novelty objectives [Zhang et al. 2019], or optimizing intrinsic rewards such as heuristically defined curiosity terms [Eysenbach et al. 2019; Sharma et al. 2019]. 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 high jumping.

2.3 Natural Pose Space

In biomechanics and neuroscience, it is well known that 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 [Overduin et al. 2015; Zhao et al. 2019]. 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 dimension [Holden et al. 2016]. The movement of joints are highly correlated as typically they are strategically coordinated and co-activated. Such correlations have been modelled through traditional dimensionality reduction techniques such as PCA [Chai and Hodgins 2005], or more recently, Variational AutoEncoders (VAE) [Habibie et al. 2017; Ling et al. 2020].

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 nonlinear optimization and improve the motion quality [Safonova et al. 2004]. Throwing motions based on muscle synergies extracted from human experiments have been synthesized on a musculoskeletal model [Cruz Ruiz et al. 2017]. Recent DRL methods either directly imitate mocap examples [Peng et al. 2018a; Won et al. 2020], which makes strategy discovery hard if possible; or adopt a *de novo* approach with no example at all [Heess et al. 2015], which often results in extremely unnatural motions for human like characters. Close in spirit to our work is [Ranganath et al. 2019], 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 set 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 activation can be achieved for powerful take-offs.

Reduced or latent parameter spaces based on statistical analysis of poses have been used for grasping control [Andrews and Kry 2013; Ciocarlie 2010; Osa et al. 2018]. A Trajectory Generator (TG) can provide a compact parameterization that can enable learning of reactive policies for complex behaviors [Iscen et al. 2018]. Motion primitives can also be learned from mocap and then be composed to learn new behaviors [Peng et al. 2019].

2.4 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 [Donnelly 2014]. Over the past 100 years, high jump has evolved dramatically in the Olympics. Here we summarize the well-known variations [Donnelly 2014; Jumper 2020], and we refer readers to our supplemental video for more visual illustrations.

- 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.

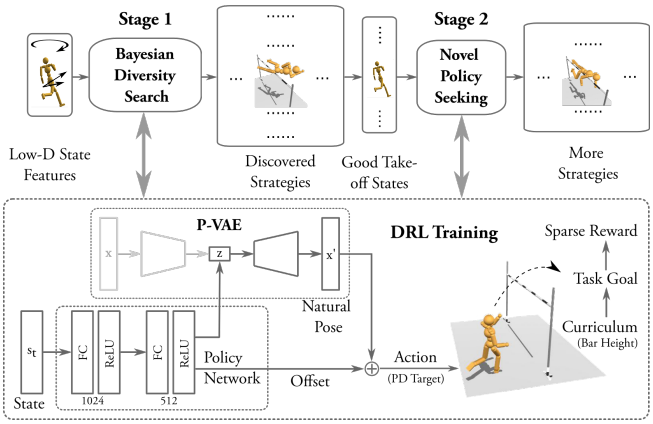


Fig. 2. Overview of our strategy discovery framework. The Stage 1 Bayesian Diversity Search algorithm explores a low-dimensional feature vector of the take-off states to discover multiple jumping strategies. The output strategies from Stage 1 and their corresponding take-off states are taken as input 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.

- 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 slide 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 [Okuyama et al. 2003] in our simulations. Dapena simulated a higher jump by making small changes to a recorded jump [Dapena 2002]. Mathematical models of the Center of Mass (CoM) movement have been developed to offer recommendations to increase the effectiveness of high jumps [Adashevskiy et al. 2013].

3 SYSTEM OVERVIEW

We now give an overview of our learning framework as illustrated in Figure 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 two 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 jump strategy. We characterize the take-off state by a feature vector that captures key aspects of the 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 take-off state features is straightforward, it is computationally impractical as evaluating one sample involves an expensive DRL learning process that 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.

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 deep reinforcement learning, that imitates a single generic run-up motion capture clip while also targeting the desired take-off state. For simplicity, the run-up controller and its training are not shown in Figure 2. These are discussed in Section 6.1.1. 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 task, the sparse fail/success 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 targets to enable strong takeoff 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 the body pose at the peak height of the jump. As shown in Figure 2, Stage 2 makes use of the same overall DRL learning procedure as in Stage 1, albeit with a slightly different reward structure.

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 4.1. To improve the learned motion quality, we propose a Pose Variational Autoencoder (P-VAE) to constrain the policy actions in Section 4.2.

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. The control policy $\pi_\theta(a|s)$ parameterized by θ models the conditional distribution over action $a \in \mathcal{A}$ given the character state $s \in \mathcal{S}$. At each time step t, the character interacts with the environment with action a_t sampled from $\pi(a|s)$ based on the current state s_t. The environment then responds with a new state s_{t+1} according to the transition dynamics $p(s_{t+1}|s_t, a_t)$, along with a reward signal r_t. The goal of reinforcement learning is to learn the optimal policy parameters θ^* which maximizes the expected return defined as

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

where T is the episode length, $\gamma \leq 1$ is a discount factor, and $p_\theta(\tau)$ is the probability of observing trajectory $\tau = \{s_0, a_0, s_1, ..., a_{T-1}, s_T\}$ given the current policy $\pi_\theta(a|s)$.

States. The state s describes the character configuration. We use a similar set of pose and velocity features as those proposed in DeepMimic [Peng et al. 2018a], 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 [Ma et al. 2019], 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 an agent begins each episode in DRL training. We explore a chosen low-dimensional feature space ($3 \sim 4D$) of the take-off states for learning diverse jumping strategies. As shown by previous work [Ma et al. 2021], the take-off moment is a critical point of jumping motions, where the volume of the 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 do not correlate 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 training 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. More details are given in Section 6.1.1.

Actions. The action a is a target pose described by internal joint rotations. We parameterize 1D revolute joint rotations by scalar angles, and 3D spherical joint rotations by exponential maps [Grassia 1998]. Given a target pose and the current character state, joint torques are computed through the Stable Proportional Derivative (SPD) controllers [Tan et al. 2011]. Our control frequency f_{control} ranges from 10 Hz to 30 Hz depending on both the task and the curriculum. For challenging tasks like high jumps, it helps to quickly improve initial policies through stochastic evaluations at early training stages. A low-frequency policy enables faster learning by reducing the needed control steps, or in another word, the dimensionality and complexity of the actions $(a_0, ..., a_T)$. This is in spirit similar to the 10 Hz control fragments used in SAMCON-type controllers [Liu et al. 2016]. Successful low-frequency 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 6.1.3.

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

$$r = r_{\text{task}} \cdot r_{\text{naturalness}} \tag{2}$$

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 4.2. 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 crossbar in Fosbury 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 6.1.2.

Policy Representation. We use a fully-connected neural network parameterized by weights θ to represent the control policy $\pi_\theta(a|s)$. Similar to the settings in [Peng et al. 2018a], 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 policies with the Proximal Policy Optimization (PPO) method [Schulman et al. 2017]. 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. We estimate the advantage of the PPO policy gradient by the Generalized Advantage Estimator GAE(λ) [Schulman et al. 2016].

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 Autoencoder (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 the global root transformations. We use publicly available human motion capture databases 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 successfully.

P-VAE Architecture and Training. Our P-VAE adopts the standard Beta Variational Autoencoder (β-VAE) architecture [Higgins et al. 2017]. 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 covariance 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:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}}(x, x') + \beta \cdot \text{KL}(\mathcal{N}(\mu_x, \Sigma_x), \mathcal{N}(0, I)), \tag{3}$$

where the first term is the MSE (Mean Squared Error) reconstruction loss, and the second term shapes the latent variable distribution to a standard Gaussian by measuring their Kulback-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 20,000 poses obtained from the CMU and SFU motion capture databases. We include a large variety of motion skills, including walking, running, jumping, breakdancing, cartwheels, flips, kicks, martial arts, etc. The input features consist 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 representation for better continuity, as described in [Ling et al. 2020; Zhou et al. 2019].

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 (Principal Component Analysis) 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 [Kingma and Ba 2014], 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 takeoff, large offsets are needed to generate powerful ground reaction forces to propel the body forward or into the air. Such complementary offsets 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:

$$r_{\text{naturalness}} = 1 - \text{Clip}\left(\left(\frac{||a_{\text{offset}}||_1}{c_{\text{offset}}}\right)^2, 0, 1\right), \tag{4}$$

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 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 [Chen et al. 2001; Tibshirani 1996], 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.

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 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 5.1.

The output of Stage 1 is a set of diverse motion strategies and their corresponding initial states. Taken 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 to previously found ones. We describe our detailed settings for the Stage 2 novel policy seeking algorithm in Section 5.2.

5.1 Stage 1: Initial States Exploration with Bayesian Diversity Search

In Stage 1, we perform diverse strategy discovery by exploring initial state 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 6.1.4. 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 representatives $X_n = \{x_1, x_2, ..., x_n | x_i \in \mathcal{X}\}$, 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 (Bayesian Diversity Search) 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 $f_i \in F_t$:

$$a(x_{t+1}) = \min_{f_i \in F_t} ||f_{t+1} - f_i||. \tag{5}$$

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 5.

We maintain the surrogate statistical model of $g(\cdot)$ using a Gaussian Process (GP) [Rasmussen 2003], 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 6.1.5. 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 [van der Wilk et al. 2020]. 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:

$$\begin{aligned} \mu_t(x) &= k(X_t, x)^T (K + \eta^2 I)^{-1} (F_t - m(x)) + m(x), \\ \sigma_t^2(x) &= k(x, x) + \eta^2 - k(X_t, x)^T (K + \eta^2 I)^{-1} k(X_t, x), \end{aligned} \tag{6}$$

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 $k(X_t, x) = [k(x, x_1), k(x, x_2), ...k(x, x_t)]^T$. I is the identity matrix, and η is the standard deviation of the observation noise. Equation 5 can then be approximated by

$$\hat{a}(x_{t+1}) = \mathbb{E}_{\hat{f}_{t+1} \sim \mathcal{N}(\mu_t(x_{t+1}), \sigma_t^2(x_{t+1}))} \left[\min_{f_i \in F_t} ||\hat{f}_{t+1} - f_i|| \right]. \tag{7}$$

Equation 7 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 7 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 consideration 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:

$$a'(x_{t+1}) = \hat{a}(x_{t+1}) + \beta \sigma_t(x_{t+1}), \tag{8}$$

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 [Srinivas et al. 2010]. 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 strategy evaluation function $g(\cdot)$ is extremely costly. To mitigate this problem, we decouple the two terms and alternate between exploration and exploitation following a similar idea proposed in [Song et al. 2019]. During exploration, our acquisition function becomes:

$$a_{\text{exp}}(x_{t+1}) = \sigma_t(x_{t+1}). \tag{9}$$

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

$$x_{t+1} = \arg\max_x \sigma_t(x). \tag{10}$$

Under the condition that $g(\cdot)$ is a sample from GP function distribution $\mathcal{GP}(m(\cdot), k(\cdot, \cdot))$, Equation 10 can be shown to maximize the Information Gain I on function $g(\cdot)$:

$$x_{t+1} = \arg\max_x I\left(X_t \cup \{x\}, F_t \cup \{g(x)\}; g\right), \tag{11}$$

where $I(A; B) = H(A) - H(A|B)$, and $H(\cdot) = \mathbb{E}[-\log p(\cdot)]$ is the Shannon entropy [Cover 1999].

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 8, N_{exp} and N_{opt} are much more intuitive to tune. We also found that empirically the algorithm performance

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.

$t = 0$; $X_0 \leftarrow \emptyset$; $F_0 \leftarrow \emptyset$;
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 9
  else
    $x_{t+1} \leftarrow \arg\max \hat{a}(\cdot)$ by DIRECT; // Equation 7
  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 6
  $t \leftarrow t + 1$;
end
return X_n

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) [Jones 2001], 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 [Liu and Nocedal 1989].

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

$$r_{\text{novelty}}(f) = \text{Clip}\left(\frac{\min_{\pi_i \in \Pi} ||f_i - f||}{d_{\text{threshold}}}, 0.01, 1\right), \tag{12}$$

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 search is in principle similar to the idea of [Sun et al. 2020; Zhang et al. 2019]. However, there are two key differences. First, in machine learning, policy novelty metrics have been designed and validated only on low-dimensional control tasks. For example in [Zhang et al. 2019], the policy novelty is measured by the reconstruction error between 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 for learning novel strategies, such as the Task-Novelty Bisector method proposed in [Zhang et al. 2019] that modifies the policy gradients to encourage novelty learning. Our novel 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.

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 6.1, and implementation details in Section 6.2.

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 the target height 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 20 *cm* 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 50 *cm* 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.

6.1.1 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 diverse jumping strategies. What is lacking is the matching run-up control policies that can prepare the character to reach these good take-off states at the end of the run. We train the run-up controllers with DeepMimic [Peng et al. 2018a], where the 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

Table 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.

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

components 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

$$r_G = \exp\left(-\frac{1}{3} \cdot ||\omega - \bar{\omega}||_1 - 0.7 \cdot (v_z - \bar{v}_z)^2\right), \tag{13}$$

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

$$r_G = \exp(-\frac{1}{3} \cdot ||\omega - \bar{\omega}||_1). \tag{14}$$

6.1.2 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 2 for Stage 1. For Stage 2, the novelty bonus r_{novelty} as discussed in Section 5.2 is added:

$$r = r_{\text{task}} \cdot r_{\text{naturalness}} \cdot r_{\text{novelty}}. \tag{15}$$

$r_{\text{naturalness}}$ is the motion naturalness term discussed in Section 4.2. For both stages, the task reward consists of three terms:

$$r_{\text{task}} = r_{\text{complete}} \cdot r_{\omega} \cdot r_{\text{safety}}. \tag{16}$$

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 9.

6.1.3 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 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 1. The curriculum could also be scheduled following a simpler scheme adopted in [Xie et al. 2020], 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 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.

Table 2. Model parameters of our virtual athlete and the mocap athlete.

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

6.1.4 Strategy Features. We choose low-dimensional and visually discriminate 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 position. The rest of the body limbs are then coordinated in the 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 show in Section 7.1. 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.

6.1.5 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 [Klein et al. 2017; Matérn 1960], defined as:

$$k_{5/2}(x,x') = \theta(1+\sqrt{5}d_\lambda(x,x') + \frac{5}{3}d_\lambda^2(x,x'))e^{-\sqrt{5}d_\lambda(x,x')} \quad (17)$$

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

6.2 Implementation

We implemented our system in PyTorch [PyTorch 2018] and PyBullet [Coumans and Bai 2019]. 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 [Okuyama et al. 2003]. Torque limits for other joints are kept the same as [Peng et al. 2018a]. 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 2. For comparison, we also list these measurements for our virtual athlete.

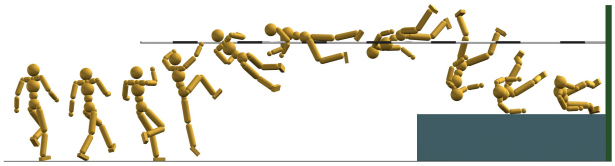
For DRL training, we set $\lambda = 0.95$ for both TD(λ) and GAE(λ). We set the discounter factor $\gamma = 1.0$ since our tasks have short horizon and sparse rewards. The PPO clip threshold is set to 0.02. The learning rate is 2.5×10^{-5} for the policy network and 1.0×10^{-2} for the value network. 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 [van der Wilk et al. 2020] 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. Simulations are run on the CPUs. 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.

7 RESULTS

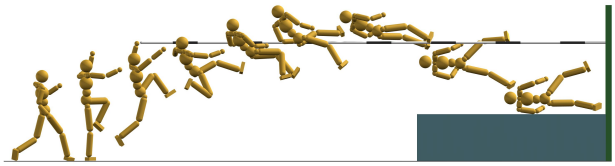
We demonstrate multiple strategies discovered through our framework for high jumping and obstacle jumping in Section 7.1. We validate the effectiveness of BDS and P-VAE in Section 7.2. Comparison with motion capture examples, and interesting variations of learned policies are given in Section 7.3. All results are best seen in the supplementary videos in order to judge the quality of the synthesized motions.

7.1 Diverse Strategies

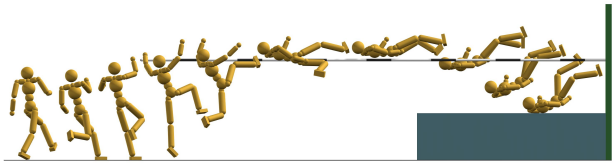
7.1.1 High Jumps. In our experiments, six different high jump strategies are discovered during the Stage 1 initial state exploration within the first ten BDS samples: Fosbury Flop, Western Roll (facing up), Straddle, Front Kick, Side Jump, Side Dive. The first three are high jump techniques standard 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 repetitions 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 hours 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.



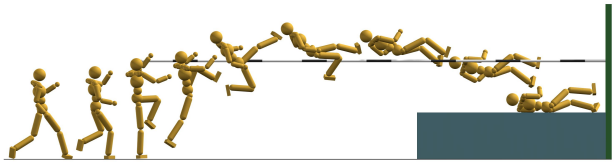
(a) Straddle – max height=190cm



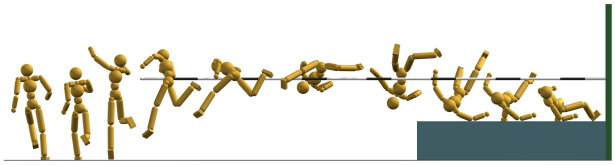
(b) Front Kick – max height=180cm



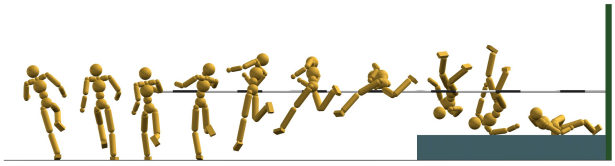
(c) Western Roll (facing up) – max height=160cm



(d) Scissor Kick – max height=150cm



(e) Side Dive – max height=130cm

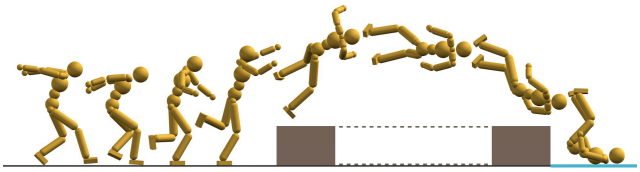


(f) Side Jump – max height=110cm

Fig. 3. Six of the eight high jump strategies discovered by our learning framework, ordered by their maximum cleared height. Fosbury Flop and Western Roll (facing sideways) are shown in Figure 1. Note that Western Roll (facing up) and Scissor Kick differ in the choice of inner or outer leg as the take-off leg. The Western Roll (facing sideways) and the Scissor Kick are learned in Stage 2. All other strategies are discovered in Stage 1.

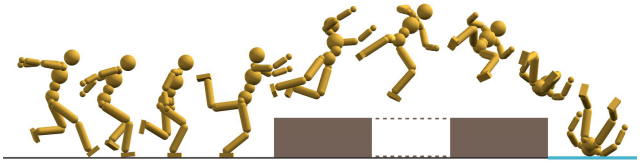
(a) Front Kick – max width=150cm

(b) Side Kick – max width=150cm

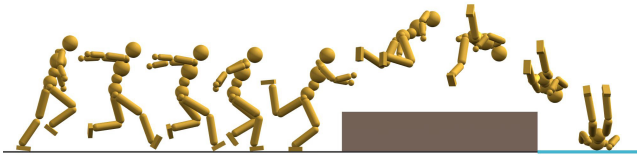


(c) Twist Jump (clockwise) – max width=150cm

(d) Straddle – max width=215cm

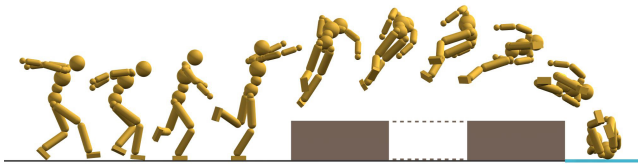


(e) Twist Jump (counterclockwise) – max width=250cm

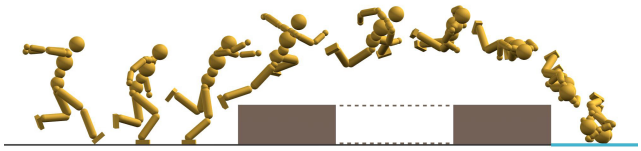


(f) Dive Turn – max width=250cm

Fig. 4. Six obstacle jump strategies discovered by our learning framework in Stage 1, ordered by their maximum cleared obstacle width. For some of the strategies, the obstacle is split into two parts connected with dashed lines to enable better visualization of the poses over the obstacle.



(a) Twist Turn – max width=250cm



(b) Western Roll – max width=250cm

Fig. 5. Two obstacle jump strategies discovered in Stage 2 of our learning framework.

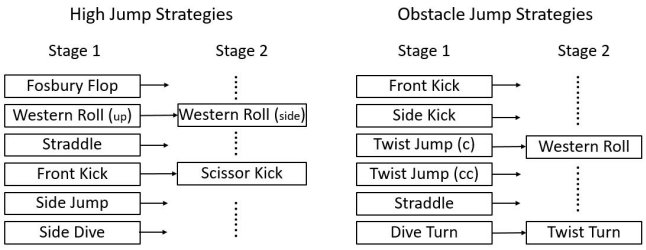
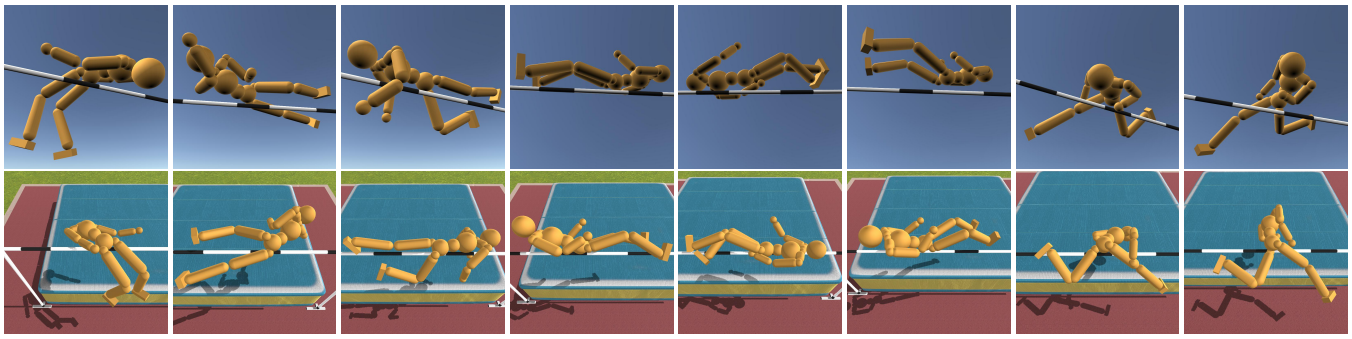


Fig. 6. Diverse strategies discovered in each stage of our framework.

Representative take-off state feature values of the discovered strategies can be found in Appendix A. We also show the DRL learning and curriculum scheduling curves for two strategies in Appendix B.

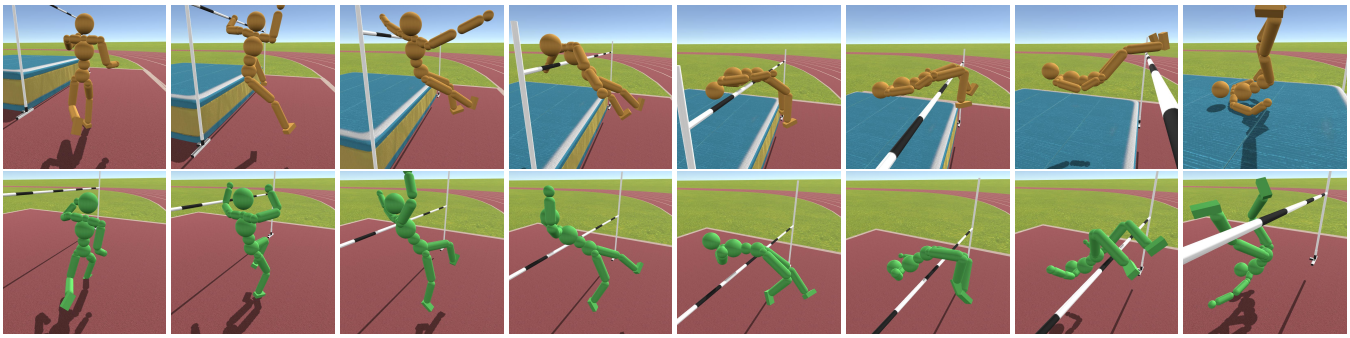
In Stage 2, we perform novel policy search for five DRL iterations from each good initial state of Stage 1. Training is warm started with the associated Stage 1 policy for efficient learning. The total time required for Stage 2 is roughly 60 hours. More strategies are discovered in Stage 2, but most are repetitions and only two of them are novel strategies not discovered in Stage 1: Western Roll (facing sideways) and Scissor Kick. Western Roll (sideways) shares the same initial state with Western Roll (up). Scissor Kick shares the same initial state with Front Kick. The strategies discovered in each stage are summarized in Figure 6. We visualize all eight distinct strategies in Figure 1 and Figure 3. We also visualize their peak poses in Figure 7.

While the final learned control policies are stochastic in nature, the majority of the results shown in our supplementary video are the deterministic version of those policies, i.e., using the mean of the learned policy action distributions. In the video we further show multiple simulations from the final stochastic policies, to help give insight into the true final endpoint of the optimization. As one might expect for a difficult task such as a maximal-height high jump, these

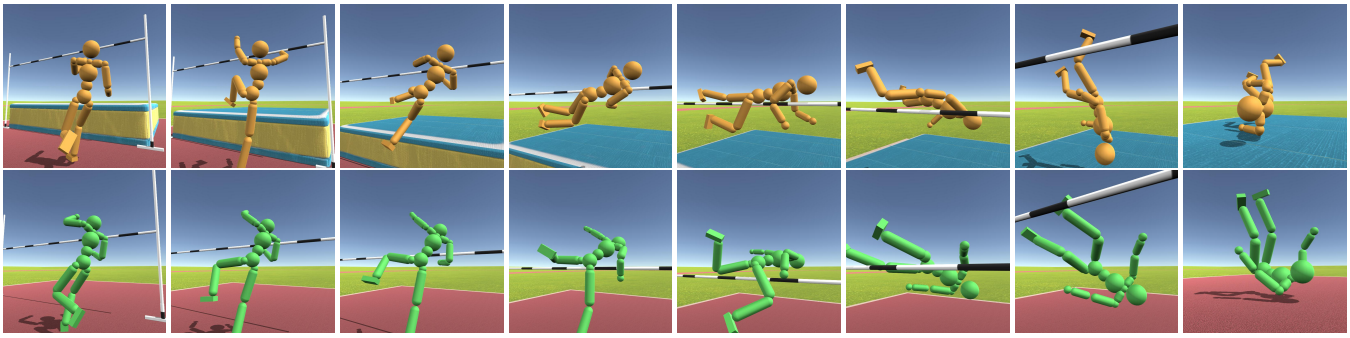


(a) Fosbury Flop (b) Western Roll (c) Straddle (d) Front Kick (e) Western Roll* (f) Scissor (g) Side Dive (h) Side Jump

Fig. 7. Peak poses of discovered high jump strategies, ordered by their maximum cleared height. First row: look-up views; Second row: look-down views.



(a) Fosbury Flop. First row: synthesized – max height=200cm; Second row: motion capture – capture height=130cm.



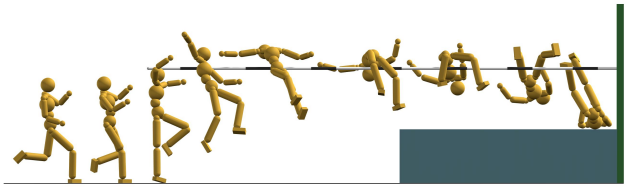
(b) Straddle. First row: synthesized – max height=195cm; Second row: motion capture – capture height=130cm.

Fig. 8. Comparison of our synthesized high jumps with those captured from a human athlete.

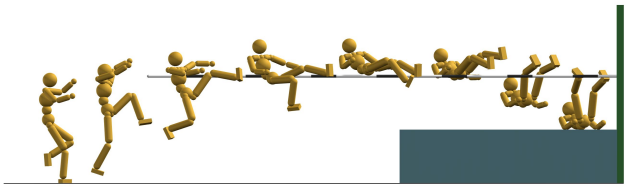
stochastic control policies will also fail for many of the runs, similar to a professional athlete.

7.1.2 Obstacle Jumps. Figure 4 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 5: 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. We encourage readers to watch the supplementary video for better visual perception of the learned strategies.

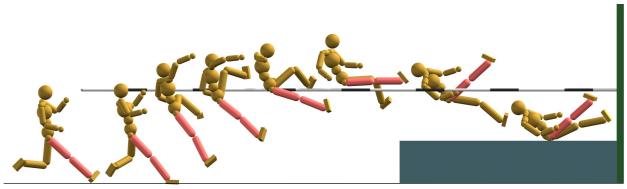
Although our obstacle jump task is not an Olympic event, it is analogous to a long jump in that it seeks to jump a maximum-length jumped. Setting the obstacle height to zero yields a standard long jump task. The framework discovers several strategies, including one similar to the standard long jump adopted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



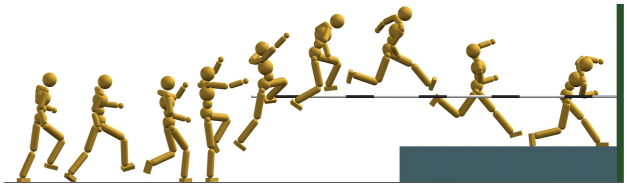
(a) Fosbury Flop – max height=160cm, performed by a character with a weaker take-off leg, whose take-off hip, knee and ankle torque limits are set to 60% of the normal values.



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



(c) Scissor Kick – max height=130cm, learned by a character with a cast on its take-off leg.



(d) Front Kick – max height=120cm, performed with an additional constraint requiring landing on feet.

Fig. 9. 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.

7.2 Validation and Ablation Study

7.2.1 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 7.1.1. 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.

7.2.2 Motion Quality with/without P-VAE. We justify the usage of P-VAE for improving motion naturalness with results shown in Figure 10. Without P-VAE, the character can still learn physically valid skills to complete the tasks successfully, but the resulting motions usually exhibit unnatural behavior. 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.

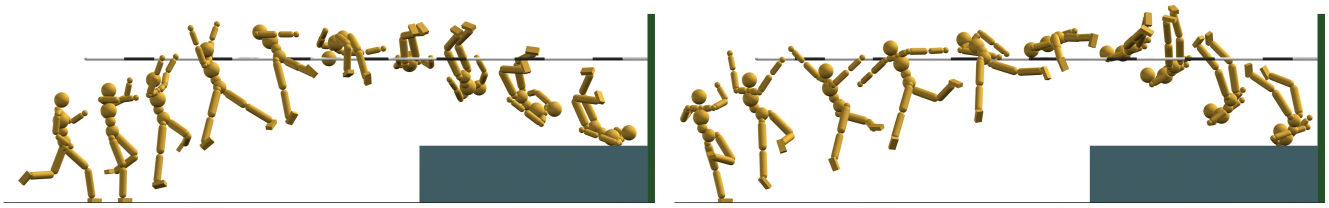
7.3 Comparison and Variations

7.3.1 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 virtual athlete, which is shorter than the real athlete as shown in Table 2. We visualize keyframes sampled from our simulated jumps and the retargeted mocap jumps in Figure 8. Note that the bar heights are set to the maximum heights achievable by our trained policies, while the bar heights for the mocap examples are just the bar heights used at the actual capture session. We did not 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.

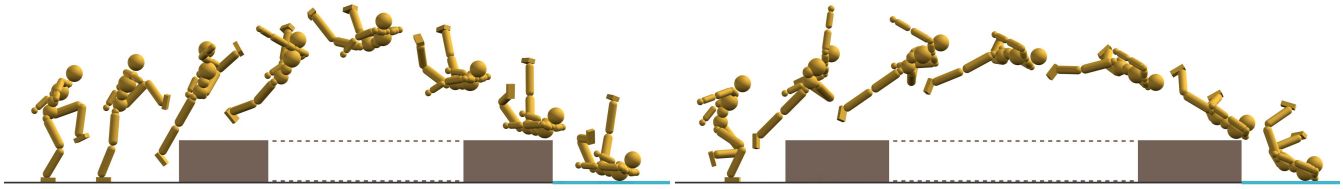
7.3.2 High Jump Variations. In addition to discovering multiple motion strategies, our framework can easily support physically valid motion variations. We show four high jump variations generated from our framework in Figure 9. We generate the first three variations by taking the initial state of the Fosbury Flop strategy discovered in Stage 1, and retrain the jumping policy with additional constraints starting from a random initial policy. Figure 9a shows a jump with a weaker take-off leg, where the torque limits are reduced to 60% of its original values. Figure 9b shows a character jumping with a spine that does not permit backward arching. Figure 9c 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 1a. For the jump in Figure 9d, we take the initial state of the Front Kick, and train with an additional constraint that requires landing on feet. In Figure 11 we also show a high jump trained on Mars, where the gravity $g = 3.711 m/s^2$ is lower, from the initial state of the Fosbury flop discovered on Earth.

8 CONCLUSION AND DISCUSSION

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 jump 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 strategies 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 algorithms in both stages



(a) High jumps trained without P-VAE, given the initial state of Fosbury Flop and Straddle respectively. Please compare with Figure 1a and Figure 3a.



(b) Obstacle jumps trained without P-VAE, given the initial state of Straddle and Twist Jump (cc) respectively. Please compare with Figure 4d and Figure 4e.

Fig. 10. Jumping strategies learned without P-VAE. Although the character can still complete the tasks, the poses are less natural.

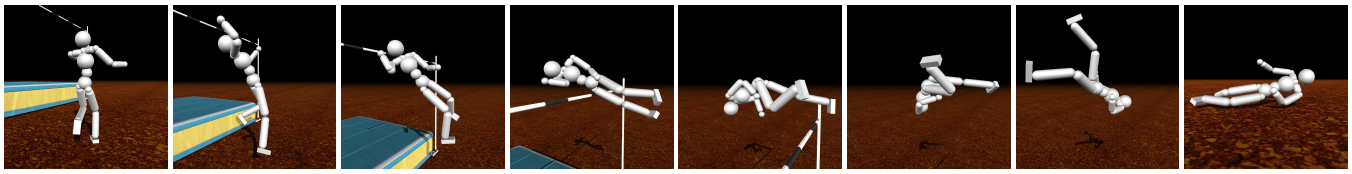


Fig. 11. High jump policy trained on Mars with a lower gravity ($g = 3.711m/s^2$), given the initial state of the Fosbury Flop discovered on Earth.

and the P-VAE are task independent. We leave further adaptation of the proposed framework to additional motor skills as future work. We believe this work demonstrates 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 2021 is 245cm, set in year 1993 by an athlete of 193cm in height. Our high jump record is 200cm for a virtual athlete 170cm tall. The performance and realism of our simulated jumps are bounded by many simplifications in our modeling and simulation. We simplify the athlete's feet and specialized high jump shoes as rigid rectangular boxes, which reduces the 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 fiberglass. 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 affect the quality of not only the run-up controllers, but also the learned jump controllers. This is because the jump controllers couple with the run-up controllers 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 high jumping, likely due to the reference run being curved rather than straight. In both reference runs, there is a low in-place jump after the last running step. We found this jumping intention successfully embedded into the take-off states, 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 6. Stage 2 is only used to search for additional unique strategies and not to fine tune 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 [Zhang et al. 2019]. 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 further 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 incorporate landing and get-ups into our framework is a worthy problem for future studies. Second, there is still room to further improve the 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 large and system dynamics cannot help prune a large portion of the state space as for the jumping tasks. 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 implementation 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 [Azimi et al. 2010], may be worth exploring. The key challenge of such an approach is how to find a set of good candidates to query simultaneously.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive feedback. We thank Beyond Capture for helping us with motion capture various high jumps. We also thank Toptal for providing us the jumping scene. Yin is partially supported by NSERC Discovery Grants Program RGPIN-06797 and RGPAS-522723. Van de Panne is partially supported by NSERC RGPIN-2020-05929.

REFERENCES

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

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

Shailen Agrawal, Shuo Shen, and Michiel van de Panne. 2013. Diverse Motion Variations for Physics-Based Character Animation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 37–44.

Mazen Al Borno, Martin de Lasa, and Aaron Hertzmann. 2013. Trajectory Optimization for Full-Body Movements with Complex Contacts. *TVCG* 19, 8 (2013), 1405–1414.

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

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

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Forbes. 2019. DReCon: data-driven responsive control of physics-based characters. *ACM Transctions on Graphics* 38, 6, Article 206 (2019).

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

Jinxiang Chai and Jessica K Hodgins. 2005. Performance animation from low-dimensional control signals. In *ACM SIGGRAPH 2005 Papers*. 686–696.

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

Matei Ciocarlie. 2010. *Low-Dimensional Robotic Grasping: Eigengrasp Subspaces and Optimized Underactuation*. Ph.D. Dissertation. Columbia University.

Simon Clavet. 2016. Motion Matching and The Road to Next-Gen Animation. In *GCD*.

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

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

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized Biped Walking Control. *ACM Transctions on Graphics* 29, 4 (2010), Article 130.

Erwin Coumans and Yunfei Bai. 2016–2019. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons.

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

Marco da Silva, Yeuhi Abe, and Jovan Popović. 2008. Interactive simulation of stylized human locomotion. In *ACM SIGGRAPH 2008 papers*. 1–10.

Jesus Dapena. 2002. The evolution of high jumping technique: Biomechanical analysis.

Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-based locomotion controllers. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 131.

Sean Donnelly. 2014. *An Introduction to the High Jump*.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2019. Diversity is All You Need: Learning Skills without a Reward Function. In *ICLR*.

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

Peter Frazier, Warren Powell, and Savas Dayanik. 2009. The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing* 21, 4 (2009), 599–613.

Thomas Geijtenbeek, Michiel Van De Panne, and A Frank Van Der Stappen. 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–11.

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

Sehoon Ha and C Karen Liu. 2014. Iterative training of dynamic skills inspired by human coaching techniques. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 1–11.

Sehoon Ha, Yuting Ye, and C Karen Liu. 2012. Falling and landing motion control for character animation. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–9.

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

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

Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. 2005. Finding diverse and similar solutions in constraint programming. In *AAAI*, Vol. 5. 372–377.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. 2017. Emergence of locomotion behaviours in rich environments. *ArXiv* abs/1707.02286 (2017).

Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. 2015. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*. 2944–2952.

Todd Hester and Peter Stone. 2017. Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence* 247 (2017), 170–186.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. 1995. Animating Human Athletics. In *Proceedings of SIGGRAPH 1995*. 71–78.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned Neural Networks for Character Control. *ACM Transctions on Graphics* 36, 4, Article 42 (2017).

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

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

Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. 2018. Policies Modulating Trajectory Generators. In *Proceedings of The 2nd Conference on Robot Learning*. PMLR 87:916–926.

Sumit Jain, Yuting Ye, and C Karen Liu. 2009. Optimization-based interactive motion synthesis. *ACM Transactions on Graphics (TOG)* 28, 1 (2009), 1–12.

Donald R. Jones. 2001. *Direct global optimization algorithmDirect Global Optimization Algorithm*. Springer US, Boston, MA, 431–440.

Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13, 4

(1998), 455–492.
Teen Jumper. 2020. 7 Classic High Jump Styles. https://teenjumper.com/2020/01/04/7-classic-high-jump-styles-and-how-to-do-them/
Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. 2018. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in neural information processing systems*. 2016–2025.
Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, and Eric P Xing. 2020. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *Journal of Machine Learning Research* 21, 81 (2020), 1–27.
Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2017. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics*. PMLR, 528–536.
Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. 2020. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3393–3403.
Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion Graphs. *ACM Transctions on Graphics* 21, 3 (2002), 473–482.
Yuki Koyama, Issei Sato, and Masataka Goto. 2020. Sequential gallery for interactive visual design optimization. *ACM Transactions on Graphics* 39, 4 (Jul 2020). https://doi.org/10.1145/3386569.3392444
Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. 2017. Sequential line search for efficient visual design optimization by crowds. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–11.
Kyungho Lee, Seyoung Lee, and Jehee Lee. 2018. Interactive character animation by learning multi-objective control. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–10.
Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. In *ACM SIGGRAPH 2010 papers*. 1–8.
Yoonsang Lee, Moon Seok Park, Taesoo Kwon, and Jehee Lee. 2014. Locomotion control for many-muscle humanoids. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–11.
Joel Lehman and Kenneth O Stanley. 2011. Novelty search and the problem with objectives. In *Genetic programming theory and practice IX*. Springer, 37–56.
Sergey Levine, Jack M Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. 2012. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–10.
Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. 2020. Character Controllers Using Motion VAEs. *ACM Trans. Graph.* 39, 4 (2020).
Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* 45, 1-3 (1989), 503–528.
Libin Liu and Jessica Hodgins. August 2018. Learning Basketball Dribbling Skills Using Trajectory Optimization and Deep Reinforcement Learning. *ACM Transactions on Graphics* 37, 4 (August 2018).
Libin Liu, Michiel van de Panne, and KangKang Yin. 2016. Guided Learning of Control Graphs for Physics-based Characters. *ACM Transctions on Graphics* 35, 3 (2016), Article 29.
Libin Liu, KangKang Yin, and Baining Guo. 2015. Improving Sampling-based Motion Control. *Computer Graphics Forum* 34, 2 (2015), 415–423.
Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. *ACM Transctions on Graphics* 29, 4, Article 128 (2010).
Li-Ke Ma, Zeshi Yang, Baining Guo, and KangKang Yin. 2019. Towards Robust Direction Invariance in Character Animation. *Computer Graphics Forum* 38, 7 (2019), 1–8.
Li-ke Ma, Zeshi Yang, Xin Tong, Baining Guo, and Kangkang Yin. 2021. Learning and Exploring Motor Skills with Spacetime Bounds. *Computer Graphics Forum* (2021).
B Matérn. 1960. Spatial variation: Meddelanden fran statens skogsforskningsinstitut. *Lecture Notes in Statistics* 36 (1960), 21.
Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive furniture layout using interior design guidelines. *ACM transactions on graphics (TOG)* 30, 4 (2011), 1–10.
Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popović, and Emanuel V Todorov. 2015. Interactive Control of Diverse Complex Characters with Neural Networks. In *Advances in Neural Information Processing Systems*. 3114–3122.
Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM SIGGRAPH* 31, 4 (2012), Article 43.
Igor Mordatch, Jack M Wang, Emanuel Todorov, and Vladlen Koltun. 2013. Animating human lower limbs using contact-invariant optimization. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–8.
Uldarico Muico, Yongjoon Lee, Jovan Popović, and Zoran Popović. 2009. Contact-aware nonlinear control of dynamic characters. In *ACM SIGGRAPH 2009 papers*. 1–9.
K. Okuyama, M. Ae, and T. Yokozawa. 2003. 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)*.
T. Osa, J. Peters, and G. Neumann. 2018. Hierarchical Reinforcement Learning of Multiple Grasping Strategies with Human Instructions. 18 (2018), 955–968.
SA Overduin, A d'Avella, J. Roh, JM Carmena, and E. Bizzi. 2015. Representation of Muscle Synergies in the Primate Brain. *Journal of Neuroscience* 37 (2015). Issue 35.
Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning Predict-and-Simulate Policies From Unorganized Human Motion Data. *ACM Transctions on Graphics* 38, 6, Article 205 (2019).
Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018a. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Transctions on Graphics* 37, 4, Article 143 (2018).
Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)* 36, 4 (2017).
Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 3681–3692.
Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018b. SFV: Reinforcement Learning of Physical Skills from Videos. *ACM Transctions on Graphics* 37, 6, Article 178 (2018).
Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI* 3 (2016), 40.
PyTorch. 2018. PyTorch. https://pytorch.org/.
Avinash Ranganath, Pei Xu, Ioannis Karamouzas, and Victor Zordan. 2019. Low Dimensional Motor Skill Learning Using Coactivation. In *Motion, Interaction and Games*. 1–10.
Carl Edward Rasmussen. 2003. Gaussian processes in machine learning. In *Summer School on Machine Learning*. Springer, 63–71.
Alla Safonova and Jessica K. Hodgins. 2007. Construction and Optimal Search of Interpolated Motion Graphs. *ACM Transctions on Graphics* 26, 3 (2007), 106–es.
Alla Safonova, Jessica K Hodgins, and Nancy S Pollard. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transctions on Graphics* 23, 3 (2004), 514–521.
Jürgen Schmidhuber. 1991. Curious model-building control systems. In *Proc. international joint conference on neural networks*. 1458–1463.
John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017).
Moonseok Park Seunghwan Lee, Kyoungmin Lee and Jehee Lee. 2019. Scalable Muscle-actuated Human Simulation and Control. *ACM Transactions on Graphics (Proc. SIGGRAPH 2019)* 38, 4 (2019).
Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. 2019. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657* (2019).
Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. 2951–2959.
Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. 2015. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*. 2171–2180.
Kwang Won Sok, Manmyung Kim, and Jehee Lee. 2007. Simulating biped behaviors from human motion data. In *ACM SIGGRAPH 2007 papers*. 107–es.
Jialin Song, Yuxin Chen, and Yisong Yue. 2019. A general framework for multi-fidelity bayesian optimization with gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 3158–3167.
Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. 2010. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design *(ICML'10)*. Omnipress, Madison, WI, USA, 1015–1022.
Biplav Srivastava, Tuan Anh Nguyen, Alfonso Gerevini, Subbarao Kambhampati, Minh Binh Do, and Ivan Serina. 2007. Domain Independent Approaches for Finding Diverse Plans.. In *IJCAI*. 2016–2022.
Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Trans. Graph.* 38, 6 (2019), 209–1.
Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 54–1.
Hao Sun, Zhenghao Peng, Bo Dai, Jian Guo, Dahua Lin, and Bolei Zhou. 2020. Novel Policy Seeking with Constrained Optimization. arXiv:2005.10696
Jie Tan, Karen Liu, and Greg Turk. 2011. Stable Proportional-Derivative Controllers. *IEEE Computer Graphics and Applications* 31, 4 (2011), 34–44.

Table 3. Representative take-off state features for discovered high jumps.

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. Representative take-off state features for discovered obstacle jumps.

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

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 267–288.

Rasmus K Ursem. 2002. Diversity-guided evolutionary algorithms. In *International Conference on Parallel Problem Solving from Nature*. Springer, 462–471.

Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, and James Hensman. 2020. A Framework for Interdomain and Multioutput Gaussian Processes. *arXiv:2003.01115* (2020).

Jack M Wang, David J Fleet, and Aaron Hertzmann. 2009. Optimizing walking controllers. In *ACM SIGGRAPH Asia 2009 papers*. 1–8.

Jack M Wang, Samuel R Hamner, Scott L Delp, and Vladlen Koltun. 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), Article 33.

Wayne Lewis Wooten. 1998. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. Ph.D. Dissertation. USA. Advisor(s) Hodgins, Jessica K. AAI9827367.

Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. 2020. ALLSTEPS: Curriculum-driven Learning of Stepping Stone Skills. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Yuting Ye and C Karen Liu. 2010a. Optimal feedback control for character animation using an abstract model. In *ACM SIGGRAPH 2010 papers*. 1–9.

Yuting Ye and C Karen Liu. 2010b. Synthesis of responsive motion using a dynamic model. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 555–562.

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Transctions on Graphics* 26, 3 (2007), Article 105.

Wenhao Yu, Greg Turk, and C Karen Liu. 2018. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics* 37, 4, Article 144 (2018).

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transctions on Graphics* 37, 4, Article 145 (2018).

Yunbo Zhang, Wenhao Yu, and Greg Turk. 2019. Learning novel policies for tasks. *arXiv preprint arXiv:1905.05252* (2019).

K. Zhao, Z. Zhang, H. Wen, Z. Wang, and J. Wu. 2019. Modular Organization of Muscle Synergies to Achieve Movement Behaviors. *Journal of Healthcare Engineering* (2019).

Peng Zhao and Michiel van de Panne. 2005. User Interfaces for Interactive Control of Physics-based 3D Characters. In *I3D: ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5745–5753.

A REPRESENTATIVE TAKE-OFF STATE FEATURES

We list representative take-off state features discovered through BDS in Table 3 for high jumps and Table 4 for obstacle jumps. The approach angle α for high jumps is defined as the wall orientation in a facing-direction invariant frame. The orientation of the wall is given by the line $x\sin\alpha - z\cos\alpha = 0$.

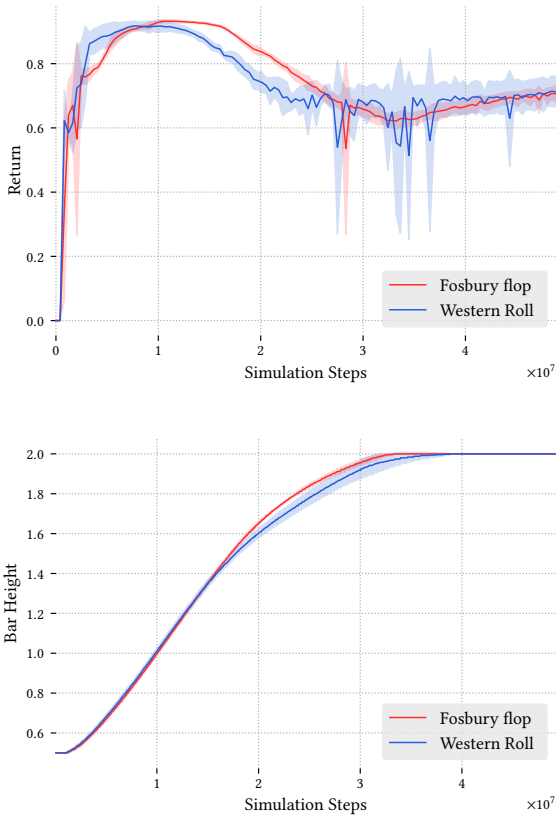


Fig. 12. Stage 1 DRL learning and curriculum scheduling curves for two high jump strategies. As DRL learning is stochastic, the curves shown are the average of five training runs. The shaded regions indicates the standard deviation.

B LEARNING CURVES

We plot Stage 1 DRL learning and curriculum scheduling curves for two high jump strategies in Figure 12. An initial solution for the starting bar height $0.5m$ can be learned relatively quickly. After a certain bar height has been reached (around $1.4m$), the return starts to drop because larger action offsets are needed to jump higher, which decreases the $r_{naturalness}$ in Equation 4 and therefore the overall return in Equation 2. Subjectively speaking, the learned motions remain as natural for high crossbars, as the lower return is due to the penalty on action offsets.