ControlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters

HEYUAN YAO, SCS & KLMP (MOE), Peking University, China
ZHENHUA SONG, SCS & KLMP (MOE), Peking University, China
BAOQUAN CHEN, SIST & KLMP (MOE), Peking University, China
LIBIN LIU*, SIST & KLMP (MOE), Peking University, China

Fig. 1. A simulated character gets up, walks, hops, jumps, and runs in our system. We demonstrate a model-based framework for learning a generative motion control policy based on variational autoencoders (VAE), which allows a simulated character to learn a diverse set of skills and use them to accomplish various downstream tasks.

In this paper, we introduce ControlVAE, a novel model-based framework for learning generative motion control policies based on variational autoencoders (VAE). Our framework can learn a rich and flexible latent representation of skills and a skill-conditioned generative control policy from a diverse set of unorganized motion sequences, which enables the generation of realistic human behaviors by sampling in the latent space and allows high-level control policies to reuse the learned skills to accomplish a variety of downstream tasks. In the training of ControlVAE, we employ a learnable world model to realize direct supervision of the latent space and the control policy. This world model effectively captures the unknown dynamics of the simulation system, enabling efficient model-based learning of high-level downstream tasks. We also learn a state-conditional prior distribution in the VAE-based generative control policy, which generates a skill embedding that outperforms the non-conditional priors in downstream tasks. We demonstrate the effectiveness of ControlVAE using a diverse set of tasks, which allows realistic and interactive control of the simulated characters.

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

Additional Key Words and Phrases: physics-based character animation, motion control, deep reinforcement learning, generative model, VAE

*corresponding author

Authors' addresses: Heyuan Yao, heyuanyao@pku.edu.cn, SCS & KLMP (MOE), Peking University, No.5 Yiheyuan Road, Haidian District, Beijing, Beijing, China, 100871; Zhenhua Song, songzhenhua@stu.pku.edu.cn, SCS & KLMP (MOE), Peking University, No.5 Yiheyuan Road, Haidian District, Beijing, Beijing, China, 100871; Baoquan Chen, baoquan@pku.edu.cn, SIST & KLMP (MOE), Peking University, No.5 Yiheyuan Road, Haidian District, Beijing, Beijing, China, 100871; Libin Liu, libin.liu@pku.edu.cn, SIST & KLMP (MOE), Peking University, No.5 Yiheyuan Road, Haidian District, Beijing, Beijing, China, 100871.

ACM Reference Format:
Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. 2022. ControlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters. *ACM Trans. Graph.* 41, 6, Article 183 (December 2022), 16 pages. https://doi.org/10.1145/3550454.3555434

1 INTRODUCTION

Learning physics-based controllers to realize complex human behaviors has been a longstanding challenge for character animation. Recent research has partially addressed this problem by imitating motion capture data of real human performance using deep reinforcement learning. However, there is a common practice in these approaches where control policies for different tasks are often trained from scratch. Although an extensive range of motions, from basic locomotion to dynamic stunts, have been successfully learned by various agents, how to effectively reuse these learned motion skills to accomplish new tasks remains a challenging problem.

Recent research in kinematic motion synthesis has demonstrated successful applications of generative models, such as variational autoencoders (VAE) [Ling et al. 2020] and normalizing flows [Henter et al. 2020], in learning a rich and versatile latent space to encode a large variety of skills that multiple downstream tasks can reuse. However, these approaches are not directly applicable to learning generative physics-based controllers because their learning objectives are usually defined in the motion domain. The gradients of these objectives typically cannot be back-propagated over the barrier of the simulation, which is often considered as a black box and not differentiable. Even with a differentiable dynamics engine, the highly non-linear nature of the articulated rigid-body system and the contact dynamics often cause the training to stop at poor local optima [Hamalainen et al. 2020; Werling et al. 2021]. A possible way to sidestep this issue is to convert the learning objective of the generative model into a special reward term of reinforcement

learning [Ho and Ermon 2016; Peng et al. 2022, 2021]. However, this approach will require training two coupled indirectly supervised systems, where the training process and the training objective often need to be carefully designed to achieve stable results.

Model-based approaches, or, more specifically, those learning world models [Ha and Schmidhuber 2018], incorporate approximate models to predict the future states of the system given its previous states and the actions taken. The differentiable world model bridges the gap between the simulation and control policy, allowing the objectives defined in the motion domain to supervise the policy directly, thus achieving efficient and stable training [Deisenroth and Rasmussen 2011; Janner et al. 2021]. Recently, model-based reinforcement learning has shown promising results in learning to track complex human motions [Fussell et al. 2021]. The success of these studies suggests a possibility that a more flexible control model, such as a generative model like VAE, can be learned with the help of a learnable world model.

In this paper, we introduce ControlVAE, a novel model-based framework for learning generative motion control policies based on variational autoencoders (VAE). Our framework can learn a rich and flexible latent representation of skills and a skill-conditioned generative control policy from a diverse set of unorganized motion sequences, which enables the generation of realistic human behaviors by sampling in the latent space and allows high-level control policies to reuse the learned skills to accomplish a variety of tasks.

VAEs are commonly trained against the standard normal distribution [Kingma and Welling 2014; Ling et al. 2020]. However, our preliminary experiments show that such a non-conditional prior is not efficient in disentangling the representations of different skills. We thus model our VAE-based generative policy with a prior distribution conditional on the simulated character state. The latent skill embeddings generated from this conditional prior distribution achieve higher performance in downstream tasks than those learned using the non-conditional priors.

We employ a learnable world model to approximate the unknown dynamics of the simulation system, which is trained using online samples. We find this world model not only provides direct supervision for learning the latent space and the control policy, but further allows efficient model-based optimization of control policies of downstream tasks with model-based learning.

To evaluate our method, we train ControlVAE on a diverse set of locomotion skills and test its performance on several challenging downstream tasks. We further conduct studies to validate our design decisions both qualitatively and quantitatively.

2 RELATED WORK

2.1 Physics-based Motion Controllers

Research on developing physics-based control strategies to realize realistic and interactive motions has a long history in computer animation. The seminal works can be dated back to the 1990s, when locomotion control was realized based on careful motion analysis and hand-crafted controllers [Hodgins et al. 1995]. Robust locomotion controllers are later developed using abstract models [Coros et al. 2010; Lee et al. 2010; Yin et al. 2007], optimal control [Muico et al. 2011], model predictive control [Hämäläinen et al. 2015; Mordatch et al. 2010], policy optimization [Tan et al. 2014], and reinforcement learning [Xie et al. 2020; Yin et al. 2021; Yu et al. 2018]. These approaches typically require sufficient prior knowledge and hand-tuned parameters or reward functions, hence can be hard to apply to complex motions and scenarios. Such difficulties motivate the so-called data-driven methods that generate natural motion by imitating human performance. A fundamental task of these approaches is to track a reference motion by learning feedback policies [Lee et al. 2010; Liu et al. 2016, 2012]. The development of deep reinforcement learning techniques further enables robust tracking of agile human motions [Peng et al. 2018] and to generalize to various body shapes [Won and Lee 2019] and environments [Xie et al. 2020].

Based on the success of individual controllers, recent research starts to focus on creating multi-skilled characters. Training a robust tracking policy to track the results of a pre-trained kinematic controller allows the combined control strategy to imitate different target motions according to the task or user input [Bergamin et al. 2019; Park et al. 2019; Won et al. 2020]. However, the performance of such a combined strategy is limited to the capability of the kinematic controller, and it can be computationally expensive to evaluate both the networks of the tracking policy and kinematic controller at runtime. Individual physics-based control policies can be organized into a graph-like structure [Liu et al. 2016; Peng et al. 2018], which can be further broken down into short snippets and managed by a high-level scheduler [Liu and Hodgins 2017]. However, the system needs to maintain all the sub-controllers at runtime for downstream tasks. More recent studies [Luo et al. 2020; Merel et al. 2018, 2020; Peng et al. 2022, 2021; Won et al. 2022] develop various generative models to incorporate a diverse set of motions. The results are compact latent representations of skills that allow high-level policies to reuse multiple skills to accomplish downstream tasks. However, learning efficient representations of skills is a nontrivial problem. We develop several novel components in this work to ensure the performance of the learned skill embeddings in the downstream tasks, which are verified in a series of experiments.

2.2 Generative Models in Motion Control

Generative models have been extensively studied in kinematic motion synthesis. Early research learns statistical representation of motion based on the Gaussian Mixture Model [Min and Chai 2012] and Gaussian Process [Levine et al. 2012; Wang et al. 2008]. In the era of deep learning, the VAEs [Kingma and Welling 2014] have been realized using the mixture-of-expert network [Ling et al. 2020] and transformers [Petrovich et al. 2021] to encode motions into latent spaces. The MoGlow framework proposed by Henter et al. [2020] draws support from the normalizing flows [Kingma and Dhariwal 2018] for probabilistic motion generation. Li et al. [2022] train a GAN-based model to synthesize convincing long motion sequences from a short example. A generative model usually learns a compact latent space that encodes a variety of motions. Sampling in the latent space leads to natural kinematic motions, which allows a task-oriented high-level controller to reuse the motions in downstream tasks using either optimization [Holden et al. 2016; Min and Chai 2012] or learned policies [Levine et al. 2012; Ling et al. 2020].

Physics-based controllers can also be formulated as generative models, where the samples in the latent space will be decoded into specific actions for the simulated character. Merel et al. [2018] learn an autoencoder that distills a large group of expert policies into a latent space, with which a high-level controller can be learned to accomplish difficult tasks like catching and carrying an object [Merel et al. 2020]. Similarly, Won et al. [2022] employ a conditional VAE and perform behavior cloning on expert trajectories. Luo et al. [2020] treat the composing weights of the motor primitives learned by the multiplicative compositional policies (MCP) [Peng et al. 2019] as the representation of skills and achieves interactive locomotion control of a simulated quadruped. Inspired by the generative adversarial imitation learning (GAIL) [Ho and Ermon 2016], Peng et al. [2021] include the adversarial motion prior into the reward function of reinforcement learning, encouraging the simulated character to finish a task using natural-looking actions. They later extend this framework to learn adversarial skill embeddings for a large range of motions [Peng et al. 2022]. Our ControlVAE is based on the conditional VAE. Unlike previous works using a standard normal distribution as the prior distribution of VAE [Ling et al. 2020; Won et al. 2022], we employ a state-conditional prior that creates better latent embeddings. We also employ a model-based learning algorithm to learn our ControlVAE model.

2.3 Model-based Learning

Model-based learning approaches realize control using the underlying model of a dynamic system. The models can be either accurate or approximated using learnable functions. Accurate models often come with differentiable simulators [Mora et al. 2021; Todorov et al. 2012; Werling et al. 2021]. They can be used in either offline or online optimization to construct motion controllers [Eom et al. 2019; Hong et al. 2019; Macchietto et al. 2009; Mora et al. 2021; Mordatch et al. 2010, 2012; Muico et al. 2011]. Approximate models, or the *World Models* [Ha and Schmidhuber 2018], are typically formulated as Gaussian Process [Deisenroth and Rasmussen 2011] or neural networks [Janner et al. 2021; Nagabandi et al. 2018]. They provide differentiable transition functions that allow gradients of learning objectives to pass through the barrier of the simulation [Chiappa et al. 2017; Heess et al. 2015; Schmidhuber 1990], thus enabling policy optimization to be solved efficiently using gradient-based techniques [Deisenroth and Rasmussen 2011; Heess et al. 2015; Janner et al. 2021; Nagabandi et al. 2018].

The application of the reinforcement learning algorithms with learnable world models is rather sparse in physics-based character animation. While early studies in this category had been done in the 1990s [Grzeszczuk et al. 1998], SuperTrack [Fussell et al. 2021] is among the first frameworks that achieve successful tracking of a large diversity of skills. Our work is inspired by SuperTrack [Fussell et al. 2021]. We also learn a world model along with the control policy to achieve efficient training. However, unlike SuperTrack [Fussell et al. 2021], which learns individual tracking controllers, our system learns a VAE-based generative control policy and reusable skill embeddings, which enable multiple downstream tasks without training from scratch each time.

A concurrent study done by Won et al. [2022] shares a similar goal to our work. They also develop a VAE-based control policy and learn a world model to facilitate training. Our method differs from their approach in three ways: (a) Won et al. [2022] employ behavior cloning to train the control policy, which requires the demonstration of many pre-trained expert policies. Instead, our framework allows direct learning of the generative control policies from raw motion clips; (b) our method trains the world model along with the control policy, ensuring they are compatible. We believe this is critical for successful model-based learning of the downstream tasks. The same results are not demonstrated by Won et al. [2022]. The world model is learned offline in their system; (c) we employ a state-conditional prior distribution in our VAE-based model, which outperforms the non-conditional prior in accomplishing downstream tasks. In contrast, Won et al. [2022] use the standard state-independent prior.

3 CONTROL VAE

Figure 2 illustrates the overall structure of ControlVAE. In ControlVAE, a motion control policy is formulated as a conditional distribution $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$. It computes an action $\boldsymbol{a}$ to control the simulated character according to its current state $\boldsymbol{s}$ and a latent variable $\boldsymbol{z} \in \mathcal{Z}$ sampled from a prior distribution $\boldsymbol{z} \sim p(\boldsymbol{z})$. We train this control policy to allow each latent variable $\boldsymbol{z}$ to encode a specific skill, which enables a high-level policy to operate in the latent space $\mathcal{Z}$ to accomplish downstream tasks.

More specifically, at each time step t, the character observes its state $\boldsymbol{s}_t$ and computes an action $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{z}_t)$, where the latent variable $\boldsymbol{z}_t$ is provided by a high-level policy. The character then executes $\boldsymbol{a}_t$ in the simulation and moves to a new state $\boldsymbol{s}_{t+1}$ with the probability of transition given by $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$. This process is then repeated, resulting in a simulated trajectory $\tau^* = \{\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_{T-1}, \boldsymbol{a}_{T-1}, \boldsymbol{s}_T\}$ in which the character keeps moving and performs the skills defined by the sequence of latent codes $\{\boldsymbol{z}_{0:T-1}\}$. Here we assume that $\{\boldsymbol{z}_{0:T-1}\}$ are generated independently of each other.

What we are interested in is the character's motion in this simulated trajectory, represented compactly as $\tau = \{\boldsymbol{s}_{0:T}\}$. To produce realistic skills, we need to train the control policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{z}_t)$ so that the distribution of the generated motions $p(\tau)$ matches the distribution of a motion dataset $\mathcal{D} = \{\tilde{\tau}_i\}$. However, the computation of $p(\tau)$ relies on a very complex likelihood $p(\boldsymbol{s}_{0:T}|\boldsymbol{z}_{0:T-1})$,

$$p(\tau) = \int_{\boldsymbol{z}_{0:T-1}} p(\boldsymbol{z}_{0:T-1}) p(\boldsymbol{s}_{0:T}|\boldsymbol{z}_{0:T-1}), \tag{1}$$

which makes the computation intractable. The variational autoencoder (VAE) [Kingma and Welling 2014] thus considers its evidence lower bound (ELBO). Considering that $\boldsymbol{s}_{t+1}$ and $\boldsymbol{a}_t$ depend only on $\boldsymbol{s}_t$ and $\boldsymbol{z}_t$, also $\{\boldsymbol{z}_t\}$ are independent to each other, the ELBO of Equation (1) can be written as

$$\begin{aligned} \log p(\tau) - \log p(\boldsymbol{s}_0) \geq & \sum_{t=0}^{T-1} \mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{s}_t, \boldsymbol{s}_{t+1})} \left[\log p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{z}_t)\right] \\ & - \sum_{t=0}^{T-1} \mathcal{D}_{\mathrm{KL}} \left(q(\boldsymbol{z}_t|\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) \| p(\boldsymbol{z}_t)\right), \end{aligned} \tag{2}$$

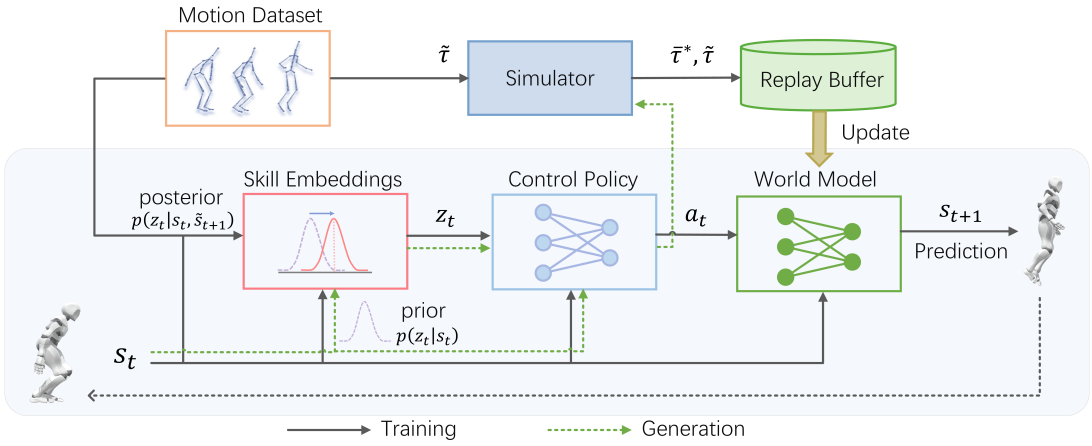


Fig. 2. Overview of our ControlVAE System.

where $q(z_t|s_t, s_{t+1})$ is the so-called variational distribution, which approximates the true posterior distribution of the skill variable $p(z_t|\tau)$ and is assumed to depend on only the state transition, denoted as (s_t, s_{t+1}), $p(s_{t+1}|s_t, z_t)$ represents the probability of this state transition conditional on z_t, and $p(s_0)$ is the distribution of the start states.

The ELBO essentially defines an autoencoder, where the variational distribution $q(z_t|s_t, s_{t+1})$ describes the encoding process that a skill representation z_t is extracted from a pair of consecutive states (s_t, s_{t+1}), and $p(s_{t+1}|s_t, z_t)$ characterizes the decoding process where s_{t+1} is reconstructed from the skill embedding z_t. The KL-divergence in Equation (2) can be considered as a regularization that encourages the approximate posterior to be close to the prior $p(z)$, so that each sample drawn from $p(z)$ can be decoded into a realistic skill.

3.1 Generation

A common choice of the prior $p(z)$ is the standard multivariate normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ [Kingma and Welling 2014; Ling et al. 2020]. However, in our preliminary experiments with this state-independent prior, the character often keeps changing skills quickly, leading to jerky movements and occasional falls. Downstream control policies also perform less efficiently on this prior. For example, the character can encounter difficulties in maintaining a constant moving direction in a direction control task. We presume that this state-independent prior distribution may not provide enough information for the regularization. The encoder thus computes an inconsistent skill distribution, where the same skill is encoded into different parts of the latent space at different states.

To deal with this problem, we leverage a conditional distribution $p(z_t|s_t)$ as the prior and formulate it as a Gaussian distribution with diagonal covariance

$$p(z_t|s_t) \sim \mathcal{N}\left(\boldsymbol{\mu}_p(s_t; \theta_p), \sigma_p^2 \boldsymbol{I}\right), \tag{3}$$

where the mean $\boldsymbol{\mu}_p$ is a neural network parameterized with θ_p and the standard deviation σ is a hyperparameter.

We then generate a motion trajectory according to a sequence of skill codes $\{z_t\}$ by simulating the character with the actions computed by the policy $\pi(a_t|s_t, z_t)$. We model the policy as a Gaussian distribution

$$\pi(a_t|s_t, z_t) \sim \mathcal{N}(\boldsymbol{\mu}_\pi, \Sigma_\pi) \tag{4}$$

with a diagonal covariance matrix Σ_π and the mean $\boldsymbol{\mu}_\pi$ as a neural network $\boldsymbol{\mu}_\pi(s_t, z_t; \theta_\pi)$ parameterized by θ_π. The generation process is then characterized by the log-likelihood in Equation (2)

$$\begin{aligned} \log p(s_{t+1}|s_t, z_t) &= \log \int_{a_t} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t, z_t) \\ &\geq \mathbb{E}_{\pi(a_t|s_t, z_t)} \left[\log p(s_{t+1}|s_t, a_t)\right]. \end{aligned} \tag{5}$$

However, the true transition probability distribution $p(s_{t+1}|s_t, a_t)$ is not known since we consider the simulation process as a black box. We opt to approximate it using a world model $\omega(s_{t+1}|s_t, a_t)$, which is another Gaussian distribution

$$\omega(s_{t+1}|s_t, a_t) \sim \mathcal{N}(\boldsymbol{\mu}_w, \Sigma_w) \tag{6}$$

with a diagonal covariance matrix Σ_w and the mean $\boldsymbol{\mu}_w$ computed as a neural network $\boldsymbol{\mu}_w(s_t, a_t; \theta_w)$ parameterized by θ_w.

3.2 Inference

During the training, we need to infer the approximate posterior distribution of the skill variable $q(z_t|s_t, s_{t+1})$ according to each pair of states (s_t, s_{t+1}) in a state transition. Again, we let $q(z_t|s_t, s_{t+1})$ be a Gaussian distribution

$$q(z_t|s_t, s_{t+1}) \sim \mathcal{N}\left(\hat{\boldsymbol{\mu}}_q, \sigma_q^2 \boldsymbol{I}\right) \tag{7}$$

with a diagonal covariance. Considering that $q(z_t|s_t, s_{t+1})$ needs to stay close to the prior $p(z_t|s_t)$, we let $\sigma_q = \sigma_p$. Rather than directly learning the mean $\hat{\boldsymbol{\mu}}_q$, we learn the residual between $\hat{\boldsymbol{\mu}}_q$ and the mean of the prior, $\boldsymbol{\mu}_p$. Specifically, we compute

$$\hat{\boldsymbol{\mu}}_q = \boldsymbol{\mu}_p + \boldsymbol{\mu}_q, \tag{8}$$

where $\boldsymbol{\mu}_q = \boldsymbol{\mu}_q(s_t, s_{t+1}; \theta_q)$ is a residual network parameterized by θ_q. This structure is inspired by the SRNN [Fraccaro et al. 2016]. It

makes the KL-divergence in Equation (2) independent of the prior $p(\boldsymbol{z}_t|\boldsymbol{s}_t)$ and allows an efficient computation of

$$\mathcal{D}_{\mathrm{KL}}\left(q(\boldsymbol{z}_t|\boldsymbol{s}_t,\boldsymbol{s}_{t+1})\|p(\boldsymbol{z}_t|\boldsymbol{s}_t)\right)=\frac{\boldsymbol{\mu}_q^T\boldsymbol{\mu}_q}{2\sigma_p^2}. \tag{9}$$

The training process can then be viewed as learning how to correct the prior distribution using the information from the next state $\boldsymbol{s}_{t+1}$.

3.3 Training

3.3.1 Control VAE. The training of ControlVAE is then formulated as a maximum likelihood estimation (MLE) problem over the motion dataset $\mathcal{D}=\{\tilde{\tau}_i\}$. We solve this problem by maximizing the ELBO in Equation (2), where the control policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t,\boldsymbol{z}_t)$, the conditional prior $p(\boldsymbol{z}_t|\boldsymbol{s}_t)$, and the approximate posterior $q(\boldsymbol{z}_t|\boldsymbol{s}_t,\boldsymbol{s}_{t+1})$ are trained together. The world model $\omega(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)$ is learned separately, as will be described later.

Instead of evaluating Equation (2) in a time step-wise manner, we compute it over synthetic trajectories. Specifically, we pick a random start state $\tilde{\boldsymbol{s}}_0$ and generate a synthetic trajectory that tracks the corresponding reference clip $\tilde{\tau}$ starting from $\tilde{\boldsymbol{s}}_0$ in $\mathcal{D}$. This is achieved by recursively applying the control policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t,\boldsymbol{z}_t)$ in the world model $\omega(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)$. In this process, the latent variables $\boldsymbol{z}_t$ are sampled from the approximate posterior $q(\boldsymbol{z}_t|\boldsymbol{s}_t,\tilde{\boldsymbol{s}}_{t+1})$, where the future states $\tilde{\boldsymbol{s}}_{t+1}$ in the state transitions are extracted from $\tilde{\tau}$. The reparameterization trick [Kingma and Welling 2014] is used when sampling $\boldsymbol{z}_t$ to ensure the backpropagation of the gradients.

Using all the approximations described above, we can now convert Equation (2) into loss functions. Considering Equation (5), we have

$$\mathcal{L}_{\mathrm{rec}}=\sum_{t=0}^{T-1}\gamma^t\mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{s}_t,\tilde{\boldsymbol{s}}_{t+1}),\pi(\boldsymbol{a}_t|\boldsymbol{s}_t,\boldsymbol{z}_t)}\left[\|\tilde{\boldsymbol{s}}_{t+1}-\boldsymbol{\mu}_w(\boldsymbol{s}_t,\boldsymbol{a}_t)\|_W^2\right] \tag{10}$$

$$\mathcal{L}_{\mathrm{kl}}=\sum_{t=0}^{T-1}\gamma^t\left\|\boldsymbol{\mu}_q(\boldsymbol{s}_t,\tilde{\boldsymbol{s}}_{t+1})\right\|_2^2/2\sigma_p^2, \tag{11}$$

where the states labeled with a tilde ($\tilde{}$) are extracted from the reference clip $\tilde{\tau}$ and those without it are from the synthetic trajectory. Considering that the synthetic trajectory is constructed using the approximate world model, the states in the trajectory can deviate gradually from the true simulation. A discount factor γ is thus employed to lower the weights of these inaccurate states along the trajectory. Here we use $\gamma=0.95$ in this paper.

In addition to the above ELBO losses, following SuperTrack [Fussell et al. 2021], we regularize the magnitude of the actions in terms of both the L_1 and L_2 metrics to prevent excessive control. The corresponding loss term is defined as

$$\mathcal{L}_{\mathrm{act}}=\sum_{t=0}^{T-1}\gamma^t\left(w_{a_1}\|\boldsymbol{a}_t\|_1+w_{a_2}\|\boldsymbol{a}_t\|_2^2\right), \tag{12}$$

which is discounted similarly to the ELBO losses. In practice, we find L_1 term can also be replaced with a larger w_{a_2} weight.

The final objective for training the ControlVAE is then given by

$$\mathcal{L}=\mathcal{L}_{\mathrm{rec}}+\beta\mathcal{L}_{kl}+\mathcal{L}_{\mathrm{act}}. \tag{13}$$

ALGORITHM 1: Training algorithm of ControlVAE

```
Function TrajectoryCollection( Env, ControlVAE):
    Select τ̃ = {s̃_0, s̃_1, ..., s̃_T} from D;
    s_0 ← s̃_0, t ← 0;
    while not terminated do
        /* Get action from ControlVAE                    */
        Sample z_t ~ q(z_t|s_t, s̃_{t+1});
        // or z_t ~ p(z_t|s_t), see Section 3.4.4
        Sample a_t ~ π(a_t|s_t, z_t) ;
        s_{t+1} ← simulation with state s_t and action a_t;
        t ← t + 1;
    end
    Store τ* = {s_0, a_0, s_1, a_1, ..., } and τ̃ = {s̃_0, s̃_1, ...}in B';
end
Function TrainWorldModel(ω, T_w):
    Sample τ̄* = {s̄_0, ā_0, s̄_1, ā_1, ..., } from B;
    s_0 ← s̄_0, L_w ← 0;
    /* Generate synthetic trajectories                   */
    for t ← 0 to T_w − 1 do
        s_{t+1} ← ω(s_t, ā_t);
        L_w ← L_w + ||s_{t+1} − s̄_{t+1}||_{W'}
    end
    Update ω with L_w
end
Function TrainControlVAE( ω, ControlVAE, T_VAE ):
    Sample τ̄* and τ̃ from B;
    s_0 ← s̄_0, L ← 0;
    /* Generate synthetic trajectories                   */
    for t ← 0 to T_VAE − 1 do
        Sample z_t ~ q(z_t|s_t, s̃_{t+1});
        Sample a_t ~ π(a_t|s_t, z_t) ;
        s_{t+1} ← ω(s_t, a_t);
        L ← L + L_rec(s_{t+1}, s̃_{t+1}) + βL_kl + L_act
    end
    Update ControlVAE with L
end
```

Following β-VAE [Higgins et al. 2017], we employ a weight parameter β for the KL-divergence loss, which increases once every 500 training epochs from 0.01 to 0.1 during the training.

3.3.2 World Model. The learning process of the world model largely repeats the same process in SuperTrack [Fussell et al. 2021]. Specifically, $\omega(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)$ is learned based on a collection of simulated trajectories, $\mathcal{B}=\{\tau_j^*\}$. We collect these trajectories by executing the current control policy in the simulation with the start states and skill latent variables extracted from random reference trajectories in $\mathcal{D}$. We then solve an MLE problem again to train $\omega(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)$. Similar to the learning of the ControlVAE, we generate a synthetic trajectory starting from a random state $\bar{\boldsymbol{s}}_0$ in $\mathcal{B}$ by executing the recorded sequence of actions $\{\bar{\boldsymbol{a}}_t\}$ in the world model. The loss function is then computed as

$$\mathcal{L}_w=\sum_{t=0}^{T'-1}\|\bar{\boldsymbol{s}}_{t+1}-\boldsymbol{\mu}_w(\boldsymbol{s}_t,\boldsymbol{a}_t)\|_{W'}^2, \tag{14}$$

where $\{\bar{\boldsymbol{s}}_{t+1}\}$ are the recorded states corresponding to $\{\bar{\boldsymbol{a}}_t\}$.

3.3.3 Training Process. During training, we first collect a number of simulated trajectories then update the world model and the ControlVAE in tandem as illustrated in Algorithm 1.

Trajectory collection. At the beginning of each training epoch, we select a start state $\boldsymbol{s}_0$ from a random trajectory $\tilde{\tau}$ in the motion dataset $\mathcal{D}$. Then the control policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{z}_t)$ is evaluated to track $\tilde{\tau}$, using the latent skill variable sampled from the posterior distribution $q(\boldsymbol{z}_t|\boldsymbol{s}_t, \tilde{\boldsymbol{s}}_{t+1})$, conditional on the current state $\boldsymbol{s}_t$ of the character and the corresponding next reference state $\tilde{\boldsymbol{s}}_{t+1}$ in $\tilde{\tau}$. The character then performs the action $\boldsymbol{a}_t$ in the simulation, resulting in a new state $\boldsymbol{s}_{t+1}$. This process is repeated until a termination condition is satisfied. We then store the simulated trajectory τ^* and corresponding reference trajectory $\tilde{\tau}$ in a temporary buffer $\mathcal{B}'$, and start a new trajectory from another random state. This trajectory collection procedure is ended when the number of states in $\mathcal{B}'$ exceeds a predefined size $N_{B'}$. Then the temporary buffer $\mathcal{B}'$ is merged into $\mathcal{B}$, replacing the oldest trajectories and keeping the size of $\mathcal{B}$ smaller than N_B. Here we use $N_B = 5 \times 10^4$ and $N_{B'} = 2048$.

We employ an early termination strategy to prevent $\mathcal{B}$ from recording too many bad simulation samples. A simulation will be terminated if the trajectory is longer than $T_{\max} = 512$ time steps or if the tracking error of the character's head has exceeded $d_{\max} = 0.5\,m$ for more than $T_{\text{term}} = 1$ second.

Update world model. We then update the world model as described in Section 3.3.2. More specifically, we extract a batch of N_{w} random clips $\{\bar{\tau}^*\}$ of length T_{w} from $\mathcal{B}$, where each $\bar{\tau}^* = \{\bar{\boldsymbol{s}}_{0:T_{\text{w}}}, \bar{\boldsymbol{a}}_{0:T_{\text{w}}-1}\}$. Then N_{w} synthetic trajectories are generated by unrolling the world model using $\bar{\boldsymbol{s}}_0$ and $\{\bar{\boldsymbol{a}}_{0:T_{\text{w}}-1}\}$ to compute the loss in Equation (14). The world model is then updated using the gradients of the loss function. This updating process is repeated 8 times in each training epoch with $T_{\text{w}} = 8$ and $N_{\text{w}} = 512$.

Update ControlVAE. At last, the ControlVAE is updated as described in Section 3.3.1. Similar to the training of the world model, we generate a batch of $N_{\text{VAE}} = 512$ synthetic trajectories to evaluate the loss functions in Equation (13), where each trajectory has $T_{\text{VAE}} = 24$ frames. To ensure a good coverage over the state space, the start states of these synthetic trajectories are randomly selected from the simulation buffer $\mathcal{B}$, and the corresponding motion clips in $\mathcal{D}$ are extracted as the reference. We update the ControlVAE models 8 times before starting the next training epoch.

3.4 Implementation

3.4.1 Policy Representation.

State. Our physics-based character is modeled as articulated rigid bodies with a floating root joint. Its state can be fully characterized by $\boldsymbol{s}^* = \{\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{v}_i, \boldsymbol{\omega}_i\}, i \in B$, where B stands for the set of rigid bodies and $\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{v}_i, \boldsymbol{\omega}_i$ are the position, orientation, linear velocity, and angular velocity of each rigid body, respectively. Similar to SuperTrack [Fussell et al. 2021], we convert the global state of the character $\boldsymbol{s}^*$ into the local coordinate frame of the root, and use the result, $\boldsymbol{s}$, as the input to the ControlVAE and the world model. Specifically, we define $\boldsymbol{s} = \{\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{q}}_i, \bar{\boldsymbol{v}}_i, \bar{\boldsymbol{\omega}}_i, h_i, \boldsymbol{y}_0\}, i \in B$, where

$$(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{q}}_i, \bar{\boldsymbol{v}}_i, \bar{\boldsymbol{\omega}}_i) = \boldsymbol{q}_0^{-1} \otimes (\boldsymbol{x}_i - \boldsymbol{x}_0, \boldsymbol{q}_i, \boldsymbol{v}_i, \boldsymbol{\omega}_i), \tag{15}$$

h_i is the height of each rigid body, and $\boldsymbol{y}_0$ is the *up* axis of the root's local coordinate frame. The rotations $\boldsymbol{q}$ are represented in 6D representation [Zhou et al. 2019], which are commonly used in recent research on motion synthesis [Fussell et al. 2021; Lee et al. 2021]. They can be computed by extracting the first two columns of a rotation matrix.

With this state representation, the reconstruction loss of Equation (10) is implemented in practice using the 1-norm distance between two states as

$$\mathcal{L}_{\text{rec}} = \sum_{t=0}^{T-1} \gamma^t \left[\|W(\tilde{\boldsymbol{s}}_{t+1} - \boldsymbol{s}_{t+1})\|_1\right], \tag{16}$$

where $W = \text{diag}(w_{\bar{x}}, w_{\bar{q}}, w_{\bar{v}}, w_{\bar{\omega}}, w_h, w_y)$ is a diagonal weight matrix that balances the magnitude of each component.

Action. We actuate our character using PD controllers. The action $\boldsymbol{a} = \{\hat{\boldsymbol{q}}_j\}, j \in J$ is thus a collection of target rotations of all the joints, where J stands for the set of joints. We use 3D axis angles to represent those target rotations.

Latent. We employ a latent space $\mathcal{Z}$ with dimension 64 to encode the motion skills. The means of the conditional prior $p(\boldsymbol{z}_t|\boldsymbol{s}_t)$ and the approximate posterior $q(\boldsymbol{z}_t|\boldsymbol{s}_t, \boldsymbol{s}_{t+1})$ of the latent variable are both modeled as neural networks with two 512-unit hidden layers and the ELU as the activation function. To emphasize the state information, we concatenate the input of each layer of the prior and posterior networks with $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t+1}$, respectively. We use $\sigma_{\boldsymbol{p}} = 0.3$ as the standard deviation of these distributions.

Policy. The mean $\boldsymbol{\mu}_\pi$ of the control policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ is modeled as a neural network. We employ a mixture-of-expert (MoE) structure similar to the decoder network of MotionVAE [Ling et al. 2020]. Specifically, we use six expert networks in this structure, each having three 512-unit hidden layers. The parameters of these experts are blended according to the weights computed by a gating network, which contains two 64-unit hidden layers. The ELU is used as the activation function for these networks. To ensure the effectiveness of the latent skill variable, $\boldsymbol{z}$ is concatenated with the input of each layer of the experts, and layer normalization is applied to normalize these concatenated inputs. The covariance matrix Σ_π of the policy distribution is set as a diagonal matrix $\sigma_\pi^2 \boldsymbol{I}$, where $\sigma_\pi = 0.05$.

3.4.2 World Model. Our world model $\omega(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \mathcal{N}(\boldsymbol{\mu}_w, \Sigma_w)$ is formulated and trained in the same way as SuperTrack [Fussell et al. 2021]. We use a neural network with four 512-unit hidden layers and ELU activation functions for the world model $\boldsymbol{\mu}_w(\boldsymbol{s}, \boldsymbol{a})$. The input to $\boldsymbol{\mu}_w(\boldsymbol{s}, \boldsymbol{a})$ is the state representation $\boldsymbol{s}$ and the target joint rotations of the action $\boldsymbol{a}$ converted into quaternions. The world model then predicts the change of the velocity and angular velocity of each rigid body, represented by $d\bar{\boldsymbol{v}}_i$ and $d\bar{\boldsymbol{\omega}}_i$, in the local coordinate frame of the root. Then, the character's new state $\boldsymbol{s}^*_{t+1}$ and thus $\boldsymbol{s}_{t+1}$ are computed accordingly using the forward Euler method.

The objective function of Equation (14) is then implemented to penalize the prediction error in the global coordinate frame:

$$\mathcal{L}_w = \sum_{t=0}^{T'-1} \left\|\tilde{\boldsymbol{s}}^*_{t+1} - \boldsymbol{s}^*_{t+1}\right\|^2_{W'}, \tag{17}$$

where the weight matrix $W' = \text{diag}(w_x, w_q, w_v, w_\omega)$ is chosen empirically to balance the magnitude of each component.

Our experiments show that large randomness can negatively impact the stableness of the training process. In practice, we directly use the mean function $\boldsymbol{\mu}_w$ to generate the synthetic rollouts instead of sampling from $\mathcal{N}(\boldsymbol{\mu}_w, \Sigma_w)$.

3.4.3 Data Balancing. Our motion dataset $\mathcal{D}$ consists of several unstructured long motion sequences of a diverse set of skills, including walking, running, jumping, turning, and more difficult skills such as getting up. During the trajectory collection process of the training, if we select the initial states uniformly from $\mathcal{D}$, easy motions may often result in longer simulated trajectories and thus occupy the simulation buffer $\mathcal{B}$, making the ControlVAE hard to learn difficult skills. To overcome this unbalancing problem, we calculate the *value* of each state and sample the initial states according to their values. The state with a lower value will have higher chance to be selected.

More specifically, we maintain a list $\mathcal{V} = \{V_t\}$ of the values of every state in the dataset $\mathcal{D}$, where all the V_t are initialized to zero. After every 200 epochs in the training, if a state with index t in $\mathcal{D}$ is encountered in the collected simulated trajectories, we update the corresponding value V_t as

$$V_t^* = \sum_{k=0}^{T-t} \gamma^k r_{t+k} + \gamma^{T+1-t} V_{T+1} \tag{18}$$

$$V_t = (1-\alpha)V_t + \alpha V_t^* \tag{19}$$

where r_{t+k} is the reward of state $\boldsymbol{s}_{t+k}$ in the simulated trajectory, T is the length of the trajectory containing $\boldsymbol{s}_t$, and the discount factor $\gamma = 0.95$. We compute the reward according to the reconstruction loss of Equation (16) as

$$r_t = e^{-\|W(\tilde{\boldsymbol{s}}_t - \boldsymbol{s}_t)\|_1 / T_V}, \tag{20}$$

where the temperature $T_V = 20$. In the next 200 training epochs, the initial state will be selected based on the probability proportional to $1/\max(0.01, V_t)$.

3.4.4 Trajectory Sampler. At last, to encourage the policy to learn new transitions between different motions, we augment the trajectory collection process by switching the reference motion clip randomly during the simulation. The beginning state of the new motion clip is also selected using the above data balancing strategy. In practice, many of such switches will cause the character to fall due to mismatched poses and velocities, but the character can learn to recover from such falls automatically during the training.

To encourage the policy to embed diverse motion transitions into prior distribution $p(\boldsymbol{z}|\boldsymbol{s})$, we further augment the trajectory collection process with states derived from the prior distribution. Specifically, the policy randomly chooses to sample latent codes from the posterior distribution $q(\boldsymbol{z}_t|\boldsymbol{s}_t, \tilde{\boldsymbol{s}}_{t+1})$ or the prior distribution $p(\boldsymbol{z}_t|\boldsymbol{s}_t)$. The probability of choosing the latter is empirically set to 0.4. In practice, the skill embeddings learned with this augmentation behave more actively and responsively in downstream tasks.

4 MODEL-BASED HIGH-LEVEL CONTROLLERS

A learned ControlVAE provides a latent skill space $\mathcal{Z}$ and a skill-conditional policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ that a high-level policy can leverage to accomplish various downstream tasks. Formally, the task policy $\pi(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{g})$ takes the character state $\boldsymbol{s}$ and a task-specific parameter $\boldsymbol{g}$ as input and computes a skill variable $\boldsymbol{z}$. The skill policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ then uses $\boldsymbol{z}$ to compute the action $\boldsymbol{a}$ accordingly. The task policy can be trained using model-free reinforcement learning algorithms as suggested in previous systems [Ling et al. 2020; Luo et al. 2020; Merel et al. 2020; Peng et al. 2019]. However, the world model learned with ControlVAE further enables more efficient model-based learning for the downstream tasks.

In this section, we introduce two model-based control strategies that can take advantage of the learned world model, as well as a corpus of locomotion tasks that can be accomplished using these controllers. Note that the parameters of the learned ControlVAE are frozen in this stage. Only the gradients are passed through the networks to optimize the downstream task policies. The character applies the learned policies in the true simulation and responds dynamically to user control and unexpected perturbations.

4.1 Model Predictive Control

The first control strategy is the sampling-based model predictive control (MPC). At each time step, we generate N_{MPC} synthetic Monte-Carlo rollouts $\{\tau_i^{\boldsymbol{g}} = \{\boldsymbol{z}_t, \boldsymbol{a}_t, \boldsymbol{s}_t\}^i\}$ of a fixed planning horizon T_{MPC} using the world model, where the skill variables $\boldsymbol{z}_t$ of these rollouts are sampled from the state-conditional prior distribution $p(\boldsymbol{z}_t|\boldsymbol{s}_t)$ of the ControlVAE. We then evaluate each rollout with a task-specific loss function. The first action of the best trajectory will be used in the simulation. In practice, we find that $N_{\text{MPC}} = 128$ and $T_{\text{MPC}} = 4$ can achieve good performance in our experiments.

4.2 Model-based Learning of Task Policies

Our ControlVAE allows fast model-based policy training for downstream tasks based on the world model. Specifically, we assume that the task policy $\pi(\boldsymbol{z}_t|\boldsymbol{s}_t, \boldsymbol{g})$ has the same structure as the approximate posterior distribution $q(\boldsymbol{z}_t|\boldsymbol{s}_t, \boldsymbol{s}_{t+1})$ of ControlVAE, which is again a Gaussian distribution $\mathcal{N}(\hat{\boldsymbol{\mu}}_g, \sigma_g^2 \boldsymbol{I})$ with a diagonal covariance $\sigma_g = \sigma_q$ and the mean function computed as

$$\hat{\boldsymbol{\mu}}_g = \boldsymbol{\mu}_p + \boldsymbol{\mu}_g, \tag{21}$$

where $\boldsymbol{\mu}_g = \boldsymbol{\mu}_g(\boldsymbol{s}, \boldsymbol{g}; \theta_g)$ is a neural network parameterized by θ_g. During the training, we generate a batch of $N_{\text{ML}} = 256$ synthetic rollouts $\{\tau_i^{\boldsymbol{g}}\}$ for a fixed horizon $T_{\text{ML}} = 16$ and update the policy $\boldsymbol{\mu}_g$ by minimizing the task-specific loss function. The reparameterization trick is also applied in this process to ensure the backpropagation of the gradients to the policy.

To generate the initial states for synthesizing these rollouts, we apply the current task policy in the simulated environment with the corresponding goal parameters $\boldsymbol{g}$ changing randomly every 72 steps. These simulated states and goal parameters are then stored in a buffer $\mathcal{B}_{\boldsymbol{g}}$ with size 4096. During the training, a synthetic rollout is generated by sampling an initial state $\boldsymbol{s}_0$ from $\mathcal{B}_{\boldsymbol{g}}$ and executing the task policy using the recorded sequence of task parameters $\{\boldsymbol{g}_t\}$ following $\boldsymbol{s}_0$.

4.3 Tasks

In this section, we introduce a corpus of locomotion tasks that can be accomplished by the above model-based control strategies.

We formulate these tasks as a set of loss functions computed over each of the generated trajectories $\tau^{\boldsymbol{g}}$. Using the skill embeddings learned in ControlVAE, the corresponding task policies can generate natural motions to complete these tasks in simulation and respond to unexpected perturbations.

Unless otherwise stated, all our loss functions have the form

$$\mathcal{L}(\tau^{\boldsymbol{g}}) = \sum_{t=1}^{T} \left[\mathcal{L}_{\boldsymbol{g}}(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t) + \mathcal{L}_{\text{fall}}(\boldsymbol{s}_t)\right] + w_z \sum_{t=0}^{T-1} \|\boldsymbol{\mu}_g\|_2^2, \tag{22}$$

where $\mathcal{L}_{\boldsymbol{g}}$ stands for the task-specific loss computed using the current state $\boldsymbol{s}_t$ and optionally the previous state $\boldsymbol{s}_{t-1}$, and $\mathcal{L}_{\text{fall}}(\boldsymbol{s}_t)$ penalizes potential falling as

$$\mathcal{L}_{\text{fall}}(\boldsymbol{s}_t) = \max(h_0^* - h_0, 0), \tag{23}$$

where h_0 is the height of the character's root, $h_0^* = 0.5\,m$ is a falling threshold. The last term of Equation (22) is a regularization term that encourages the high-level policy to stay close to the prior distribution. We find this term crucial to a stable and successful training. $w_z = 20$ works for all our experiments.

Height control. We define a simple loss to control the height of the character as

$$\mathcal{L}_{\boldsymbol{g}} = H \cdot h_0, \tag{24}$$

where h_0 represents the current height of the character's root joint. The task parameter $H \in \{-1, 1\}$ indicates the desired motion for the character, where $H = -1$ encourages the character to squat down and eventually lie on the ground and $H = 1$ will let the character get up and jump when possible to maintain a high root position.

Heading control. In the heading control task, the character needs to travel while heading towards a target direction $\theta_h^* \in [-\pi, \pi]$ at a given speed $v^* \in [0.0, 3.0]\ m/s$ along that direction. The loss function is thus defined as

$$\mathcal{L}_{\boldsymbol{g}} = w_{\theta_h} |\theta_h^* - \theta_h| + w_v \frac{|v^* - v|}{\max(v^*, 1)}, \tag{25}$$

where θ_h and v are the character's current heading direction and speed. We compute θ_h according to the orientation of the character's root and v as the component of the root's velocity along this direction. The weights are set to $(w_{\theta_h}, w_v) = (2.0, 1.0)$ for this task. Note that we normalize the speed loss according to the target speed to encourage the training to pay attention to these low-speed motions.

Steering control. The objective of the steering task is to control both the heading direction θ_h and the travel direction θ_v simultaneously. Specifically, we define the loss function as

$$\mathcal{L}_{\boldsymbol{g}} = w_{\theta_h} |\theta_h^* - \theta_h| + w_{\theta_v} |\theta_v^* - \theta_v| + w_v \frac{|v^* - \|\bar{\boldsymbol{v}}_0\|_2|}{\max(v^*, 1)}, \tag{26}$$

where $\bar{\boldsymbol{v}}_0$ represents planar components of the root's linear velocity $\boldsymbol{v}_0$ and θ_v is its directional angle, computed as the angle between $\bar{\boldsymbol{v}}_0$ and the x-axis of the global coordinate frame. The weights are set similarly as $(w_{\theta_h}, w_{\theta_v}, w_v) = (2.0, 2.0, 1.0)$ for this task.

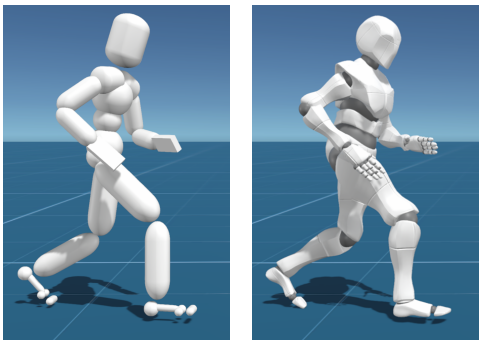
ALGORITHM 2: Model-based training algorithm of task policy

Function GenSynTraj($\omega, \boldsymbol{s}_0, p(\boldsymbol{z}|\boldsymbol{s}_t, *)$ **) :**
  `// p(z|s_t, *) is a conditional distribution of z`
  for $t \leftarrow 0$ **to** $T_{SL} - 1$ **do**
    Sample $\boldsymbol{z}_t \sim p(\boldsymbol{z}|\boldsymbol{s}_t, *)$;
    Sample $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{z}_t)$;
    $\boldsymbol{s}_{t+1} \leftarrow \omega(\boldsymbol{s}_t, \boldsymbol{a}_t)$;
  end
end
Function TrainSkillCtrl(**Env**, **ControlVAE**, ω, π, D, C **) :**
  `// D is the discriminator and C is the classifier`
  Collect simulated trajectory $\bar{\tau}^*$ with random goals $\{\boldsymbol{g}_t\}$;
  Store $\bar{\tau}^*$ and $\{\boldsymbol{g}_t\}$ into $\mathcal{B}$;
  `/* Synthesize real rollouts */`
  Select motion clips $\tilde{\tau}$ with a random skill label $\boldsymbol{c}$ in $\mathcal{D}$;
  $\tau^{\boldsymbol{c}} = \{\boldsymbol{s}_0^{\boldsymbol{c}}, \boldsymbol{z}_0^{\boldsymbol{c}} \dots\} \leftarrow$ **GenSynTraj(** $\tilde{\boldsymbol{s}}_0, q(\boldsymbol{z}|\boldsymbol{s}_t, \tilde{\boldsymbol{s}}_{t+1})$ **)** ;
  `/* Synthesize fake rollouts */`
  Select $\bar{\tau}^*$ and $\{\boldsymbol{g}_t\}$ from $\mathcal{B}$;
  $\tau^{\boldsymbol{g}} = \{\boldsymbol{s}_0, \boldsymbol{s}_1, \dots\} \leftarrow$ **GenSynTraj(** $\bar{\boldsymbol{s}}_0, \pi(\boldsymbol{z}|\boldsymbol{s}_t, \boldsymbol{g}_t)$ **)** ;
  Calculate $\mathcal{L}_{\boldsymbol{g}'}$, $\mathcal{L}_D$, $\mathcal{L}_C$ using $\tau^{\boldsymbol{g}}$;
  `/* Regularization term */`
  for $t \leftarrow 0$ **to** $T_{SL} - 1$ **do**
    $\boldsymbol{g}_t^{\boldsymbol{c}'} \leftarrow$ calculate goal parameters from $\boldsymbol{s}_t^{\boldsymbol{c}}, \boldsymbol{s}_{t+1}^{\boldsymbol{c}}$;
    $\boldsymbol{z}_t^{\boldsymbol{c}'} \sim \pi(\boldsymbol{z}|\boldsymbol{s}_t^{\boldsymbol{c}}, \boldsymbol{g}_t^{\boldsymbol{c}'})$;
    Calculate $\mathcal{L}_{\text{reg}}$ using $\boldsymbol{z}_t^{\boldsymbol{c}'}, \boldsymbol{z}_t^{\boldsymbol{c}}$;
  end
  `/* Update network parameters */`
  Update π with Equation (31);
  Update D using $\tau^{\boldsymbol{c}}, \tau^{\text{fake}}$ with Equation (27);
  Update C using $\tilde{\tau}, \boldsymbol{c}$ with Equation (30);
end

Skill control. The objective of the skill control is to let the character use a specific skill to accomplish a given task, such as responding to steering control while jumping or hopping. In this task, a skill is specified using a 4-second motion clip manually selected from the dataset. We have selected 5 skills for this task, including walking, running, hopping, jumping, and skipping, as shown in Figure 11. The task parameter $\boldsymbol{c}$ is thus defined as a one-hot vector indicating the index of the skill.

Unfortunately, we do not yet have a direct mapping between a specific skill $\boldsymbol{c}$ and its corresponding latent code $\boldsymbol{z}$, so that the character has to figure out by itself whether it is performing the correct skill corresponding to the reference motion clip. Inspired by AMP [Peng et al. 2021], which learns an adversarial discriminator to enforce a specific motion style in reinforcement learning, we employ a similar discriminator in our model-based learning process that penalizes incorrect motions as an adversarial loss function.

More specifically, we train the task-specific control policy to mimic the behavior of a tracking policy characterized by the approximate posterior $q(\boldsymbol{z}_t|\boldsymbol{s}_t, \boldsymbol{s}_{t+1})$, as illustrated in Algorithm 2. In each training iteration, we first generate a batch of control rollouts $\{\tau^{\boldsymbol{g}}\}$ as described in Section 4.2, where each $\tau^{\boldsymbol{g}}$ is conditioned on a skill



(a) Simulation Model (b) Visualization Model

Fig. 3. The character model used in this paper.

vector $\boldsymbol{c}$. Then, for each $\tau^{\boldsymbol{g}}$, we extract a short random reference clip with the same length from the reference motion of skill $\boldsymbol{c}$. By tracking this short reference clip using the approximate posterior $q(\boldsymbol{z}_t|\boldsymbol{s}_t, \tilde{\boldsymbol{s}}_{t+1})$, where $\boldsymbol{s}_t$ is the generated state and $\tilde{\boldsymbol{s}}_{t+1}$ is from the reference clip, we obtain a tracking rollout $\tau^{\boldsymbol{c}}$. We then consider $\{\tau^{\boldsymbol{c}}\}$ as the *real* data samples and the control rollouts $\{\tau^{\boldsymbol{g}}\}$ as the *fake* data samples and employ a discriminator D to distinguish between them. Following Peng et al. [2021], we adapt a least-squares GAN [Mao et al. 2017] in this training. The adversarial loss for the discriminator D is defined as

$$\begin{aligned}\arg\min_D \; & \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+1}\sim\{\tau^{\boldsymbol{c}}\}}\left[(D(\boldsymbol{s}_t,\boldsymbol{s}_{t+1};\boldsymbol{c})-1)^2\right] \\ & + \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+1}\sim\{\tau^{\boldsymbol{g}}\}}\left[(D(\boldsymbol{s}_t,\boldsymbol{s}_{t+1};\boldsymbol{c})+1)^2\right] \\ & + w_g\mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+1}\sim\{\tau^{\boldsymbol{c}}\}}\left[\|\nabla_{\boldsymbol{s}_t,\boldsymbol{s}_{t+1}}D(\boldsymbol{s}_t,\boldsymbol{s}_{t+1};\boldsymbol{c})\|^2\right]. \end{aligned} \tag{27}$$

Note that the discriminator D is also conditioned on the skill vector $\boldsymbol{c}$. The last term of the above equation regularizes the gradient of the discriminator with respect to the real data samples, which allows a more stable training [Peng et al. 2021]. We set its weight $w_g = 20$.

The objective of the character is now to complete a target task, *e.g.* heading control, while behaving indistinguishably from the reference motion. This can be achieved using the adversarial loss

$$\mathcal{L}_D = (D(\boldsymbol{s}_t,\boldsymbol{s}_{t+1};\boldsymbol{c})-1)^2, \tag{28}$$

where $\boldsymbol{s}_t, \boldsymbol{s}_{t+1} \sim \{\tau^{\boldsymbol{g}}\}$.

To further facilitate the character to perform the correct skill in the skill set, we train a separate skill classifier C to predict the possiblity that a state transition $(\boldsymbol{s}_t, \boldsymbol{s}_{t+1})$ belongs to a specific skill $\boldsymbol{c}$. It is trained with the transitions in the *real* data samples $\{\tau^{\boldsymbol{c}}\}$

$$\arg\min_C \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+1}\sim\{\tau^{\boldsymbol{c}}\}}\left[\mathcal{H}\left[C(\boldsymbol{s}_t,\boldsymbol{s}_{t+1}),c\right]\right], \tag{29}$$

where $\mathcal{H}$ is the cross-entropy loss. Then, we add a classifier loss

$$\mathcal{L}_C = \mathcal{H}\left[C(\boldsymbol{s}_t,\boldsymbol{s}_{t+1}),c\right] \tag{30}$$

to the task's loss function, where again $\boldsymbol{s}_t, \boldsymbol{s}_{t+1} \sim \{\tau^{\boldsymbol{g}}\}$.

The final loss function for the style control is defined as

$$\mathcal{L}_{\boldsymbol{g}} = \mathcal{L}_{\boldsymbol{g}'} + w_D\mathcal{L}_D + w_C\mathcal{L}_C + w_r\mathcal{L}_{\text{reg}}, \tag{31}$$

where $\mathcal{L}_{\boldsymbol{g}'}$ is the loss function of the target task. The last term of Equation (31), $\mathcal{L}_{\text{reg}}$, is a regularization term inspired by [Luo et al. 2020]. It considers the result of synthetically tracking the reference clip $\tau^{\boldsymbol{c}}$ of skill $\boldsymbol{c}$ as the output of a perfect policy finishing the task specified by $\tau^{\boldsymbol{c}}$ itself. The task policy being trained thus need to clone the behavior of this perfect policy. To achieve this, we extract the task parameters $\{\boldsymbol{g}_t^{\boldsymbol{c}'}\}$ from $\tau^{\boldsymbol{c}}$ and assume that the posterior $\boldsymbol{z}_t^{\boldsymbol{c}} \sim q(\boldsymbol{z}_t|\boldsymbol{s}_t, \tilde{\boldsymbol{s}}_{t+1})$ is the output of the perfect policy. The regularization term is then defined as

$$\mathcal{L}_{\text{reg}} = \|\boldsymbol{z}_t^{\boldsymbol{c}} - \boldsymbol{z}_t^{\boldsymbol{c}'}\|_1, \tag{32}$$

where $\boldsymbol{z}_t^{\boldsymbol{c}'} \sim \pi(\boldsymbol{z}|\boldsymbol{s}_t, \boldsymbol{g}_t^{\boldsymbol{c}'})$ is the prediction of the current task policy for the task. This term is crucial to maintain a stable training in practice. We use $(w_D, w_C, w_r) = (5.0, 0.5, 10.0)$ in our experiments.

Table 1. Length of each motion after resampling

Motion	Frames (20 fps)
Walk	5227
Run	4757
Jump	4889
Getup	3365

5 RESULTS

5.1 System Setup

As shown in Figure 3, we simulate a character model that is 1.6 m tall, weighs 49.5 kg, and consists of 20 rigid bodies. A uniform set of PD control parameters $(k_p, k_d) = (400, 50)$ is used for all the joints, except for the toe joints $(10, 1)$ and the wrist joints $(5, 1)$. The character is simulated using the Open Dynamics Engine (ODE) at 120 Hz. The stable-PD mechanism [Liu et al. 2013; Tan et al. 2011] is implemented to ensure a stable simulation with the large timestep.

Our system executes ControlVAE, the world model, and all the task-specific policies at 20 Hz. It is lower than the simulation frequency, thus the same action is used in the simulation until the control policies are evaluated next time. ControlVAE assumes no knowledge about the simulation. It extracts the position and orientation of each rigid body from the physics engine and computes the corresponding velocities and angular velocities using finite difference. We implement and train the ControlVAE using PyTorch [Paszke et al. 2019]. Once trained, the entire system runs faster than real-time on a desktop, allowing interactive control of the simulated character.

5.2 ControlVAE training

We train our ControlVAE on an unstructured motion capture dataset consisting of four long motion sequences selected from the open-source LaFAN dataset [Harvey et al. 2020] as shown in Table 1. This dataset contains a diverse range of motions, including walking, running, turning, hopping, jumping, skipping, falling, and getting up. The motion sequences are downsampled to 20 fps and augmented with their mirrored sequences, resulting in an augmented dataset with approximately 30 minutes high-quality motion.

We train the ControlVAE models using the RAdam [Liu et al. 2020] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and the learning rates of policy and world model are (1e-5, 2e-3). Following the standard technique for achieving a robust training, the gradient norms are

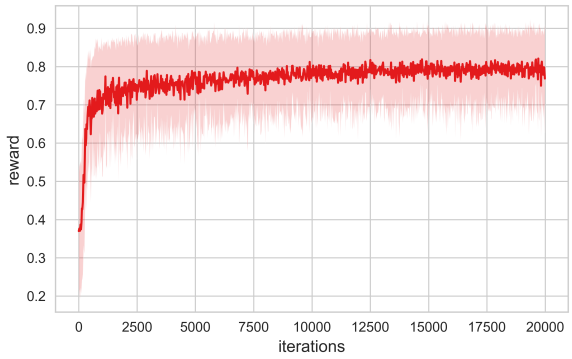


Fig. 4. A typical learning curve of ControlVAE.

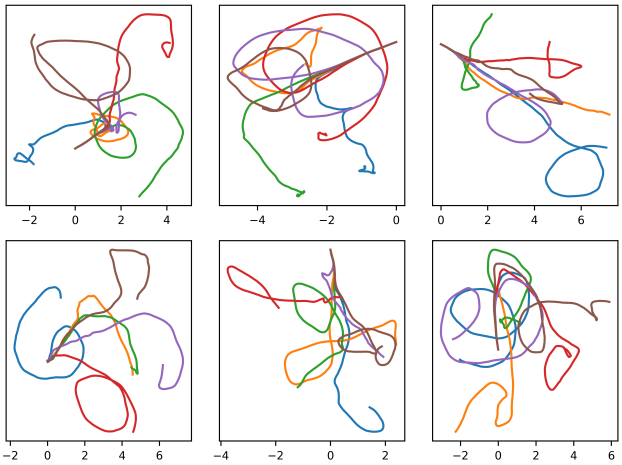


Fig. 5. Visualization of the random trajectories generated in the random sampling experiment, showing the planar positions of the character. These trajectories are generated by drawing random latent codes from the prior distribution of a learned ControlVAE model. Starting from the same initial states, the simulated character moves towards different directions while performing realistic motions.

clipped between $[-1, 1]$. Figure 4 shows a typical learning curve of ControlVAE, where the reward is calculated using Equation 20 on the simulated trajectories collected during the training. The training process begins to converge after about 10,000 iterations, and the motion quality keeps improving in the following training. Our full training takes 20,000 iterations and about 50 hours with four parallel working threads on an Intel Xeon Gold 6240 @ 2.60GHz CPU and one NVIDIA GTX 2080Ti graphics card.

5.3 Evaluation

We evaluate the effectiveness of a learned ControlVAE using two simple tasks.

Reconstruction. To show that the ControlVAE has the same ability as other autoencoders in reconstructing an input motion, we use the learned approximate posterior $q(\boldsymbol{z}_t|\boldsymbol{s}_t, \boldsymbol{s}_{t+1})$ as a tracking control and let ControlVAE to track motion sequences randomly chosen from the dataset. The character performs the input motion accurately in the simulation, and when another random clip is given, it automatically performs a smooth transition and then tracks the target motion again. In some extreme cases where the difference between current state and target motion is too large, the character may fall on the ground. It then automatically discovers a recovery strategy to get up, which is not included in the dataset.

Random sampling. The ControlVAE learns a generative control policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ conditional on the latent skill variable $\boldsymbol{z}$, which allows us to generate a diverse range of behaviors even by drawing random samples in the latent space $\mathcal{Z}$. Specifically, given a random initial state, we draw samples from the state-conditional prior distribution $p(\boldsymbol{z}|\boldsymbol{s})$ at every control step. The simulated character then performs random skills smoothly, such as stopping and then starting to walk, taking random turns, hopping and skipping for a short period, etc. Figure 5 shows the root trajectories of several sample trajectories, where the character starts from a random state and performs random actions for 200 control steps, or 10 seconds. It can be seen that our ControlVAE generates a diverse range of motions in this random walk test.

5.4 Model Predictive Control

We test our MPC strategy on the height and heading control tasks. With interactive user input, the character lies down, gets up, and moves toward the target direction. The MPC policy can achieve real-time performance on the multicore computer we used to train ControlVAE with an NVIDIA GTX 2080Ti graphics card. Due to the limited number of samples, our sampling-based MPC cannot guarantee smooth and accurate results, but the resulting motions are generally acceptable. In practice, MPC can be used as an experimental tool to test our training settings, for example, to see whether a loss function is suitable for the task.

5.5 Model-based Training

Training settings. We model the task policy $\boldsymbol{\mu}_g$ using a neural network with three hidden layers, each having 256 units for heading control task and 512 for all other tasks. We use the RAdam [Liu et al. 2020] optimizer with the learning rate decaying exponentially as $0.001 * \max(0.99^{\text{iteration}}, 0.1)$. We find this exponentially decaying learning rate improves the stability of the training process.

Heading control. In this task, our character is able to move and respond to user input such as changing the moving direction and speed. It automatically transits between different skills to achieve the target direction and speed (see Figure 10(b)). It is also capable of resisting external perturbation or recovering from tumbling when moving (see Figure 10(f) and 10(g)). Figure 6 shows the response of the character to a speed change. We can see that our character reacts quickly to adapt to the speed change and stops at last as the final target speed is zero.

Figure 10(a) shows a derived task of going towards a target location. We achieve this by computing the target direction and speed

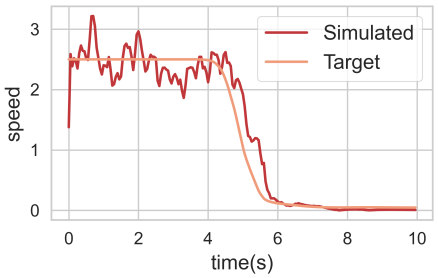


Fig. 6. Visualization of the character's speed in the heading control task. The character slows down and stops under user control.

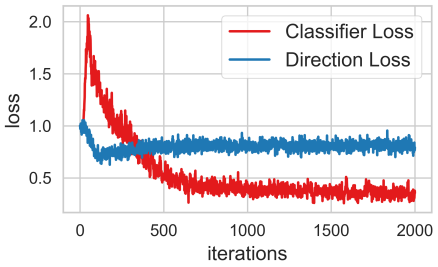


Fig. 7. Typical learning curves of the skill control task. Red: the classifier loss. Blue: the direction loss. A lower classifier loss suggests that the learned policy can generate motions of the target skill. A lower direction loss indicates that the policy can accomplish the heading control task. The curves are normalized based on the values of the first iteration.

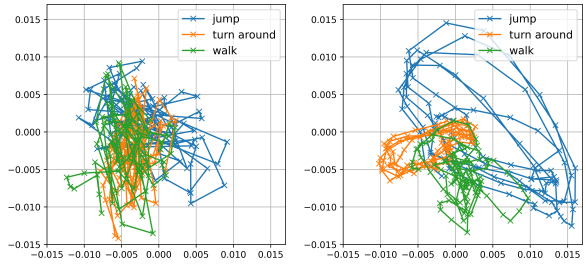
according to the relative distance between the character and the goal. The character can move towards the goal and stop when reaching it.

Steering control. In the steering control task, our character learns to move to the target direction while facing towards another direction. As shown in Figure 10(c), the character learns to walk sideways under user control. It demonstrates a more complex foot pattern compared with that used in simply moving forward.

Skill control. In the skill control task, our character learns to accomplish the heading control task using a user-specified skill. Both the discriminator D and classifier C are modeled as neural networks with two 256-unit hidden layers. They are updated with the RAdam optimizer with the learning rates 0.0001 and 0.01, respectively. As shown in Figure 10(d) and Figure 10(e), the character can turn while maintaining the given style. Figure 7 shows the learning curve of this task. We can see that as the training processes, the policy learns the heading control first and gradually grasps the styles.

5.6 Comparison

In this section, we conduct several experiments to justify our design choices and the performance of the ControlVAE.



(a) VAE with non-conditional prior (b) ControlVAE

Fig. 8. Visualization of different skills in the latent space.

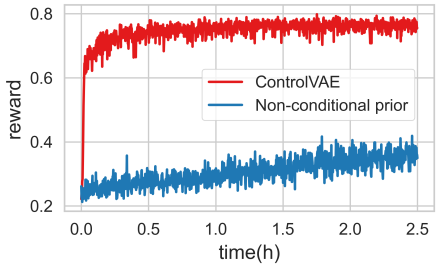


Fig. 9. Training curves of the heading control policies using ControlVAE and VAE with non-conditional prior.

Comparison with VAE with non-conditional prior. One of the key components of ControlVAE is the state-conditional prior distribution. To show the effectiveness of this design, we train a different ControlVAE with the standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ as the prior distribution. This can be achieved equivalently by defining the posterior as $q(\boldsymbol{z}_t|\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) \sim \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{I})$ and sampling $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ when needed in the training. We refer to the ControlVAE with this configuration as the *VAE with non-conditional prior* (VAE-NC).

Figure 8 shows a comparison between the learned latent spaces of ControlVAE and VAE-NC, where we encode three motion clips of different skills into the latent spaces using the learned approximate posterior distributions. Generally speaking, the state-conditional prior allows each skill to be embedded into the latent space more continuously than the standard normal distribution, and the latent codes of different skills are more separated. This can be further justified by downstream tasks. In the random sampling test, the character controlled by VAE-NC behaves more unstably, often falls on the ground, and struggles to get up. Also in the heading control task, the character controlled by VAE-NC can hardly follow the target direction but keep turning and falling. The corresponding learning curve in Figure 9 also shows that the VAE with non-conditional prior performs worse than ControlVAE.

Comparison with reinforcement learning. Many recent works achieve successful hierarchical control policies using model-free reinforcement learning algorithms, where a task policy is trained on top of a learned skill latent space [Ling et al. 2020; Luo et al. 2020; Merel

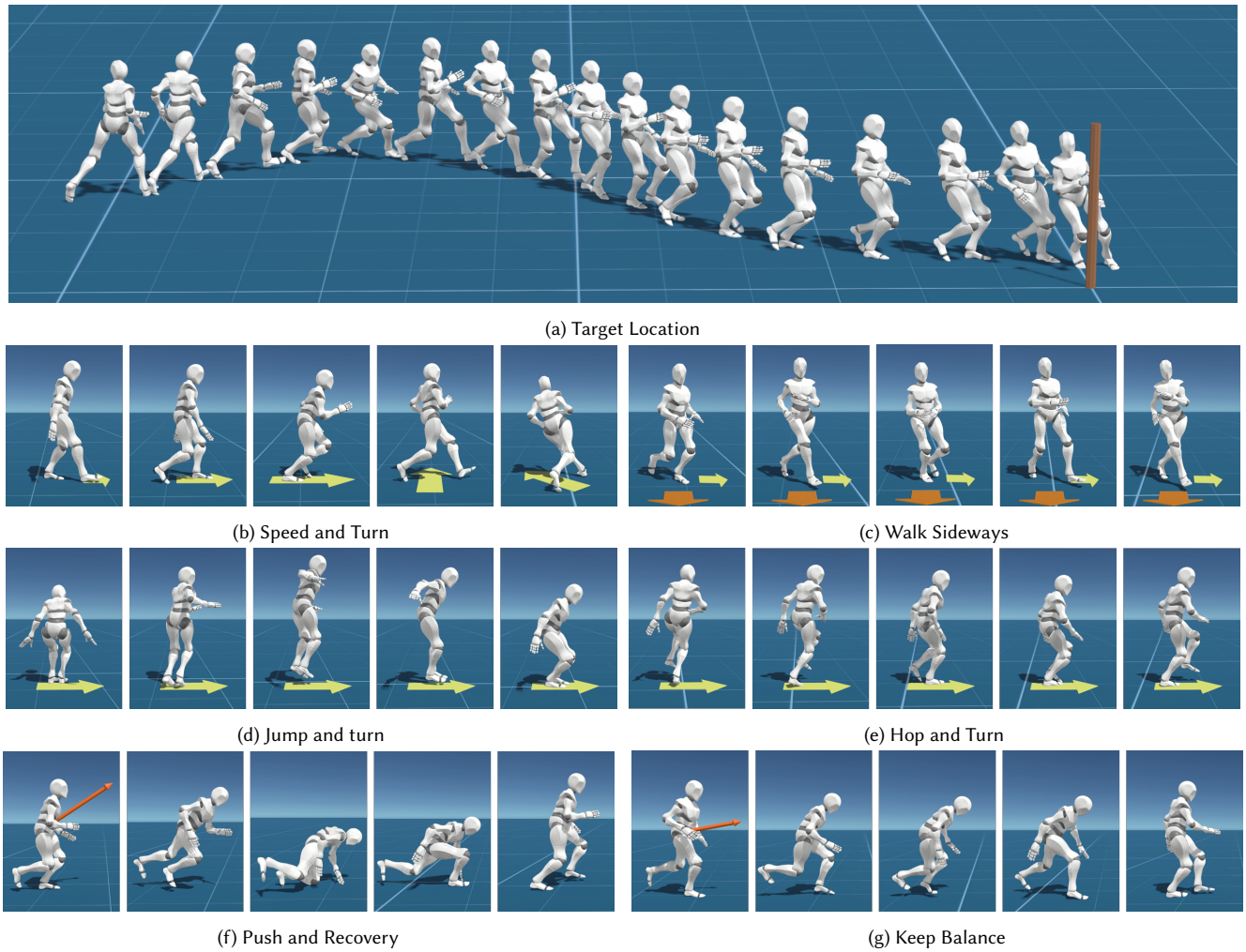


Fig. 10. Simulated character performing tasks with a high-level control policy.

et al. 2020]. Our ControlVAE also support training task-specific policies in this way. As a comparison with our model-based learning paradigm, we train the same heading control task using the PPO algorithm [Schulman et al. 2017], which is often implemented as a model-free reinforcement learning approach and is widely used in physics-based character animation.

More specifically, the reinforcement learning algorithm optimizes a control policy π by maximizing the expected return over all possible simulation trajectories induced by π. To train a task-specific control policy $\pi(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{g})$, we generate these trajectories by sampling skill variables $\boldsymbol{z} \sim \pi(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{g})$, converting $\boldsymbol{z}$ into an action $\boldsymbol{a}$ using the skill-conditioned policy $\pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})$ learned by ControlVAE, and then executing $\boldsymbol{a}$ in the simulation. The reward function is simply defined $r_t = e^{-\mathcal{L}_g}$, where $\mathcal{L}_g$ is the task-specific loss function of the heading control task in Equation (25). Note that we do not include the falling penalty $\mathcal{L}_{\text{fall}}$ in the reward, but instead early-terminate the trajectory when falls are detected. The control policy $\pi(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{g})$ is modeled the same as the one we used in the model-based learning.

We train the policy $\pi(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{g})$ using our own implementation of the PPO algorithm, which can achieve comparable performance on simple tracking tasks with DeepMimic [Peng et al. 2018]. Figure 12 shows the learning curve of this training, as well as the learning curve of the model-based learning on the same task. In general, the policy trained using model-based learning can achieve good performance quickly, while PPO implementation takes more time and transitions to reach a comparable level of performance. It should be noted that in Figure 12, both the real and synthetic transitions are counted. The model-based learning process mainly consumes the synthetic transitions, which can be computed much more efficiently than the simulation due to the approximation nature of the world model and the help of modern GPU hardware. Our model-based learning approach can also achieve better performance with the same numbers of transitions due to the advantage of direct gradient delivery. In practice, the model-based learning can finish training with good results in half an hour, while the reinforcement learning will take a much longer time to obtain a quality control policy.

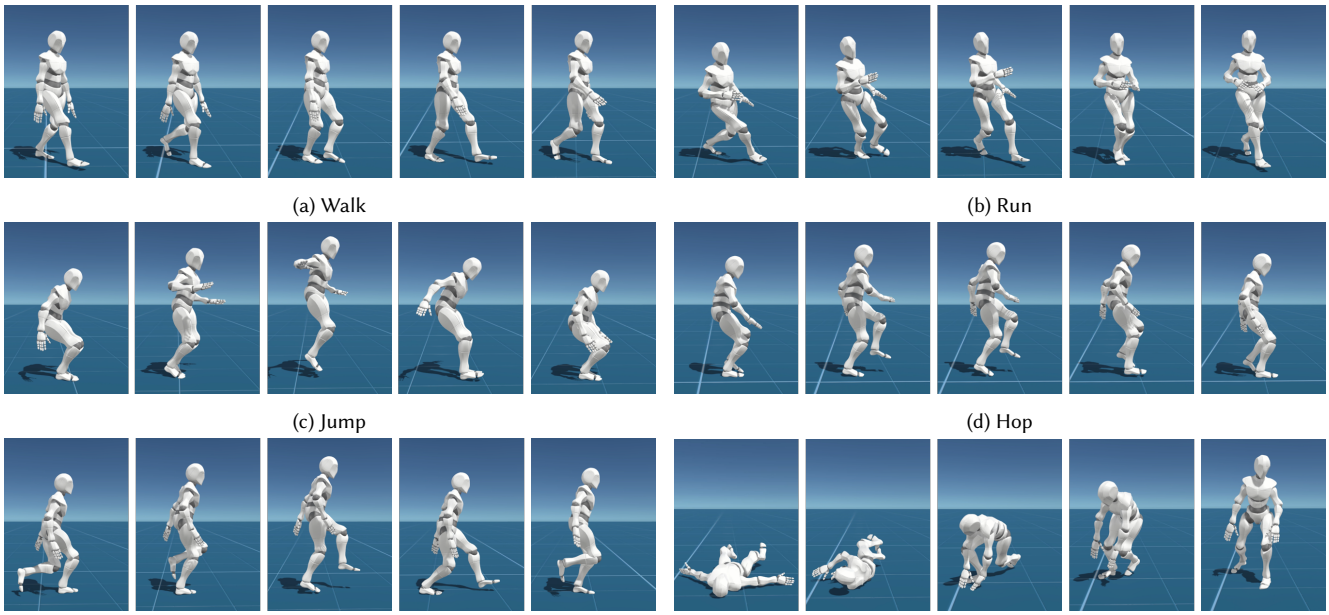


Fig. 11. Simulated character performing different skills in the skill control task

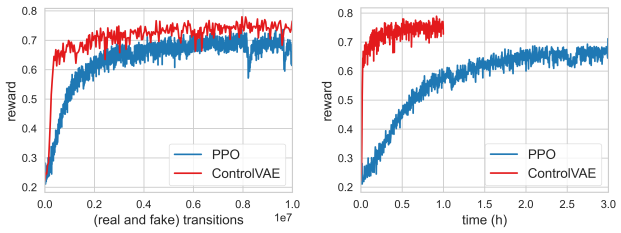
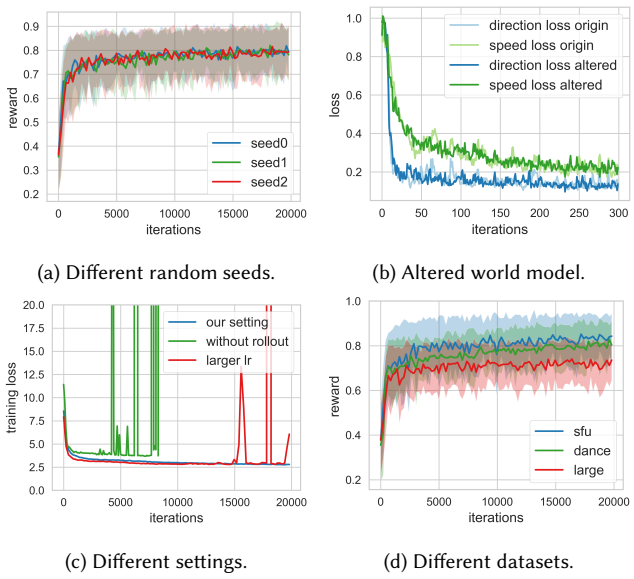


Fig. 12. Training curves of the heading control policies using model-based learning and reinforcement learning. Left: training progresses with respect to the number of transitions. Right: training progresses with respect to time.



(a) Different random seeds. (b) Altered world model.

(c) Different settings. (d) Different datasets.

Fig. 13. (a) Training ControlVAE with different random seeds. (b) Training heading control with the original/altered world model. We train ControlVAE and the original world model with seed 0. The altered world model is trained with seed 1. (c) Training curves with different settings: using a larger learning rate and training the world model without synthetic rollouts. (d) Training ControlVAE on different datasets.

5.7 Robustness, Scalability, and Generalization

Random seed. In Figure 13(a) we show the results of training three ControlVAEs using different random seeds. It can be seen that our method performs consistently in this test. Another interesting fact is that the world models trained with different random seeds can replace each other in training the downstream locomotion tasks. To show this, we train two heading control policies using the same ControlVAE skill policy but with different world models trained with different random seeds. As shown in Figure 13(b), the performances of the results are very close.

Training settings. Figure 13(c) shows the effect of different training settings. We first test to train the world model without the synthetic rollouts, *i.e.* $T_w = 1$. The result shows that the training process is fairly unstable and blows up early. In the second test, we train ControlVAE with a larger learning rate ($2e-5$). The training process converges faster than our default settings but can become unstable as the training progresses. However, even though the training loss may occasionally increase quickly, we can early-terminate

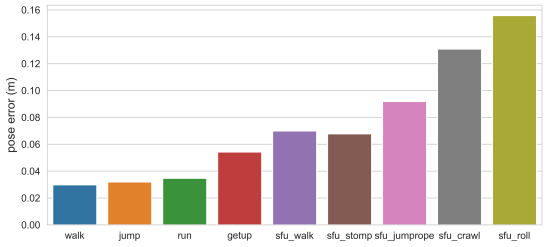


Fig. 14. Tracking error of different motions. The motion with the prefix *sfu* are selected form the SFU motion dataset [2011], which are not used in training the ControlVAE.

the learning during the stable region, and the learned ControlVAE can still accomplish the downstream tasks.

Coverage. To test how well the learned skill embeddings cover the training skills, we let ControlVAE track all the motion sequences in the training dataset and check the accuracy of the reconstruction. We divide the long motion sequences into 2-second clips, and calculate the average tracking error between the relative position of each body to the root and that in the reference. As shown in Figure 14, the learned ControlVAE can reconstruct the training motions accurately, which indicates that all the training skills are embedded in the latent space. We further test the ability of the learned ControlVAE in reconstructing unseen motions. Five motion clips from the SFU dataset [2011] are selected in this test, as shown in Figure 14, which are not used in the training. As indicated in Figure 14, ControlVAE can track the unseen motions that are similar to the existing skills in the training dataset accurately, such as stylized walking and stomping, albeit with some motion details missing. For the motions that are significantly different, such as rolling, the character only struggles on the ground, causing a large tracking error.

Dataset. We test our ControlVAE on two other datasets. One is a selected subset of the SFU mocap dataset [2011], containing approximately 7-minute diverse locomotion clips. The other is a 4-minute dance clip selected from the LaFAN dataset [Harvey et al. 2020]. These tests have similar training processes, and random sampling in the latent spaces creates diverse locomotion behaviors and dances.

We further conduct an experiment on a large-scale dataset composed of 3.3 hours (after the mirror augmentation) of motions selected from the LaFAN [Harvey et al. 2020] dataset, which covers most of the motion categories except for those interact with external objects or uneven terrains. The result in Figure 13(d) shows that the training process finishes in roughly the same time as that on the small datasets but converges to a lower reward. The learned skill embeddings can still recover an input motion with a larger visual discrepancy, but the performance on the downstream tasks are significance degenerated.

6 DISCUSSION

In this paper, we present ControlVAE, a model-based framework for learning generative motion control policies for physics-based characters. We show that state-conditional VAEs can be efficiently trained using a model-based method, resulting in a rich and flexible latent space that captures a diverse set of skills and a skill-conditioned control policy that effectively converts each sample in the latent space into realistic behaviors in physics-based simulation. We show that taking advantage of these generative control policies, we can generate various motion skills simply by sampling from the latent space. Task-specific control policies can be further trained to operate in this latent space, allowing a variety of high-level tasks to be accomplished using realistic motions.

The key observation of this work is that by learning a differentiable world model, we can effectively bridge the gap between the learning of control policies and the losses defined on the simulated motion. Furthermore, the learned world model provides an effective differentiable approximation of the real simulation, which allows high-level policy to be trained efficiently using model-based learning. We believe that our results open the door to many interesting topics, such as model-based learning of controllers using other generative models like GAN and normalizing flows. In addition, combining model-based learning with other successful motion synthesis algorithms could be a potential way to realized physics-based motion synthesis [Starke et al. 2021], style transfer [Aberman et al. 2020], and motion in-painting [Harvey et al. 2020].

For the future work, we wish to explore the possibility of adapting the learned world model to environmental changes, such as dealing with additional objects and characters, which potentially allows us to extend a learned skill space to unseen environments. Currently, our model-based learning schemes can not handle such tasks, for which the model-free reinforcement learning methods are still needed. In addition, similar to other data-driven method, the performance of our system is still bounded by the training dataset. For example, we have encountered problems in learning a high-level policy to control the character to travel while facing towards certain directions because the corresponding motions are sparse in the dataset. Our ControlVAE also performs less efficiently in learning large-scale motion dataset with hours of motions, potentially due to the limited capacity of the network architecture. Learning these sparse skills more effectively, as well as scaling up to large-scale datasets, will be a very interesting direction for future exploration.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments. This work was supported in part by NSFC Projects of International Cooperation and Exchanges (62161146002).

REFERENCES

Kfir Aberman, Yijia Weng, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. 2020. Unpaired Motion Style Transfer from Video to Animation. *ACM Transactions on Graphics* 39, 4 (July 2020), 64:64:1–64:64:12.

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: Data-Driven Responsive Control of Physics-Based Characters. *ACM Transactions on Graphics* 38, 6 (Nov. 2019), 206:1–206:11.

Silvia Chiappa, Sébastien Racanière, Daan Wierstra, and Shakir Mohamed. 2017. Recurrent Environment Simulators. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, Toulon, France.

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized Biped Walking Control. *ACM Transactions on Graphics* 29, 4 (July 2010), 130:1–130:9.

Marc Peter Deisenroth and Carl Edward Rasmussen. 2011. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11)*. Omnipress, Madison, WI, USA, 465–472.

Haegwang Eom, Daseong Han, Joseph S. Shin, and Junyong Noh. 2019. Model Predictive Control with a Visuomotor System for Physics-based Character Animation. *ACM Transactions on Graphics* 39, 1 (Oct. 2019), 3:1–3:11.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. 2016. Sequential Neural Models with Stochastic Layers. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 2207–2215.

Levi Fussell, Kevin Bergamin, and Daniel Holden. 2021. SuperTrack: Motion Tracking for Physically Simulated Characters Using Supervised Learning. *ACM Transactions on Graphics* 40, 6 (Dec. 2021), 197:1–197:13.

Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. 1998. NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. Association for Computing Machinery, New York, NY, USA, 9–20.

David Ha and Jürgen Schmidhuber. 2018. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 2455–2467.

Perttu Hämäläinen, Joose Rajamäki, and C. Karen Liu. 2015. Online Control of Simulated Humanoids Using Particle Belief Propagation. *ACM Transactions on Graphics* 34, 4 (July 2015), 81:1–81:13.

Perttu Hamalainen, Juuso Toikka, Amin Babadi, and Karen Liu. 2020. Visualizing Movement Control Optimization Landscapes. *IEEE Transactions on Visualization and Computer Graphics* (2020), 1–1.

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust Motion In-Betweening. *ACM Trans. Graph.* 39, 4, Article 60 (Jul 2020), 12 pages.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. 2015. Learning Continuous Control Policies by Stochastic Value Gradients. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc.

Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. 2020. MoGlow: Probabilistic and Controllable Motion Synthesis Using Normalising Flows. *ACM Trans. Graph.* 39, 6, Article 236 (nov 2020), 14 pages.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*.

Jonathan Ho and Stefano Ermon. 2016. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc.

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating Human Athletics. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 71–78.

Daniel Holden, Jun Saito, and Taku Komura. 2016. A Deep Learning Framework for Character Motion Synthesis and Editing. *ACM Transactions on Graphics* 35, 4 (July 2016), 138:1–138:11.

Seokpyo Hong, Daseong Han, Kyungmin Cho, Joseph S. Shin, and Junyong Noh. 2019. Physics-Based Full-Body Soccer Motion Control for Dribbling and Shooting. *ACM Transactions on Graphics* 38, 4 (July 2019), 74:1–74:12.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. 2021. When to Trust Your Model: Model-Based Policy Optimization. https://doi.org/10.48550/arXiv.1906.08253 arXiv:1906.08253 [cs, stat]

Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014*. Banff, AB, Canada.

Kyungho Lee, Sehee Min, Sunmin Lee, and Jehee Lee. 2021. Learning Time-Critical Responses for Interactive Character Control. *ACM Transactions on Graphics* 40, 4 (July 2021), 147:1–147:11.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-Driven Biped Control. *ACM Transactions on Graphics* 29, 4 (July 2010), 129:1–129:8.

Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. 2012. Continuous Character Control with Low-Dimensional Embeddings. *ACM Trans. Graph.* 31, 4, Article 28 (Jul 2012), 10 pages.

Peizhuo Li, Kfir Aberman, Zihan Zhang, Rana Hanocka, and Olga Sorkine-Hornung. 2022. GANimator: Neural Motion Synthesis from a Single Sequence. *ACM Trans. Graph.* 41, 4, Article 138 (jul 2022), 12 pages.

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character Controllers Using Motion VAEs. *ACM Transactions on Graphics* 39, 4 (July 2020), 40:40:1–40:40:12.

Libin Liu and Jessica Hodgins. 2017. Learning to Schedule Control Fragments for Physics-Based Characters Using Deep Q-Learning. *ACM Transactions on Graphics* 36, 4 (June 2017), 42a:1.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the Variance of the Adaptive Learning Rate and Beyond. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Libin Liu, Michiel Van De Panne, and Kangkang Yin. 2016. Guided Learning of Control Graphs for Physics-Based Characters. *ACM Transactions on Graphics* 35, 3 (May 2016), 29:1–29:14.

Libin Liu, KangKang Yin, Michiel van de Panne, and Baining Guo. 2012. Terrain Runner: Control, Parameterization, Composition, and Planning for Highly Dynamic Motions. *ACM Transactions on Graphics* 31, 6 (Nov. 2012), 154:1–154:10.

Libin Liu, KangKang Yin, Bin Wang, and Baining Guo. 2013. Simulation and Control of Skeleton-Driven Soft Body Characters. *ACM Transactions on Graphics* 32, 6 (Nov. 2013), 1–8.

Ying-Sheng Luo, Jonathan Hans Soeseno, Trista Pei-Chun Chen, and Wei-Chao Chen. 2020. CARL: Controllable Agent with Reinforcement Learning for Quadruped Locomotion. *ACM Transactions on Graphics* 39, 4 (July 2020), 38:38:1–38:38:10.

Adriano Macchietto, Victor Zordan, and Christian R. Shelton. 2009. Momentum Control for Balance. In *ACM SIGGRAPH 2009 Papers (SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, 1–8.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least Squares Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. 2018. Neural Probabilistic Motor Primitives for Humanoid Control. In *International Conference on Learning Representations*.

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. 2020. Catch & Carry: Reusable Neural Controllers for Vision-Guided Whole-Body Tasks. *ACM Transactions on Graphics* 39, 4 (July 2020), 39:39:1–39:39:12.

Jianyuan Min and Jinxiang Chai. 2012. Motion Graphs++: A Compact Generative Model for Semantic Motion Analysis and Synthesis. *ACM Trans. Graph.* 31, 6, Article 153 (nov 2012), 12 pages.

Miguel Angel Zamora Mora, Momchil P Peychev, Sehoon Ha, Martin Vechev, and Stelian Coros. 2021. PODS: Policy Optimization via Differentiable Simulation. In *International Conference on Machine Learning*. PMLR, 7805–7817.

Igor Mordatch, Martin de Lasa, and Aaron Hertzmann. 2010. Robust Physics-Based Locomotion Using Low-Dimensional Planning. *ACM Transactions on Graphics* 29, 4 (July 2010), 71:1–71:8.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of Complex Behaviors through Contact-Invariant Optimization. *ACM Transactions on Graphics* 31, 4 (July 2012), 43:1–43:8.

Uldarico Muico, Jovan Popović, and Zoran Popović. 2011. Composite Control of Physically Simulated Characters. *ACM Transactions on Graphics* 30, 3 (May 2011), 16:1–16:11.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. 2018. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 7559–7566.

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning Predict-and-Simulate Policies from Unorganized Human Motion Data. *ACM Trans. Graph.* 38, 6, Article 205 (nov 2019), 11 pages.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics* 37, 4 (July 2018), 143:1–143:14.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Number 331. Curran Associates Inc., Red Hook, NY, USA, 3686–3697.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. ASE: Large-Scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters. *ACM Trans. Graph.* 41, 4, Article 94 (jul 2022), 17 pages.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *ACM Transactions on Graphics* 40, 4 (July 2021), 144:1–144:20.

Mathis Petrovich, Michael J. Black, and Gül Varol. 2021. Action-Conditioned 3D Human Motion Synthesis with Transformer VAE. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE,

10965–10975.
Jürgen Schmidhuber. 1990. *Making the World Differentiable: On Using Self-Supervised Fully Recurrent Neural Networschmidhuber1990makingks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments.* Technical Report.
John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347
Sebastian Starke, Yiwei Zhao, Fabio Zinno, and Taku Komura. 2021. Neural Animation Layering for Synthesizing Martial Arts Movements. *ACM Trans. Graph.* 40, 4, Article 92 (Jul 2021), 16 pages.
Jie Tan, Yuting Gu, C. Karen Liu, and Greg Turk. 2014. Learning Bicycle Stunts. *ACM Transactions on Graphics* 33, 4 (July 2014), 50:1–50:12.
Jie Tan, Karen Liu, and Greg Turk. 2011. Stable Proportional-Derivative Controllers. *IEEE Computer Graphics and Applications* 31, 4 (July 2011), 34–44.
Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A Physics Engine for Model-Based Control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5026–5033.
Simon Fraser University and National University of Singapore. 2011. SFU Motion Capture Database. https://mocap.cs.sfu.ca.
Jack M. Wang, David J. Fleet, and Aaron Hertzmann. 2008. Gaussian Process Dynamical Models for Human Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 2 (Feb. 2008), 283–298.
Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C Karen Liu. 2021. Fast and feature-complete differentiable physics engine for articulated rigid bodies with contact constraints. In *Robotics: Science and Systems*.
Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Transactions on Graphics* 39, 4 (July 2020), 33:33:1–33:33:12.
Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-Based Character Controllers Using Conditional VAEs. *ACM Trans. Graph.* 41, 4, Article 96 (Jul 2022), 12 pages.
Jungdam Won and Jehee Lee. 2019. Learning Body Shape Variation in Physics-Based Characters. *ACM Trans. Graph.* 38, 6, Article 207 (nov 2019), 12 pages.
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* (Virtual Event, Canada) *(SCA '20)*. Eurographics Association, Goslar, DEU, Article 20, 12 pages.
KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Transactions on Graphics* 26, 3 (July 2007), 105–es.
Zhiqi Yin, Zeshi Yang, Michiel van de Panne, and Kangkang Yin. 2021. Discovering Diverse Athletic Jumping Strategies. *ACM Trans. Graph.* 40, 4, Article 91 (Jul 2021), 17 pages.
Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning Symmetric and Low-energy Locomotion. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 1–12. arXiv:1801.08093
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/CVF Conference on Computer Vision and Pattern Recognition*. 5745–5753.