GPC: Large-Scale Generative Pretraining for Transferable Motor Control

YI SHI, Simon Fraser University, Canada and NVIDIA, USA
YIFENG JIANG, NVIDIA, USA
CHEN TESSLER, NVIDIA, USA
XUE BIN PENG, Simon Fraser University, Canada and NVIDIA, Canada

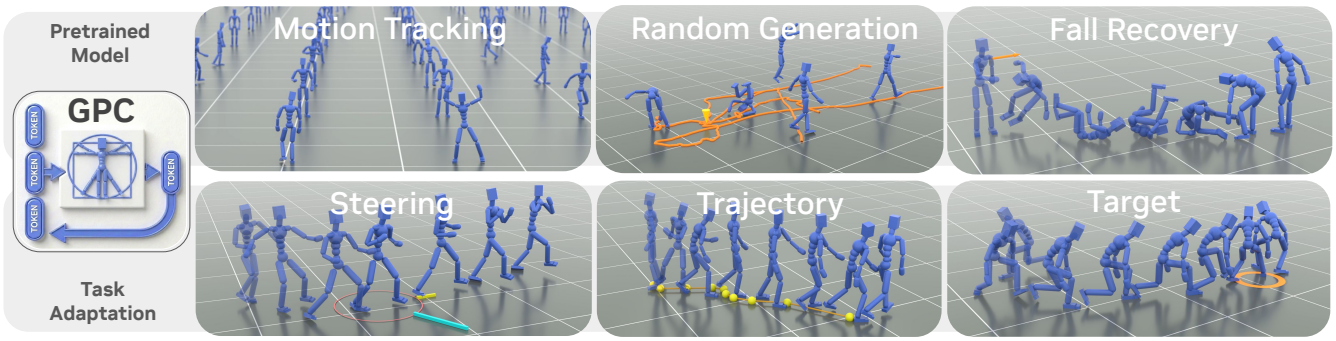


Fig. 1. We present Generative Pretrained Controllers (GPC), which models a wide range of motor skills through next-token prediction with a learned discrete representation of behaviors. GPC can be trained on large-scale motion datasets, and then adapted to downstream tasks, while retaining the naturalistic behaviors acquired by the pretrained model.

Developing controllers capable of completing a wide range of tasks in a natural and life-like manner is a key challenge in enabling practical applications of physics-based character animation. In this work, we introduce Generative Pretrained Controllers (GPC), which leverage tokenization and next-token modeling to create general-purpose, reusable *generative* controllers from large-scale motion datasets. Our framework utilizes end-to-end reinforcement learning to jointly optimize a "motion vocabulary", modeled via Finite Scalar Quantization (FSQ), along with a corresponding control policy that can map the discrete codes to physics-based controls. After the "codebook" has been learned, the underlying structure of this large vocabulary is modeled by training a GPT-style autoregressive transformer, leading to a powerful generative controller that generates controls for a physically simulated character by performing next-token prediction. Once the generative controller has been trained, we propose a suite of adaptation techniques for finetuning the controller for new downstream tasks. Our proposed framework greatly simplifies the training process compared to previous tokenized methods, and achieves a 99.98% success rate in reproducing a vast corpus of motion clips. The generative controller exhibits a variety of natural emergent behaviors, such as responsive behaviors to perturbations and recovery behaviors after falling. This results in highly robust general purpose controllers for a variety of downstream applications.

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Procedural animation**.

ACM Reference Format:
Yi Shi, Yifeng Jiang, Chen Tessler, and Xue Bin Peng. 2026. GPC: Large-Scale Generative Pretraining for Transferable Motor Control. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA*. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3799902.3811038

1 Introduction

Training physics-based controllers that can endow virtual characters with versatile, human-like behaviors is essential for applications such as film, video games, and extended reality (XR), as it enables characters to interact realistically with dynamic environments and generalize beyond manually authored animations. Achieving this goal requires controllers capable of capturing a broad spectrum of human motor skills and providing a convenient mechanism to reuse these skills across different tasks. Prior works have leveraged generative models, such as variational autoencoders (VAEs) and generative adversarial networks (GANs) [Goodfellow et al. 2014; Kingma and Welling 2022], to learn latent representations of a wide range of motor skills from diverse motion datasets [Luo et al. 2023; Peng et al. 2022; Tessler et al. 2024]. Once such a generative model is trained, a high-level controller can be developed to select appropriate skills from the model's latent space to perform various downstream tasks. Constraining the high-level controller to operate within this structured latent space, instead of directly issuing

Authors' Contact Information: Yi Shi, Simon Fraser University, Burnaby, British Columbia, Canada and NVIDIA, Santa Clara, California, USA, yish@nvidia.com; Yifeng Jiang, NVIDIA, Santa Clara, California, USA, yifengj@nvidia.com; Chen Tessler, NVIDIA, Santa Clara, California, USA, ctessler@nvidia.com; Xue Bin Peng, Simon Fraser University, Burnaby, British Columbia, Canada and NVIDIA, Vancouver, British Columbia, Canada, japeng@nvidia.com.

SIGGRAPH Conference Papers '26, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/2026/07
https://doi.org/10.1145/3799902.3811038

low-level joint commands, provides an inductive bias toward behaviors that are more consistent with those in the dataset. However, most existing generative controllers adopt continuous latent spaces, which are prone to mode collapse and unnatural behaviors caused by drift and *gaps* in the latent manifold [Dou et al. 2023; Peng et al. 2022; Won et al. 2022].

To address these challenges, recent work has explored discrete latent models based on VQ-VAEs [van den Oord et al. 2018]. By mapping continuous data into discrete codes, these models sidestep the difficulty of modeling complex continuous distributions by encoding continuous features into discrete embeddings [Bae et al. 2025a,b; Yao et al. 2024; Zhu et al. 2023]. However, VQ-VAE–based tokenization can be prone to degeneracies, such as low code usage, which often requires complex training heuristics to mitigate [van den Oord et al. 2018]. These degeneracies can compromise the expressiveness of the learned representation, which can in turn increase the difficulty of capturing diverse behaviors from large motion datasets [Yao et al. 2024]. As a result, existing VQ-VAE–based tracking controllers have been trained primarily on datasets of modest scale, ranging from small curated datasets to subsets of larger datasets, such as AMASS [Mahmood et al. 2019], which contain approximately 20 hours of motion data [Yao et al. 2024].

In this work, we introduce generative pretrained controllers (GPC), which learn a discretized latent representation of motor skills using Finite Scalar Quantization (FSQ) [Mentzer et al. 2023]. Unlike VQ-VAE–based methods, FSQ does not require an explicit codebook, which substantially simplifies training by eliminating the need for codebook embedding updates, auxiliary losses, and ad hoc heuristics such as dead-code re-initialization [van den Oord et al. 2018]. By training with a simple motion-tracking objective, our model is able to learn a discrete latent space that captures a wide range of behaviors, including highly dynamic skills such as vaulting, cartwheels, and flips. Once the discrete skill representation has been constructed, we train a GPT-style autoregressive transformer to model the distribution of skill tokens [Radford et al. 2018, 2019], which serves as the generative controller over skills. The learned generative controller produces natural emergent behaviors, such as human-like responses to perturbations and recovery strategies. To adapt the pretrained generative controller to new tasks, we leverage parameter-efficient fine-tuning (PEFT), which inserts lightweight modulation layers into the pretrained generative controller. The PEFT layers can be updated with additional task objectives, while keeping the underlying generative model fixed. This enables GPC to leverage previously learned skills to perform new tasks while preserving the natural, life-like behaviors encoded in the pretrained model.

The core contribution of this work is a generative pretrained controller that uses a discrete latent space modeled with Finite Scalar Quantization (FSQ). The controller is trained end-to-end with reinforcement learning on a large-scale human motion dataset containing over 600 hours of diverse behaviors. The proposed framework supports parameter-efficient fine-tuning (PEFT), enabling efficient adaptation of the generative controller to new downstream tasks without retraining. The resulting FSQ-based tracking controller produces natural behaviors and achieves a 99.98% tracking success rate on a large-scale dataset. Once trained, GPC can be effectively adapted and reused across a diverse set of downstream tasks through parameter-efficient fine-tuning.

2 Related Work

Our work proposes a framework for training and deploying generative controllers for physically simulated characters. The framework is centered on a generative controller that models a distribution over motor skills learned from large motion datasets. In the following, we review prior work most closely related to the key components.

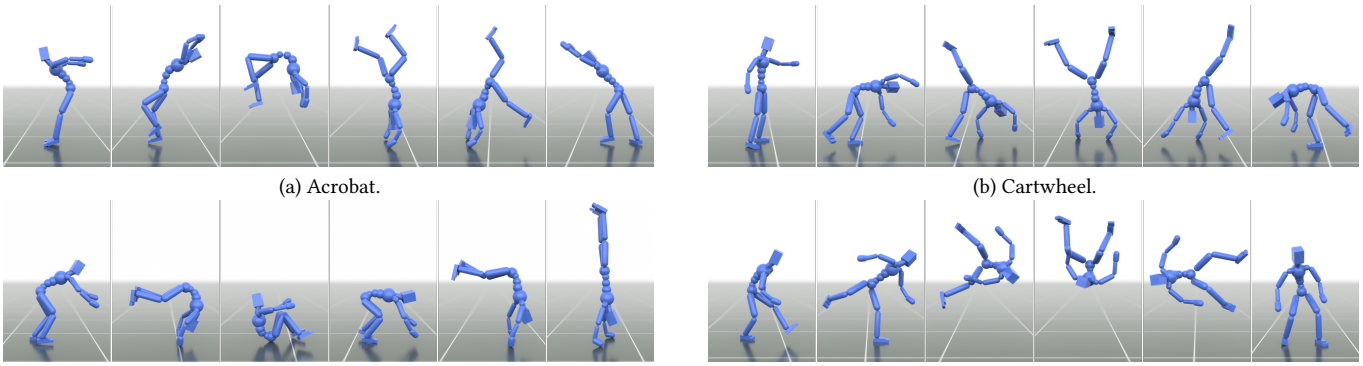
2.1 Physics-Based Character Controller

Physics-based character animation leverages physics simulations to produce physically plausible motions. The motions of the physically simulated characters are driven by *controllers*, which specify control signals for each joint in the character's body. Early methods often rely on manually-crafted controller strategies that leverage human knowledge of a particular type of motor skill, such as walking, running, or balance recovery [Coros et al. 2008; da Silva et al. 2008; Hodgins et al. 1995; Ye and Liu 2010; Yin et al. 2007]. While effective for narrowly defined behaviors, these methods are often difficult to scale to more complex motions and require substantial manual effort to design for new skills or environments.

To reduce this burden, subsequent work has explored example-guided reinforcement learning, in which controllers are trained using motion demonstrations, typically obtained from motion capture as dense reward signals that guide policy learning and substantially reduce the need for manually specified reward functions [Chentanez et al. 2018; Peng et al. 2018, 2017]. Subsequent studies further improve the generality of imitation-learning–based motion tracking by adapting controllers to different body shapes [Won and Lee 2019], reducing reliance on carefully tuned objective hyperparameters [Zhang et al. 2025], and scaling to large motion datasets with lengths of dozens of hours [Luo et al. 2023].

2.2 Generative Controller

Human motions are inherently multi-modal, with a diverse range of possible behaviors that a human may perform in a given scenario. To cope with the complexity of these behavioral distributions, some prior work adopts a hierarchical framework that separates motion generation from physical execution [Tevet et al. 2024; Wang et al. 2024; Xu et al. 2025; Ye et al. 2023]. In these approaches, kinematic generative models are first trained to produce diverse motion trajectories, and then the generated trajectories are tracked by physics-based motion tracking controllers during simulation. While this decomposition allows the generative model and the physics-based controller to focus on complementary objectives, it introduces a mismatch between the two modules, undermining physical performance. Some methods instead explicitly learn skill representations that can be reused and composed for downstream tasks. For example, ASE learns reusable skill embeddings through adversarial imitation learning within a GAN framework [Peng et al. 2022], allowing high-level policies to produce complex behaviors by composing learned primitives. Subsequent work further enhances the structure and interpretability of these representations by incorporating semantic



(a) Acrobat. (b) Cartwheel.

(c) Handstand. (d) Sideflip.

Fig. 2. Keyframes of the results produced by the FSQ tracking controller on agile motions from Bones.

motion labels [Dou et al. 2023], enabling more controllable and steerable behaviors in downstream tasks [Tessler et al. 2023]. Tessler et al. [2024] learn latent skill representations using a masked-inpainting training procedure, which enables the model to be reused for new downstream tasks via flexible kinematic constraints. Training generative controllers end-to-end on large and diverse motion datasets, however, remains challenging. To address this, several methods adopt a two-stage training paradigm, in which an expert motion-tracking controller is first trained, and its behavior is then distilled into a generative model [Merel et al. 2019; Won et al. 2022]. In this line of research, recent studies scale the expert training stage to larger datasets up to 40 hours of motion and employ more expressive generative models, such as conditional VAEs and diffusion models, to capture a broader range of behaviors [Huang et al. 2025; Luo et al. 2024; Truong et al. 2024]. Despite their successes, these methods rely on continuous latent spaces that are vulnerable to mode collapse and unstable towards off-manifold states, often leading to failures when producing highly dynamic motions.

2.3 Skill Quantization

Discrete latent-variable models mitigate these issues by imposing an explicit structure on the representation space, helping constrain generation to remain closer to the data manifold. As a result, discrete representations have been widely adopted in image generation [Esser et al. 2021; van den Oord et al. 2018], speech modeling [Baevski et al. 2020], and motion synthesis [Guo et al. 2024; Jiang et al. 2023; Starke et al. 2024]. In physics-based character control, Zhu et al. [2023] trains a generative controller that samples from a discrete latent space constructed by a VQ-VAE tracking controller. Yao et al. [2024] learn a world model using a residual VQ-VAE and extend it to text-conditioned motion generation. Bae et al. [2025a] introduces body–part–specific quantization to achieve fine-grained control over different body parts of a character. Many prior methods rely on VQ-VAE–based discretization, which requires careful tuning and auxiliary heuristics. In contrast, we adopt Finite Scalar Quantization (FSQ) [Mentzer et al. 2023], which removes the need for a learned codebook and avoids common VQ-VAE failure modes, enabling more stable training of discrete motion representations on large-scale datasets. We then model the relationship between sequences of discrete latent codes using an autoregressive transformer, which captures the temporal structure of diverse behaviors in large motion datasets.

2.4 Parameter-Efficient Fine-Tuning

As the size of training datasets and models has drastically increased in recent years, efficient methods for fine-tuning large models on new tasks have become vital to the practical application of large pretrained models. PEFT methods freeze the pretrained backbone and introduce a small number of task-specific parameters to enable efficient adaptation. One class of PEFT approaches augments frozen models with auxiliary task-specific networks, such as adapters [Houlsby et al. 2019; Pfeiffer et al. 2021]. In diffusion models, ControlNet follows this paradigm by attaching trainable control branches to a frozen generator to introduce new conditioning signals, albeit with a relatively high parameter cost [Zhang et al. 2023b]. A complementary class of PEFT methods focuses on modulating existing weights rather than introducing separate networks. LoRA injects trainable low-rank updates directly into pretrained weight matrices, significantly reducing memory and computation overhead [Hu et al. 2021]. Its successors improve on LoRA in terms of efficiency [Dettmers et al. 2023], and automatic rank selection [Zhang et al. 2023a]. DoRA further refines this formulation by decomposing weight updates into magnitude and directional components, improving optimization stability and expressiveness [Liu et al. 2024]. Our approach bridges these paradigms by introducing task-specific conditioning via a learned task token while employing low-rank adaptation to modulate a frozen generative prior, enabling efficient downstream adaptation without large auxiliary networks.

3 Background

3.1 Reinforcement Learning

We leverage Reinforcement learning (RL) in training tracking controllers and controllers for downstream tasks. RL studies how an agent can learn a policy for sequential decision-making, commonly modeled as a Markov Decision Process (MDP) [Sutton et al. 1998]. At each time step t, the agent observes a state $\mathbf{s}_t$ and selects an action $a_t \sim \pi(a_t|s_t)$ according to a policy π. The environment then

transitions to the next state $\mathbf{s}_{t+1}$ according to the transition dynamics $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$, and the agent receives a scalar reward $r_t = r(s_t, a_t, s_{t+1})$. The agent's objective is then to learn an optimal policy that maximizes the expected discounted return:

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

where $\gamma \in [0, 1)$ is a discount factor that balances short-term and long-term rewards.

3.2 Vector Quantization

Vector-Quantized Variational Autoencoder (VQ-VAE) introduces discrete latent representations by mapping continuous embeddings from an encoder to the nearest entry in a discrete codebook. The encoder maps an input $\mathbf{x}$ to a latent $\mathbf{z} = \mathcal{E}(\mathbf{x})$, which is then quantized via nearest-neighbor lookup in a learned codebook $C = \{\mathbf{e}_k\}_{k=1}^K$:

$$k^* = \arg\min_k \|\mathbf{z} - \mathbf{e}_k\|_2. \tag{2}$$

The decoder then reconstructs the data by taking the embeddings from the codebook $\hat{\mathbf{x}} = \mathcal{D}(\mathbf{e}_k)$. Training a VQ-VAE requires three loss components: 1) the reconstruction loss that encourages the decoder to reproduce the input from the quantized latent codes, 2) A codebook loss that updates the embedding vectors towards the encoder's outputs, ensuring that the discrete codes represent the data distribution, 3) a commitment loss that penalizes deviations of the encoder outputs from their assigned codebook entries, encouraging stable code usage by penalizing large deviations between the encoder output and its selected codebook vector, which discourages frequent switching between different codes for similar input or same input during successive updates during training. The VQ-VAE training objective is then given by,

$$\mathcal{L}_{VQ} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \|\text{sg}[\mathcal{E}(x)] - \mathbf{e}_{k^*}\|_2^2 + \beta\|\mathbf{z} - \text{sg}[\mathbf{e}_{k^*}]\|_2^2. \tag{3}$$

VQ-VAEs are difficult to train as the discrete codebook updates can suffer from poor utilization and codebook collapse. These issues are typically addressed using heuristics such as exponential moving average (EMA) updates for the codebook, and periodically reinitialization of unused codes to encourage high codebook utilization.

4 Framework Overview

In this work, we present a framework for training generative controllers on large-scale motion datasets. These general-purpose controllers are capable of modeling large repertoires of motor skills for physics-based character animation. As illustrated in Fig. 3, the framework consists of three stages.

In the first stage, Skill Quantization (Section 5), we construct a discrete latent representation of skills using Finite Scalar Quantization (FSQ). The latent codes are optimized directly with end-to-end RL to model a wide range of motor skills from a large motion dataset. This end-to-end RL training helps to ensure that the learned discrete codes correspond to skills that can be faithfully executed by a physically simulated controller. This provides a robust foundation for subsequent generative controller training and downstream task adaptation.

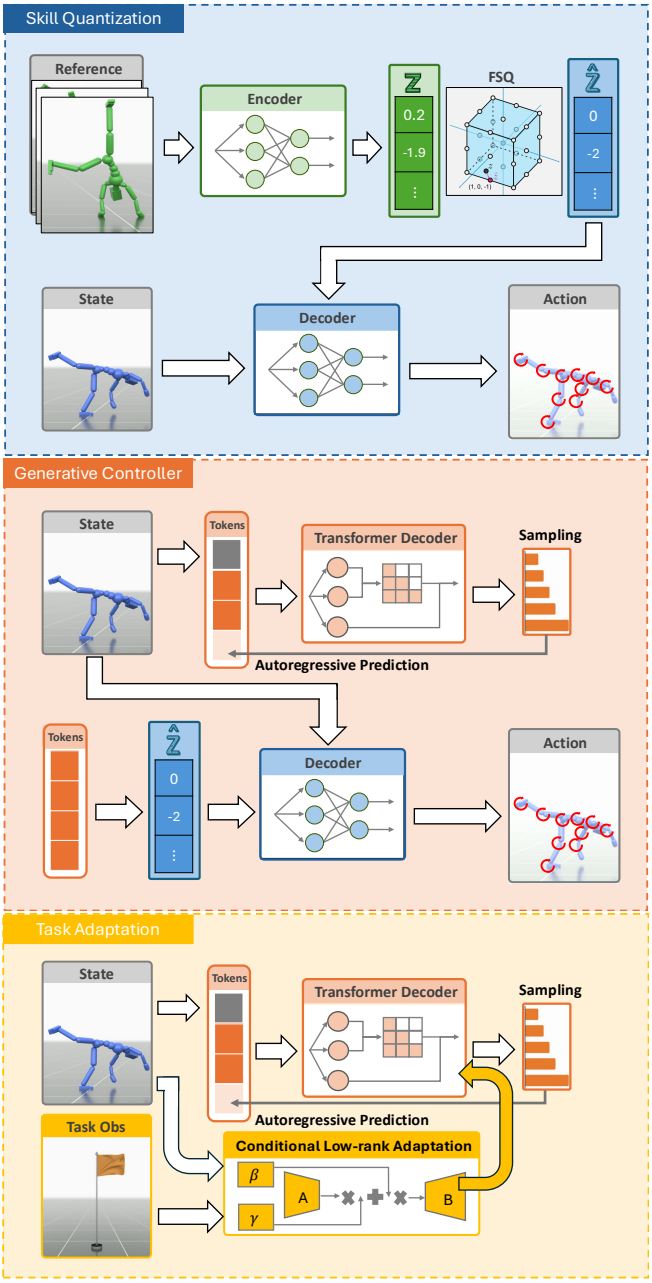
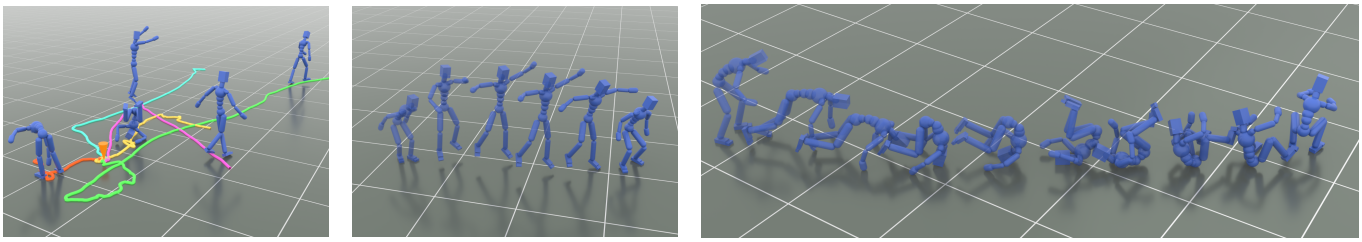


Fig. 3. **Skill quantization**: An FSQ motion tracking controller maps reference motions to discrete latent tokens and decodes them to actions. The FSQ module is trained end-to-end with reinforcement learning using a motion-tracking objective (top). **Generative controller training**: a transformer decoder models the skill distribution via causal self-attention, enabling autoregressive sampling of discrete skill tokens conditioned on the character state $\mathbf{s}$. The generative controller is trained with teacher forcing and cross-entropy loss to predict the tokens produced by FSQ (middle). **Task adaptation**: Lightweight CoLA layers are added to adapt the frozen pre-trained generative controller to complete downstream tasks (bottom). This adaptation is parameter-efficient, adding less than 1% additional parameters to the model.

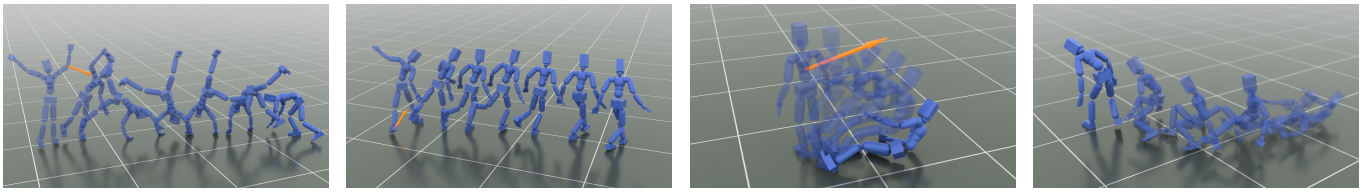


(a) Trajectories from unconditional sampling with GPC.

(b) Jumping skill synthesized by the generative controller.

(c) A motion sequence featuring a forward leap, front roll, and transition to a squat.

Fig. 4. Unconditional sampling of the generative controller. GPC produces a wide array of highly dynamic skills, such as jumping, leaping, and rolling.



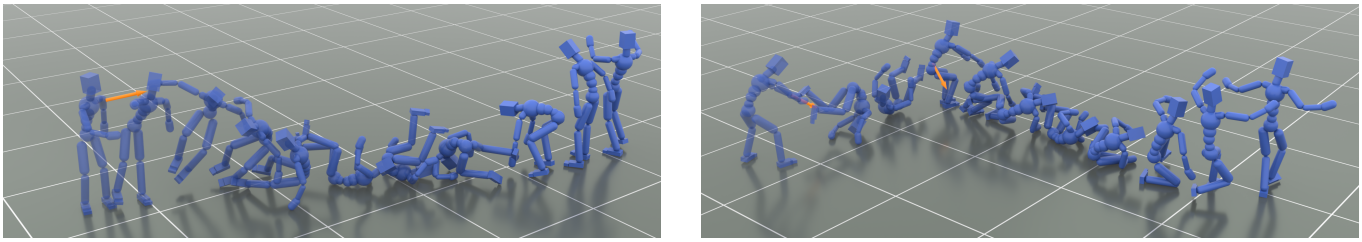
(a) A perturbation force on the arm triggers a cartwheel-like recovery.

(b) After a perturbation force is applied to the character's leg, it automatically adjusts its step to regain balance.

(c) A strong external force applied to the spine destabilizes the character and causes it to fall in a natural way, and is ready to recover.

(d) The character executes a get-up skill and recovers to a standing pose from a fallen state shown in Figure 5(c).

Fig. 5. GPC produces robust and natural recovery behaviors when subjected to external perturbations.



(a) When a character falls down due to external forces, it automatically transitions to rolling skills to recover and get back up.

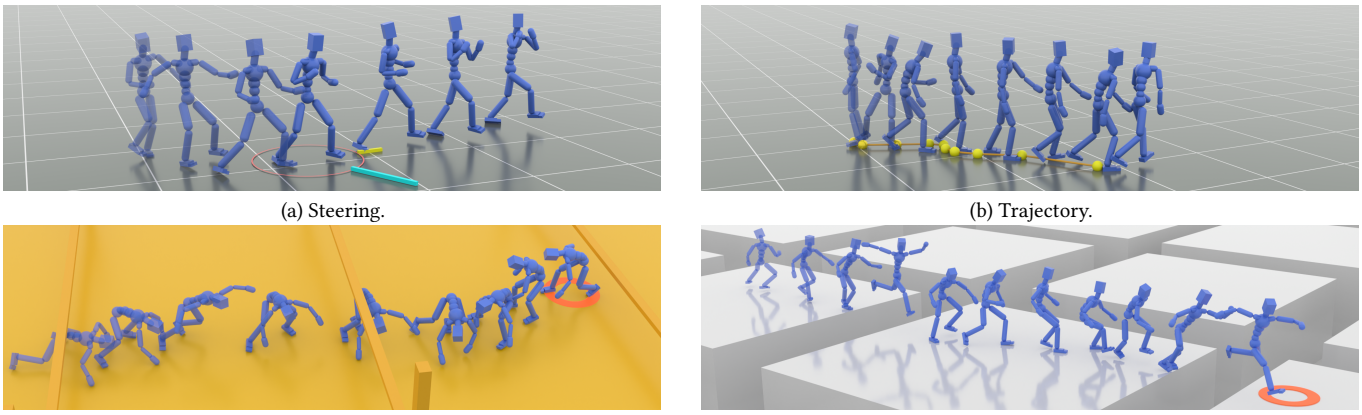
(b) The character stabilizes with a front roll and gets up when encountering two subsequent forces.

Fig. 6. GPC exhibits diverse and versatile responses to force perturbations with a generative controller trained with Bones.

In the second stage, Generative Controller Training (Section 6), we train a generative controller that models the distribution of discrete skill tokens using a transformer decoder. The generative controller autoregressively generates sequences of tokens that drive the character to produce naturalistic behaviors. Given the character's state and a selected latent token, the pretrained decoder outputs actions that drive the movement of the character's body. This method leads to the emergence of rich, reusable behaviors such as life-like responses and recoveries from perturbations. These behaviors emerge without requiring explicit rewards or specialized training. Finally, in Section 7, we introduce a parameter-efficient fine-tuning (PEFT) procedure that can adapt the pretrained generative controller to a wide range of downstream control tasks using only a small number of additional parameters, while preserving the diversity and naturalness of the pretrained controller.

5 Skill Quantization

Training GPC requires a discrete latent representation of motor skills. This is obtained by training an FSQ motion-tracking controller, where the discrete codes are optimized via end-to-end RL to model different skills. We follow a standard motion-tracking framework as described in DeepMimic [Peng et al. 2018]. Given a reference motion sequence which specifies target states $\hat{\mathbf{s}}_{t:T}$ at each timestep t, a policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t, \hat{\mathbf{s}}_{t:t+h})$ is trained to select the appropriate sequence of actions that enable the simulated character to imitate the reference motion. The policy is conditioned on the character's proprioceptive state $\mathbf{s}_t$ at time step t and a sequence of h future target states from the reference motion. The training objective is defined via a tracking reward $\mathbf{r}_t$. The reward measures the discrepancy between the simulated character state $\mathbf{s}_{t+1}$ and the reference motion. While this formulation is sufficient for accurately reproducing a wide range of behaviors [Luo et al. 2023; Tessler et al. 2024], our



(a) Steering. (b) Trajectory. (c) Barrier. (d) Platform.

Fig. 7. CoLA can be used to efficient finetune GPC to new tasks. By only training adaptation layers during finetuning, the pretrained generative controller can effectively complete downstream tasks while preserving natural behaviors encoded in the original base model.

goal is to learn a structured and reusable *discrete* representation of skills that supports generative modeling and downstream control. We incorporate FSQ-based quantization into the policy through an encoder–decoder design architecture. The encoder maps a sequence of target states $\hat{\mathbf{s}}_{t:t+h}$ to a continuous latent vector $\mathbf{z_t} = \mathcal{E}(\hat{\mathbf{s}}_{t:t+h}) \in \mathbb{R}^d$. Each dimension of this latent is then independently quantized using FSQ into L fixed scalar levels,

$$\hat{\mathbf{z}}_t = \left\lfloor \left\lfloor \frac{L}{2} \right\rfloor \tanh(\mathbf{z}_t) \right\rceil, \tag{4}$$

where $\lfloor \cdot \rceil$ denotes element-wise rounding. This look-up-free quantization defines an implicit discrete codebook of size L^d, alleviating the need to learn an explicit codebook, as done in VQ-VAEs [van den Oord et al. 2018], and thereby mitigating common pitfalls of VQ-VAEs such as codebook collapse [Mentzer et al. 2023]. The resulting discrete latent code $\hat{\mathbf{z}}_t \in \mathbb{R}^d$ serves as a compact representation of the target motion $\hat{\mathbf{s}}_{t:t+h}$. The decoder $\mathcal{D}$ receives the character's current state and discrete code as input, and outputs the actions, $\mathbf{a}_t \sim \mathcal{D}(\mathbf{a}_t \mid \mathbf{s}_t, \hat{\mathbf{z}}_t)$. The actions represent target joint rotations for proportional-derivative (PD) controllers positioned at each of the character's joints. The PD controller converts these targets into joint torques that are applied to drive the movement of the simulated character. The encoder–decoder policy is trained end-to-end with Proximal Policy Optimization (PPO) [Schulman et al. 2017]. Gradients are propagated through the FSQ quantization operation using a straight-through estimator (STE) [Bengio et al. 2013].

6 Generative Controller

The encoder from the previous quantization stage can produce a discrete latent code. This representation yields a sequence of d discrete tokens at each time step. However, directly modeling all d L-ary tokens results in long token sequences, which increases the computational and memory cost of autoregressive modeling. To reduce sequence length, a fixed grouping scheme is applied that packs every G consecutive L-ary tokens into a token with a larger vocabulary. In our experiments, we find this simple grouping scheme for creating grouped token strikes an effective balance between vocabulary size and sequence length. This method reduces both the context length and the cost of self-attention as shown in Table 3.

The grouped token sequence $\tilde{\mathbf{z}}_t = (\tilde{\mathbf{z}}_t^0, \ldots, \tilde{\mathbf{z}}_t^{d'-1})$ can be treated as a skill representation, and its joint conditional distribution is modeled autoregressively such that each token is generated based on the character's current state and all previously generated tokens:

$$p_\theta(\tilde{\mathbf{z}}_t \mid \mathbf{s}_t) = p_\theta(\tilde{\mathbf{z}}_t^0 \mid \mathbf{s}_t) \prod_{j=1}^{d'-1} p_\theta(\tilde{\mathbf{z}}_t^j \mid \mathbf{s}_t, \tilde{\mathbf{z}}_t^{<j}). \tag{5}$$

This factorization is modeled using a GPT-style transformer decoder with causal self-attention, where each token j attends only to previously generated tokens $\tilde{\mathbf{z}}_t^{<j}$, ensuring consistency between training and inference. At each step, the transformer predicts a conditional categorical distribution over the next latent code $\tilde{\mathbf{z}}_t^j$. The model is trained using a cross-entropy objective,

$$\mathcal{L}_{\text{CE}} = -\sum_{j=0}^{d'-1} \log p_\theta\left(\tilde{\mathbf{z}}_t^j \mid \mathbf{s}_t, \tilde{\mathbf{z}}_t^{<j}\right). \tag{6}$$

This training objective encourages the transformer to capture the temporal dependencies among sequences of grouped tokens.

During inference time, at each timestep t, we autoregressively sample latent tokens $\tilde{z}_t^j \sim p(\tilde{z}_t^j \mid \mathbf{s}_t, \tilde{\mathbf{z}}_t^{<j})$ to determine the action that should be executed by the controller. At each decoding step, the transformer produces a categorical distribution over L^G discrete codes, from which the next token is sampled using nucleus (top-p) sampling applied to the softmax-normalized logits [Holtzman et al. 2020]. Nucleus sampling restricts sampling to a subset of the most likely tokens, which mitigates the chances of sampling low-probability outliers while still preserving diversity in the generated behaviors. The sampled token is then provided as input at the subsequent step to generate the next token. The final sequence of tokens $\tilde{\mathbf{z}}_t$ is decoded by the frozen FSQ decoder to generate an action given the character's current state.

7 Task Adaptation

Once trained, GPC can be adapted to task-specific objectives while preserving pretrained behaviors. However, finetuning the full model for each downstream task can be computationally and data intensive. To address this, we introduce Conditional Low-rank Adaptation (CoLA), which enables parameter-efficient fine-tuning (PEFT) through lightweight adaptation layers that introduce less than 1% additional parameters to the model.

7.1 Conditional Low-rank Adaptation

CoLA serves as a lightweight adaptation framework that enables specialized learning without the overhead of updating the entire model architecture. CoLA extends the DoRA strategy by incorporating task-specific modulation within a low-rank space [Liu et al. 2024]. Given a pretrained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the adaptation decomposes weight updates into separate magnitude and direction components. To condition the model on task-specific observations $\mathbf{c}$, CoLA applies Feature-wise Linear Modulation (FiLM) to the low-rank components [Perez et al. 2018],

$$\mathbf{W}\mathbf{x} = \mathbf{W}_0\mathbf{x} + \mathbf{m}\frac{\mathbf{B}\left(\text{diag}(\boldsymbol{\gamma}(\mathbf{c}))\,\mathbf{A}\mathbf{x} + \boldsymbol{\beta}(\mathbf{c})\right)}{\left\|\mathbf{B}\left(\text{diag}(\boldsymbol{\gamma}(\mathbf{c}))\,\mathbf{A}\mathbf{x} + \boldsymbol{\beta}(\mathbf{c})\right)\right\|_F}, \tag{7}$$

where $\mathbf{x}$ is the input to a particular layer, m is a learned vector controlling the overall magnitude. $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ are trainable low-rank matrices with $r \ll \min(d_{\text{in}}, d_{\text{out}})$. $\boldsymbol{\gamma}(\mathbf{c})$ and $\boldsymbol{\beta}(\mathbf{c})$ are produced by lightweight MLPs, and apply task-conditioned modulation in the low-rank space $\mathbb{R}^r$. This design ensures stability through decoupled updates, while providing flexible task modulation with minimal additional parameters.

7.2 Fine-Tuning via Reinforcement Learning

Task adaptation is performed via reinforcement learning fine-tuning (RLFT) by optimizing task-specific reward functions. Training is carried out using Proximal Policy Optimization (PPO) [Schulman et al. 2017]. The action space consists of sequences of discrete skill tokens with length d', where each token is drawn from an L^G-ary categorical distribution corresponding to grouped latent skill representations $\tilde{\mathbf{z}}$. At each timestep, the controller autoregressively predicts a token sequence $\tilde{\mathbf{z}}$ conditioned on the current character state and task observations, which is decoded into actions by the frozen FSQ decoder. The generative adapted controller outputs logits over an L^G-ary categorical distribution for each token, and per-token log-probabilities are aggregated to form the PPO objective, with advantages shared across tokens within the same decision step. During rollouts, exploration is regularized using Nucleus Sampling guided by the unconditional generative model. Nucleus sampling only samples tokens from high-probability regions of the predicted distribution, thereby helping to preserve natural behaviors when fine-tuning GPC on downstream tasks.

7.3 Supervised Fine-Tuning

In addition to reinforcement learning fine-tuning, supervised fine-tuning (SFT) provides an effective mechanism for guiding task adaptation when example motions are available. Given a large pretrained generative controller that models a diverse set of skills, exploration during task training can become inefficient. SFT mitigates this challenge by leveraging example motions to bias adaptation toward a subset of skills that are appropriate for the target task. The generative controller is adapted by minimizing a cross-entropy loss over the discrete latent codes, effectively increasing the likelihood of dataset-specific skill tokens conditioned on the character state. This fine-tuning process biases the controller toward selecting desired skills when performing exploration during RL finetuning. The generative controller is first adapted via supervised fine-tuning (SFT) by minimizing a cross-entropy loss over the discrete latent codes, which increases the likelihood of selecting task-relevant skill tokens during exploration in subsequent RL fine-tuning. Starting from the SFT-adapted initialization, the model can then be further refined through reinforcement learning fine-tuning (RLFT) to reliably compose and execute these skills for task completion.

Table 1. Skill quantization performances. We compare our method trained end-to-end against an MLP baseline and a VQ-VAE model. Success rate (**Succ.**) measures the fraction of evaluation episodes that successfully complete the tracking task. An episode is deemed successful if the average joint position error, computed over all frames, is below 0.5 m. **MPJPE** stands for Mean Per-Joint Position Error, measured in millimeters (mm).

	Bones [680 hr]		**AMASS [40 hr]**	
Method	**Succ. (%) ↑**	**MPJPE ↓**	**Succ. (%) ↑**	**MPJPE ↓**
MLP	**99.98**	**25.56**	**99.59**	**30.26**
VQ-VAE	99.94	37.92	99.30	59.28
FSQ	99.98	34.90	99.51	44.43

Table 2. Comparison between end-to-end trained FSQ and a variant (denoted as FSQ-K) in which the encoder is first trained on kinematic reference data using supervised learning and then kept frozen during subsequent policy training with reinforcement learning. **Util.** denotes the codebook utilization rate that is averaged over grouped tokens.

Method	**Succ. (%) ↑**	**MPJPE ↓**	**Util. (%) ↑**
FSQ-K	99.03	78.26	76.34
FSQ	**99.98**	**34.90**	**82.15**

8 Experiment

We evaluate our proposed framework through experiments that examine the impact of end-to-end RL training and parameter-efficient fine-tuning on downstream task performance (Section 8.5). Next, we evaluate the performance of different skill quantization methods (Section 8.2), and analyze the effects of different token grouping strategies on the performance of the resulting models (Section 8.4).

8.1 Experimental Setup

All experiments are conducted in a physically simulated environment implemented with Isaac Gym [Makoviychuk et al. 2021]. Our training and evaluation pipeline is built on top of the ProtoMotions framework [Tessler et al. 2025], which provides scalable infrastructure for large-scale motion tracking and reinforcement learning with simulated characters. For the Bones dataset [Bones Studio

Table 3. Ablation of the grouping factor G for a generative controller ($L = 9$, $d = 40$). G denotes the number of tokens per group and N_{vocab} the resulting vocabulary size. We report motion quality metrics including average pairwise distance (APD, m) [Aliakbarian et al. 2020; Rempe et al. 2021], average displacement error (ADE, m), and acceleration error (Accel., m/s^2), as well as computational cost. We find that for $G \geq 8$, the computational cost becomes prohibitive in terms of memory usage, and the resulting vocabulary size far exceeds that of modern LLMs [Bai et al. 2023; Radford et al. 2019].

G	N_{vocab}	N_{token}	**APD (m)** ↑	**ADE (m)** ↓	**Accel. (m/s^2)**	N_{param}	**Mem (GB)**	**FPS**
8	4.3×10^7	5	-	-	-	4.4×10^{10}	1.7×10^2	-
5	59,049	8	0.34	0.30	3.67	6.0×10^7	0.27	92.85
4	6,561	10	0.29	0.29	3.21	6.7×10^5	0.03	115.47
2	81	20	0.26	0.27	3.04	8.3×10^4	<0.01	56.43
1	9	40	0.27	0.24	2.88	9×10^3	<0.01	25.15

Table 4. We compare the performance of a controller fine-tuned sequentially via SFT and RL against a baseline optimized only using RL without SFT. Incorporating SFT biases the controller toward a more restricted set of skills, yielding less diverse behaviors, with decreased APD, ADE, and entropy. This reduction in diversity does not compromise task performance, as shown by comparable returns and robust performance under external perturbations.

Method	**Return**	**Pert. Ret.**	**Entropy**	**APD**	**ADE**
SFT	143.36	125.44	2.57	0.24	0.15
w/o SFT	230.42	176.35	5.08	0.26	0.27

2026], both the FSQ-based tracking controller and the generative controller are trained using 24 NVIDIA A100 GPUs. The rest of the experiments are conducted on a single A100 GPU. Once trained, the models are capable of running on consumer-grade hardware with NVIDIA RTX 4090 GPU. Our FSQ model uses 40 discrete latent tokens with 9 quantization levels per token, resulting in an implicit codebook of size 9^{40}.

8.2 Skill Quantization

To evaluate the effectiveness of different quantization methods when applied to motion tracking, we compare our FSQ-based tracking controller against a VQ-VAE-based controller and an MLP baseline without a quantization layer on Bones [Bones Studio 2026] and AMASS [Mahmood et al. 2019]. All models are trained end-to-end under the same training protocol. The results are summarized in Table 1. The MLP baseline achieves the best tracking accuracy across the various metrics, and its strong performance can likely be attributed to the absence of a quantization bottleneck. When comparing different quantization methods, our FSQ model with reinforcement learning consistently outperforms the VQ-VAE–based approach. Our method attains higher success rates and lower MPJPE, indicating improved overall tracking performance. Furthermore, FSQ does not require any of the additional stabilization techniques commonly needed for VQ-VAE methods, such as codebook heuristics or auxiliary losses. Qualitative examples of the motions produced by our controller are shown in Figure 2.

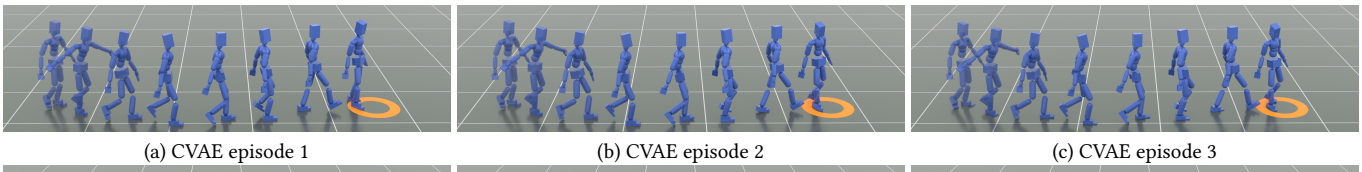
A key characteristic of our approach is that all components of the model, including the encoder and decoder, are trained through end-to-end RL, which enables the latent representations to be directly optimized to be amenable to physics-based control. In contrast, several prior works with discrete latent representations train VQ-VAE models via supervised distillation from pretrained tracking controllers, rather than optimizing the representations in an end-to-end manner [Bae et al. 2025b]. To evaluate the impact of end-to-end RL training, we compare our model against a variant that uses a pretrained FSQ encoder that was trained strictly on kinematic motion data with supervised learning. As shown in Table 2, the model trained with end-to-end RL consistently outperforms the variant with a pretrained kinematic encoder, demonstrating the importance of end-to-end optimization for learning representations that are well suited for physics-based control.

8.3 Generative Controller

By training GPC on large-scale motion datasets, the learned generative controller can produce a wide range of life-like skills and natural transitions between diverse behaviors. Figure 4 illustrates examples of the diverse behaviors produced through unconditional sampling from GPC starting at the same initial state. GPC is able to produce a wide range of behaviors, including rolling, jumping, dancing, and acrobatics. Beyond generating diverse behaviors, the controller also exhibits emergent human-like responses to external perturbations. As shown in Fig. 6, when the character falls down, GPC automatically transitions to a fall-recovery behavior to get back up.

8.4 Token Grouping

Token grouping is an important design choice for improving GPC's performance while reducing the inference cost associated with long context lengths. As shown in Table 3, we evaluate several grouping strategies that yield grouped tokens with a vocabulary size of L^G. A grouping factor of $G = 5$ achieves the highest APD, indicating increased behavioral diversity, at the expense of reduced tracking accuracy and smoothness, as reflected by higher ADE and joint accelerations. Despite this trade-off, this setting produces the best motion quality in qualitative evaluations. This improvement may be attributed to the reduced sequence length, thereby lowering the challenges of modeling the relationship across long token sequences. Token grouping may also allow each token to model higher-level semantic information of different behaviors. A similar effect has been observed in language modeling, where moving from character-level to subword-level representations improves both efficiency and



(a) CVAE episode 1 (b) CVAE episode 2 (c) CVAE episode 3

(d) GPC episode 1 (e) GPC episode 2 (f) GPC episode 3

Fig. 8. Comparison between downstream task behavior of GPC and CVAE under identical task conditions. GPC produces more diverse behaviors than CVAE.

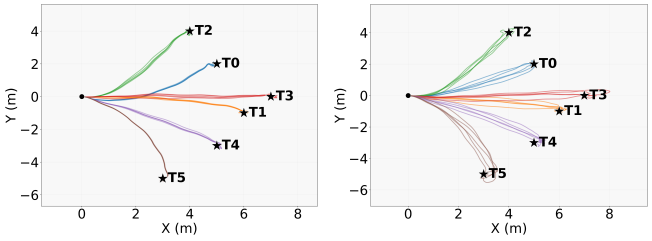


Fig. 9. Trajectory comparison. The CVAE baseline (left) collapses to similar behaviors, while GPC (right) produces diverse trajectories.

expressiveness by embedding semantic structure directly into the tokens [Kim et al. 2015; Sennrich et al. 2016].

8.5 Tasks

To validate the effectiveness of our pipeline, we apply our proposed adaptation approach to fine-tune a pretrained generative controller for a variety of downstream tasks. We evaluate our framework across locomotion tasks, including target reaching, trajectory following, and joystick steering, alongside scene interaction tasks that require robust behavior and the mastery of dynamic skills like jumping and crawling. Figures 7 demonstrate the successful completion of these tasks. During task execution, the adapted controller retains the emergent behaviors of the original generative controller while remaining highly robust to external perturbations.

8.6 Supervised Fine-Tuning

To validate the effect of SFT on downstream task adaptation, we compare task performance and behavioral characteristics of controllers adapted with SFT first against controllers finetuned purely via RL. In both settings, models are initialized with the same GPC model. The SFT variant is fine-tuned on a 20-second crouch walk motion. We evaluate each model over 512 episodes and report policy entropy and the averaged test return. Table 4 indicates that the SFT-trained controller leads to lower entropy in the predicted logits of the adapted generative controller, indicating more consistent stylized behaviors. This observation is qualitatively supported by Figure 11, which illustrates the trajectories of the head height. The SFT model exhibits a clear preference for maintaining a crouched

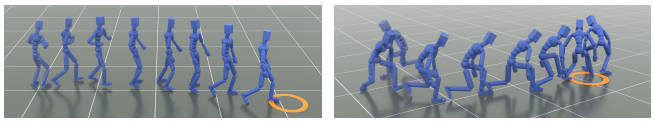


Fig. 10. Qualitative comparison of task completion with and without SFT (left). SFT enables stylistic control by biasing the controller towards a desired behavioral style, such as crouched walking (right).

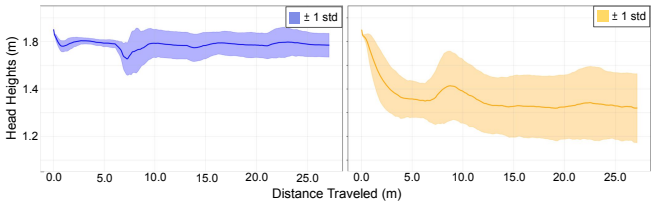


Fig. 11. Trajectories of head joint height during task completion for controllers adapted (left) without SFT and (right) with SFT on a crouch motion.

posture, as indicated by the averaged head height being within the range 0.8-1.3m.

8.7 Diverse Behaviors on Downstream Tasks

As illustrated in Figures 8 and 9, the adapted GPC retains the stochasticity from the base generative controller when solving downstream tasks, as skill codes are sampled from the distribution over latent codes. This stochasticity allows the policy to preserve behavioral diversity, exhibiting diverse behaviors when performing identical tasks. In contrast, the CVAE-based task controller behaves deterministically, as the task controller directly selects the latent codes. Note that the slight trajectory variations observed for the CVAE baseline in Figure 9 (left) stem from simulator stochasticity rather than from the model. The CVAE policy lacks behavioral diversity, repeatedly executing similar motions as shown in Figure 8.

9 Discussion and Limitations

In this work, we present GPC, a generative controller trained on a large-scale motion dataset with over 600 hours of motion clips. While GPC has been effective for modeling a wide range of motor skills and adapting the learned skills to new tasks, our framework

has several limitations. First, we primarily focus on locomotion-based tasks. Exploring additional multimodal extensions could enable more flexible and intuitive control, such as incorporating higher-level text-based conditioning to guide task completion. Furthermore, extending the framework to human–object interaction tasks would further broaden its potential applications. We believe these directions represent promising avenues for future work toward more general-purpose and scalable generative controllers for physics-based character animation.

10 Acknowledgments

We thank Yuxuan Mu, Ziyu Zhang, Dun Yang, Kaifeng Zhao, Sunmin Lee, Haotian Zhang, and Davis Rempe for their support and insightful discussions.

References

Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Lars Petersson, and Stephen Gould. 2020. A Stochastic Conditioning Scheme for Diverse Human Motion Prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5222–5231.

Jinseok Bae, Younghwan Lee, Donggeun Lim, and Young Min Kim. 2025a. PLT: Part-Wise Latent Tokens as Adaptable Motion Priors for Physically Simulated Characters. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 132, 10 pages. doi:10.1145/3721238.3730637

Jinseok Bae, Jungdam Won, Donggeun Lim, Inwoo Hwang, and Young Min Kim. 2025b. Versatile Physics-based Character Control with Hybrid Latent Representation. *Computer Graphics Forum* (2025). doi:10.1111/cgf.70018

Alexei Baevski, Steffen Schneider, and Michael Auli. 2020. vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations. arXiv:1910.05453 [cs.CL]

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen Technical Report. arXiv:2309.16609 [cs.CL] https://arxiv.org/abs/2309.16609

Yoshua Bengio, Nicolas Léonard, and Aaron Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432* (2013).

Bones Studio. 2026. BONES-SEED: Skeletal Everyday Embodied Dataset. https://bones.studio/datasets.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. arXiv:2106.01345 [cs.LG] https://arxiv.org/abs/2106.01345

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-based Motion Capture Imitation with Deep Reinforcement Learning. In *Proceedings of the 11th International Conference on Motion, Interaction and Games (MIG '18)*. ACM, Limassol, Cyprus, Article 4, 10 pages. doi:10.1145/3274247.3274506

Stelian Coros, Philippe Beaudoin, Kang Kang Yin, and Michiel van de Panne. 2008. Synthesis of constrained walking skills. In *ACM SIGGRAPH Asia 2008 Papers* (Singapore) *(SIGGRAPH Asia '08)*. Association for Computing Machinery, New York, NY, USA, Article 113, 9 pages.

Marco da Silva, Yeuhi Abe, and Jovan Popović. 2008. Simulation of Human Motion Data Using Short-Horizon Model-Predictive Control. *Computer Graphics Forum* 27, 2 (2008), 371–380.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv:2305.14314 [cs.LG]

Zhiyang Dou, Xuelin Chen, Qingnan Fan, Taku Komura, and Wenping Wang. 2023. C·ASE: Learning Conditional Adversarial Skill Embeddings for Physics-based Characters. (2023). arXiv:2309.11351

Patrick Esser, Robin Rombach, and Björn Ommer. 2021. Taming Transformers for High-Resolution Image Synthesis. arXiv:2012.09841 [cs.CV]

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [stat.ML]

Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. 2024. Momask: Generative masked modeling of 3d human motions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1900–1910.

Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 60–1.

Dan Hendrycks and Kevin Gimpel. 2023. Gaussian Error Linear Units (GELUs). arXiv:1606.08415 [cs.LG] https://arxiv.org/abs/1606.08415

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

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.

Neville Hogan and Dagmar Sternad. 2009. Sensitivity of Smoothness Measures to Movement Duration, Amplitude, and Arrests. *Journal of Motor Behavior* 41, 6 (2009), 529–534. doi:10.3200/35-09-004-RC

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. arXiv:1904.09751 [cs.CL]

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL]

Xiaoyu Huang, Takara Truong, Yunbo Zhang, Fangzhou Yu, Jean Pierre Sleiman, Jessica Hodgins, Koushil Sreenath, and Farbod Farshidian. 2025. Diffuse-CLoC: Guided Diffusion for Physics-based Character Look-ahead Control. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–13. SIGGRAPH 2025.

Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline Reinforcement Learning as One Big Sequence Modeling Problem. arXiv:2106.02039 [cs.LG] https://arxiv.org/abs/2106.02039

Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. 2023. Motiongpt: Human motion as a foreign language. *Advances in Neural Information Processing Systems* 36 (2023), 20067–20079.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. 2024. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-Aware Neural Language Models. arXiv:1508.06615 [cs.CL]

Diederik P Kingma and Max Welling. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114 [stat.ML]

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2023. Flow Matching for Generative Modeling. arXiv:2210.02747 [cs.LG] https://arxiv.org/abs/2210.02747

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. 2025. Muon is Scalable for LLM Training. arXiv:2502.16982 [cs.LG]

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. DoRA: Weight-Decomposed Low-Rank Adaptation. arXiv:2402.09353 [cs.CL]

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101 [cs.LG]

Zhengyi Luo, Jinkun Cao, Kris Kitani, Weipeng Xu, et al. 2023. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10895–10904.

Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. 2024. Universal Humanoid Motion Representations for Physics-Based Control. arXiv:2310.04582 [cs.CV]

Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. 2019. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5442–5451.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. arXiv:2108.10470 [cs.RO]

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2023. Finite Scalar Quantization: VQ-VAE Made Simple. arXiv:2309.15505 [cs.CV]

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. 2019. Neural probabilistic motor primitives for humanoid control. arXiv:1811.11711 [cs.LG]

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, Andreas Vlachos and Isabelle Augenstein (Eds.). Association for Computational Linguistics, Dubrovnik, Croatia, 2014–2037. doi:10.18653/v1/2023.eacl-main.148

Rafael Müller, Simon Kornblith, and Geoffrey Hinton. 2020. When Does Label Smoothing Help? arXiv:1906.02629 [cs.LG] https://arxiv.org/abs/1906.02629

Liang Pan, Zeshi Yang, Zhiyang Dou, Wenjia Wang, Buzhen Huang, Bo Dai, Taku Komura, and Jingbo Wang. 2025. Tokenhsi: Unified synthesis of physical human-scene interactions through task tokenization. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 5379–5391.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach 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*. Curran Associates Inc., Red Hook, NY, USA.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)* 37, 4 (2018), 1–14.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *Acm transactions on graphics (tog)* 36, 4 (2017), 1–13.

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 Transactions On Graphics (TOG)* 41, 4 (2022), 1–17.

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 (ToG)* 40, 4 (2021), 1–20.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. arXiv:2005.00247 [cs.CL]

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. OpenAI technical report.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084 [cs.CL] https://arxiv.org/abs/1908.10084

Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. 2021. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 11488–11499.

Stéphane Ross, Geoffrey Gordon, and J. Andrew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1715–1725. doi:10.18653/v1/P16-1162

Sebastian Starke, Paul Starke, Nicky He, Taku Komura, and Yuting Ye. 2024. Categorical codebook matching for embodied character controllers. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–14.

Richard S Sutton, Andrew G Barto, et al. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. 2024. MaskedMimic: Unified Physics-Based Character Control Through Masked Motion Inpainting. *ACM Transactions on Graphics (TOG)* (2024).

Chen Tessler, Yifeng Jiang, Xue Bin Peng, Erwin Coumans, Yi Shi, Haotian Zhang, Davis Rempe, Gal Chechik†, and Sanja Fidler. 2025. ProtoMotions3: An Open-source Framework for Humanoid Simulation and Control. https://github.com/NVLabs/ProtoMotions/.

Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. 2023. CALM: Conditional Adversarial Latent Models for Directable Virtual Characters. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA.

Guy Tevet, Sigal Raab, Setareh Cohan, Daniele Reda, Zhengyi Luo, Xue Bin Peng, Amit H Bermano, and Michiel van de Panne. 2024. Closd: Closing the loop between simulation and diffusion for multi-task character control. *arXiv preprint arXiv:2410.03441* (2024).

Takara Everest Truong, Michael Piseno, Zhaoming Xie, and Karen Liu. 2024. Pdp: Physics-based character animation via diffusion policy. In *SIGGRAPH Asia 2024 Conference Papers*. 1–10.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2018. Neural Discrete Representation Learning. arXiv:1711.00937 [cs.LG]

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

Jingbo Wang, Zhengyi Luo, Ye Yuan, Yixuan Li, and Bo Dai. 2024. PACER+: On-Demand Pedestrian Animation Controller in Driving Scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 718–728.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-based Character Controllers Using Conditional VAEs. *ACM Trans. Graph.* 41, 4, Article 96 (2022).

Jungdam Won and Jehee Lee. 2019. Learning body shape variation in physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12. doi:10.1145/3355089.3356499

Michael Xu, Yi Shi, KangKang Yin, and Xue Bin Peng. 2025. Parc: Physics-based augmentation with reinforcement learning for character controllers. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*. 1–11.

Heyuan Yao, Zhenhua Song, Yuyang Zhou, Tenglong Ao, Baoquan Chen, and Libin Liu. 2024. Moconvq: Unified physics-based motion control via scalable discrete representations. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–21.

Yuting Ye and C. Karen Liu. 2010. Optimal feedback control for character animation using an abstract model. *ACM Trans. Graph.* 29, 4, Article 74 (July 2010), 9 pages.

Yuan Ye, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. 2023. PhysDiff: Physics-Guided Human Motion Diffusion Model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

KangKang Yin, Kevin Loken, and Michiel Van de Panne. 2007. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 105–es.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023b. Adding Conditional Control to Text-to-Image Diffusion Models.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023a. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *The Eleventh International Conference on Learning Representations*.

Ziyu Zhang, Sergey Bashkirov, Dun Yang, Yi Shi, Michael Taylor, and Xue Bin Peng. 2025. Physics-Based Motion Imitation with Adversarial Differential Discriminators. In *SIGGRAPH Asia 2025 Conference Papers (SIGGRAPH Asia '25 Conference Papers)*.

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.

Qingxu Zhu, He Zhang, Mengting Lan, and Lei Han. 2023. Neural categorical priors for physics-based character control. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–16.

Appendix

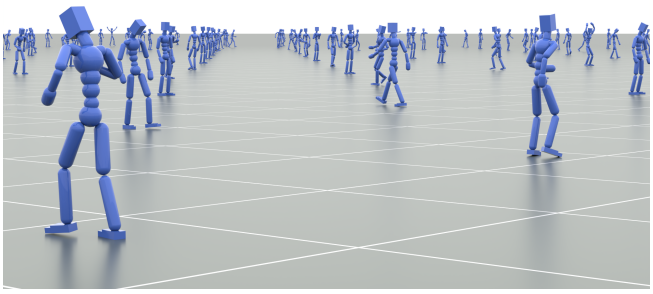


Fig. 12. Sampled motions from Bones.

A. Dataset and Simulated Humanoid Configurations

We now introduce the datasets used in our experiments.

AMASS is an aggregation of multiple motion capture datasets, covering a broad spectrum of human behaviors [Mahmood et al. 2019]. We follow the dataset segmentation and humanoid configuration from prior studies [Luo et al. 2023; Tessler et al. 2024]. The dataset comprises approximately 11,300 motion clips with a total duration of 40 hours. Owing to its moderate scale, we use it for our ablation experiments.

Bones is a large-scale motion capture dataset comprising approximately 343,000 clips totaling over 680 hours in length [Bones Studio 2026]. The collection includes a broad repertoire of daily motions from stylized walking and in-place motions to dynamic acrobatics (Figure 12). These sequences feature a mean length of 200 frames and a median duration of 160 frames. Our experiments on Bones utilize a customized humanoid character modeled with 23 rigid bodies and 66 actuated degrees of freedom (DoFs). Each non-root joint is modeled as a 3-DoF rotational joint, parameterized by three orthogonal hinge axes, providing full rotational freedom at each articulation. The humanoid has a standing height of 1.85 m and an arm span of 1.75 m. A visualization of the humanoid is shown in Figure 13.

LAFAN1 (Ubisoft La Forge Animation Dataset) is a high-quality dataset comprising motion capture sequences with a duration of 4.6 hours [Harvey et al. 2020]. Captured from five distinct subjects, the dataset encompasses 77 sequences across 15 categories, including standard locomotion, stylized dynamic maneuvers, and recovery from falls.

Beyond is a motion dataset first introduced in PARC [Xu et al. 2025], providing parkour-style locomotions. We use 14 unique motion capture clips, with a total length of 18,600 frames and 10.3 minutes, covering diverse athletic and dynamic locomotion behaviors, including crawling and long strides, parkour rolls and hops, sharp-turn running with obstacle avoidance, and transitional actions such as getting up from the ground.

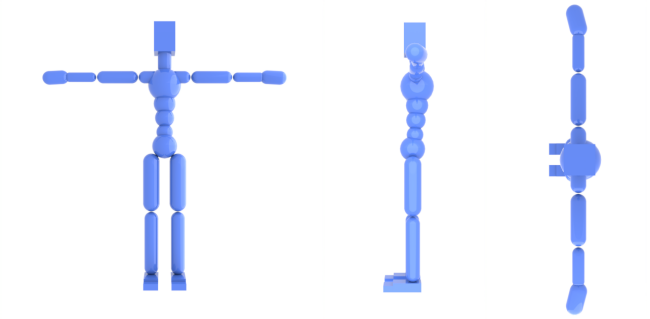


Fig. 13. The simulated character used in experiments conducted on Bones.

B. Skill Quantization

B.1. Observation

The observation of the FSQ-based motion tracking controller is composed of the character state $\mathbf{s}_t$ and a window of reference motion states $\hat{\mathbf{s}}_{t:t+h}$, where h stands for set of target frame indices. Following prior work [Peng et al. 2018; Tessler et al. 2024], the character state $\mathbf{s}_t$ consists of proprioceptive information: joint positions relative to the root, root height relative to the terrain, joint orientations in the 6D rotation representation [Zhou et al. 2019], and joint linear and angular velocities. All features are expressed in a canonicalized local coordinate frame aligned with the root joint to promote translational and rotational invariance. The reference motion states use the same feature representation. We use a fixed window of target motions consisting of the next $(1, 2, 5, 7, 12, 18, 25)$ frames from the reference motion. This dilated window design enables the representation to capture longer information while remaining compact and easy to model during training. Our framework decouples skill representation learning from action generation across the encoder and decoder. The encoder takes only the reference motion states $\hat{\mathbf{s}}_{t:t+h}$ as input and produces a discrete skill latent, while the decoder conditions on the current character state $\mathbf{s}_t$ and the latent to generate actions. This separation prevents a potential failure mode in which the encoder bypasses the decoder, and discourages the decoder from ignoring the character state and over-relying on the latent code produced by the encoder. As a result, the discrete latent space is encouraged to capture state-independent, reusable motion skills that can be effectively leveraged for training the generative controller.

B.2. Reward

The tracking reward r in our framework is designed to encourage the simulated character to reproduce the kinematic reference motions from the dataset.

$$r = w_{\text{gp}}\, r_{\text{gp}} + w_{\text{gr}}\, r_{\text{gr}} + w_{\text{rh}}\, r_{\text{rh}} + w_{\text{jv}}\, r_{\text{jv}} + w_{\text{jav}}\, r_{\text{jav}} \tag{8}$$

$r_{(\cdot)}$ denote the individual reward terms and $w_{(\cdot)}$ their corresponding weights. Each term measures the discrepancy between the reference motion and the simulated character in terms of global joint positions (gp), global joint rotations (gr), root height (rh), joint velocities (jv), and joint angular velocities (jav).

Table 5. Composite Reward Components for our tracking controller

Name	Weight w_k	Coeff. c_k
r_{gp}	0.5	−100
r_{gr}	0.3	−5
r_{jv}	0.1	−0.5
r_{jav}	0.1	−0.1
r_{rh}	0.2	−100

Table 6. Hyperparameter configurations of Skill Quantization.

Encoder	Layer	1024, 1024, 1024, 512, 256
	Activation	ReLU
	Output dim	40
Decoder	Layer	1024, 1024, 1024, 512, 256
	Activation	ReLU
	Output dim	N_{DoF}
Critic	Layer	1024, 1024, 1024, 1024
	Activation	ReLU
	Output dim	1
FSQ	N_{level}	9
	N_{token}	40
PPO	Actor optimizer	Muon
	Critic optimizer	AdamW
	Clip ratio ϵ	0.2
	GAE λ	0.95
	Discount factor γ	0.99

B.3. Optimizer

When training FSQ tracking controller, we optimize the actor with Muon [Jordan et al. 2024], and train the critic with AdamW [Loshchilov and Hutter 2019]. Muon is a geometry-aware optimizer tailored to matrix-shaped parameters, which can greatly speed up the training. We follow a procedure where it is only applied to the hidden weight matrices of the encoder-decoder, an auxiliary AdamW optimizer is used for low-dimensional and interface parameters (e.g., biases, the encoder and decoder boundary linear layers), which improves stability while retaining Muon's favorable scaling behavior on the internal representation-learning weights [Liu et al. 2025].

B.4. Prioritized Sampling

During skill quantization, our objective is to train a single tracking policy capable of imitating hundreds of hours of motion clips. When learning from large motion datasets, a common strategy to improve sample efficiency and success rate is to apply prioritized sampling, which allocates more training updates to difficult or underperforming motions [Won and Lee 2019; Zhu et al. 2023]. We adopt a success-rate-based sampling weight update strategy from the tracking controller training setting in ProtoMotions [Tessler et al. 2025]. Tracking a motion is considered successful if its maximum per-joint tracking error over the episode falls below a threshold τ, where we use $\tau = 0.5$ m. We assign each motion clip a unique index m and associate it with a sampling weight w_m that determines its probability of being selected for rollouts during training. During evaluation, we partition motions into a success set $\mathcal{S}$ and a failure set $\mathcal{F}$ according to this criterion, and update weights by down-weighting successful motions and up-weighting failed motions:

$$w_m \leftarrow \begin{cases} w_m \cdot \gamma^K, & m \in \mathcal{S}, \\ w_m / \gamma^K, & m \in \mathcal{F}, \end{cases} \tag{9}$$

where $\gamma \in (0, 1)$ controls the adjustment magnitude and K is the number of training epochs between consecutive evaluations, so that the cumulative weight adjustment γ^K scales with the length of the update interval.

Table 7. Comparison of tracking controllers on the AMASS PHC train subset. PDP and FPO are diffusion-based tracking policies. PDP is trained via DAgger distillation from an MLP tracking policy [Ross et al. 2011], whereas all other methods are trained end-to-end from scratch with model-free RL. The MLP baseline remains the strongest overall, achieving the best tracking success rate and the lowest tracking error. Our FSQ-based tracking policy attains a comparable success rate to the continuous trackers but exhibits higher MPJPE.

Method	Type	End2end	Succ. (%)	MPJPE (mm)
PDP	Continuous	No	98.90	37.32
FPO	Continuous	Yes	96.40	41.98
GPC	Discrete	Yes	99.51	44.43
MLP	Continuous	Yes	**99.59**	**30.26**

B.5. Comparison vs Diffusion-based Tracking Policy

Diffusion and flow matching models have demonstrated success in high-dimensional generative modeling [Ho et al. 2020; Lipman et al. 2023], prompting recent efforts to leverage their expressiveness for tracking-based control. PDP uses distillation with an MLP expert policy [Ross et al. 2011], to a DDPM [Ho et al. 2020], whereas FPO trains a flow matching model end-to-end from scratch via PPO.

As shown in Table 7, we benchmark our FSQ-based tracking controller against these continuous baselines, alongside a standard MLP policy. The MLP baseline remains the strongest overall, attaining both the highest success rate (99.59%) and the lowest tracking error (30.26 mm), which suggests that the added expressiveness of generative trackers does not directly translate into tighter tracking under a pure imitation objective. Among the generative approaches, our FSQ controller achieves the highest success rate (99.51%), surpassing PDP and the end-to-end FPO baseline, while incurring a modest increase in MPJPE relative to the continuous trackers. We attribute this gap to the inherent quantization error of discrete action representations, which trades off fine-grained precision for a structured action space. However, our discrete formulation provides a natural interface to the next-token prediction paradigm, enabling seamless integration with downstream autoregressive models in a way that continuous policies cannot easily support.

B.6. Probing FSQ Skill Latent Space

Since FSQ rounds each latent channel directly to a small set of fixed scalar levels during quantization, we hypothesize that the FSQ encoder can map discrete skill latents with similar skill patterns to nearby regions on the manifold. To test this property, we design a skill retrieval experiment on a subset extracted from Bones dataset, with motion categories defined at two granularities derived from file names: (1) a coarse level with 8 classes (*walk, jog, jump, kick, punch, crawl, idle, dance*), covering 6,469 motions; and (2) a fine level with 20 classes that further distinguishes locomotion styles (e.g., *tired_walk, angry_walk, zombie_walk, happy_jog*), covering 3,498 motions. The two-level granularity allows us to assess whether the skill latent captures the details of both broad action categories and subtle stylistic variation. To obtain a global representation for each motion clip, we adopt a mean-pooling strategy following Sentence-BERT [Reimers and Gurevych 2019]. We pass every frame of a motion sequence through the encoder of a pretrained FSQ tracking policy and average the resulting quantized codes across all timesteps, yielding a single mean-pooled FSQ embedding that summarizes the entire clip. We use L_2 distance as the retrieval metric, as cosine similarity would discard the magnitude information carried by the discrete FSQ skill latents. We evaluate with two metrics following the evaluation protocol used for text embeddings in the Massive Text Embedding Benchmark (MTEB) [Muennighoff et al. 2023]:

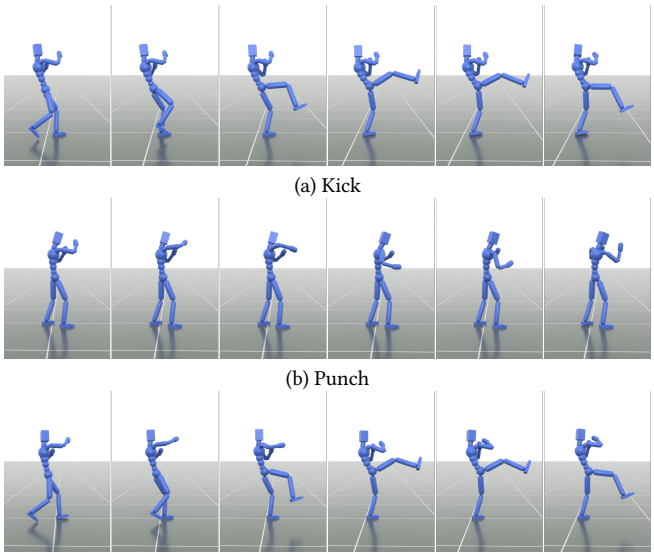
- **Nearest-Neighbor Retrieval.** For each motion, we retrieve the top-K nearest neighbors and report Precision@K, Mean Reciprocal Rank (MRR), and Mean Average Precision at depth 10 (MAP@10).
- **Pair Discrimination.** We sample 10,000 motion pairs, balanced 50/50 between same-category and cross-category pairs, and evaluate whether the similarity metric can distinguish the two groups. We report Spearman correlation, AUC-ROC, and accuracy at the optimal threshold.

Table 8 summarizes the results. At the coarse level, the skill latents achieve a Precision@1 of 0.72 and MRR of 0.81, meaning the nearest neighbor in FSQ latent space is the correct motion type 72% of the time, and the first correct match appears near the top of the ranked list on average. At the fine-grained level (20 classes), retrieval performance naturally decreases (P@1=0.66, MRR=0.76), as the model must distinguish between closely related styles such as *tired_walk* and *scared_walk*. Pair discrimination, however, improves (AUC-ROC=0.69, Spearman=0.33), suggesting that fine-grained classes are more internally homogeneous than coarse ones.

We analyze the learned representation structure by visualizing t-SNE projections of the FSQ encoder's embeddings (Fig. 15) [van der Maaten and Hinton 2008]. Cosine similarity is calculated as a distance metric for the final hidden-layer embeddings and pre-discretization FSQ latents, as shown in Figure 15 (a) and (b). Euclidean distance is utilized for post-discretization FSQ latents, shown in Figure 15 (c). Across all three plots, the representations exhibit clustering effect based on distinct characteristics of motions. The proximity of similar locomotion styles within the latent space suggests that the FSQ bottleneck produces a structured geometry. This shows that the model is learning meaningful relationships between skills, rather than assigning tokens at random. Beyond clustering, FSQ latent codes support skill composition. As shown in Fig. 14, interpolating between 'punching' and 'kicking' produces a motion that performs both strikes.

Table 8. FSQ latent retrieval evaluation. Higher is better for all metrics.

Metric	Coarse (8 cls)	Fine (20 cls)
Retrieval (leave-one-out)		
Precision@1	0.72	0.66
Precision@5	0.63	0.53
Precision@10	0.58	0.46
MRR	0.81	0.76
MAP@10	0.48	0.37
Pair Discrimination		
Spearman	0.26	0.33
AUC-ROC	0.65	0.69
Accuracy	0.62	0.65



(a) Kick

(b) Punch

(c) Punch while kicking

Fig. 14. Composited skill achieved by framewise code blending.

C. Generative Controller

C.1. Architecture and Training Configuration

The architecture of our generative controller is built around a causal transformer. It receives the current state as a context token, projected through MLP-based state encoder layers, and autoregressively predicts N_{token} discrete tokens that represent the quantized latent of skills. System architecture and hyperparameters are reported in Table 10. We implement the generative controller using PyTorch's official transformer module [Paszke et al. 2019], with GELU activations [Hendrycks and Gimpel 2023], N_{heads} attention heads, and N_{layers} transformer encoder layers. Causal masking is applied to enable teacher forcing during training. The training objective is a cross-entropy loss with label smoothing [Müller et al. 2020], which

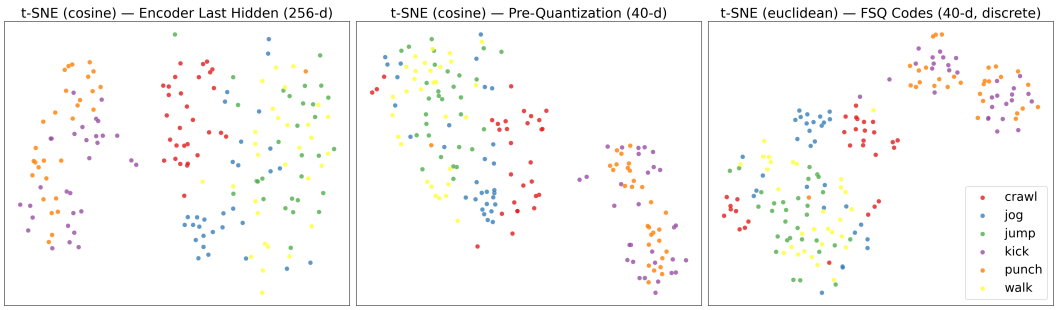


Fig. 15. t-SNE visualization of FSQ embeddings at various network depths. Note that the pre-quantization FSQ codes and the output embeddings of the last hidden layer are clustered using cosine distance, while the discretized FSQ codes are clustered with a Euclidean distance metric.

discourages overconfident predictions across the 9^5 possible token combinations. To further stabilize the training process and improve generalization, we maintain an Exponential Moving Average (EMA) of the model weights with a smoothing factor of 0.9. At inference time, we employ autoregressive prediction and use Nucleus Sampling with top-p = 0.9.

C.2. Connection to Transformer-based RL and Control Sequence Modeling

Transformer-based offline RL architectures, notably Decision Transformer and Trajectory Transformer [Chen et al. 2021; Janner et al. 2021], circumvent traditional step-wise policy optimization by trajectory modeling. Those approaches treat states, actions, and returns, as a joint sequence prediction task. Decision Transformer operates on continuous embeddings, while Trajectory Transformer discretizes states and actions through per-dimension binning. GPC shares the transformer backbone with those methods but differs from this line of work in three key respects. First, skills are tokenized via FSQ into a discrete vocabulary, rather than through the per-dimension binning of Trajectory Transformer or the continuous embeddings of Decision Transformer. Second, the backbone adopts a GPT-style autoregressive formulation [Radford et al. 2018, 2019], trained purely via next-token prediction over this compact discrete vocabulary. Third, GPC operates in the online reinforcement learning regime and requires a simulator for training, in contrast to the offline setting of Decision Transformer and Trajectory Transformer.

In the online regime, recent studies have utilized transformers as control policies [Pan et al. 2025]. TokenHSI unifies multiple skills in a single transformer by tokenizing per-task goals as continuous embeddings, and trains the network as a deterministic policy with an AMP-style adversarial reward [Peng et al. 2021]. While TokenHSI also supports task adaptation, it transfers from a fixed set of pretrained tasks to new ones. GPC instead trains a generative prior unconditionally on a large-scale motion dataset, and downstream tasks are addressed by selecting skills from this learned latent space.

C.3. GPC vs. Continuous Priors: Unconditional Sampling

Our study evaluates unconditional generation quality by training GPC and CVAE based on an FSQ tracking controller that is pretrained on the AMASS PHC train subset. We assess their performance across a comprehensive suite of metrics. Normalized Jerk is computed over 0.4s sliding windows of rigid-body positions following the formulation by Hogan and Sternad [2009]. We report a whole-body and a foot-only variant for normalized jerk, and the mean acceleration magnitude (m/s^2) to further evaluate the degree of jitters. Diversity is evaluated using the average pairwise distance (APD), computed as the mean L_2 distance across all sample pairs for each starting pose. We compute APD over full episodes on both root xy-trajectories (APD Root) and full-body pose trajectories (APD Pose). Survival rate is a robustness metric representing the proportion of episodes in which the character concludes the simulation with a root height above 0.5 m, demonstrating it is not lying on the ground at the end, even if falls occurred earlier in the duration.

Table 9 compares GPC and CVAE unconditional priors on motion quality, diversity, and robustness. Both models are evaluated from 15 starting poses with 32 parallel rollouts each, running for 20 s without resets. We test three conditions: no external force, a moderate push impulse (2.4 m/s), and a strong push impulse (9.8 m/s), all applied at step 200 with identical, seeded force directions across methods.

Without perturbation, GPC achieves lower normalized jerk and lower mean acceleration than the CVAE prior, while foot-level jerk remains relatively higher, indicating more foot jitter behavior. GPC exhibits higher trajectory diversity in both root paths and full-body poses. Survival rates are similar, indicating that both priors maintain physically plausible behavior over extended horizons. The robustness gap becomes pronounced under external forces. With a moderate push, GPC retains a 68% survival rate versus 44% for CVAE. Under the strong push, the survival rate is still above 50%, while CVAE collapses to 3.1%. Characters controlled by GPC frequently recover after being knocked down, whereas those using CVAE rarely do. In the absence of perturbations, CVAE demonstrates better performance with respect to foot jerk. Although GPC also exhibits a sharper increase in foot jerk, this likely reflects the aggressive, corrective foot placements required during recovery. Overall, GPC offers a favorable combination of motion quality, diversity, and robustness compared to the CVAE prior.

D. Task Adaptation

D.1. Locomotion Tasks

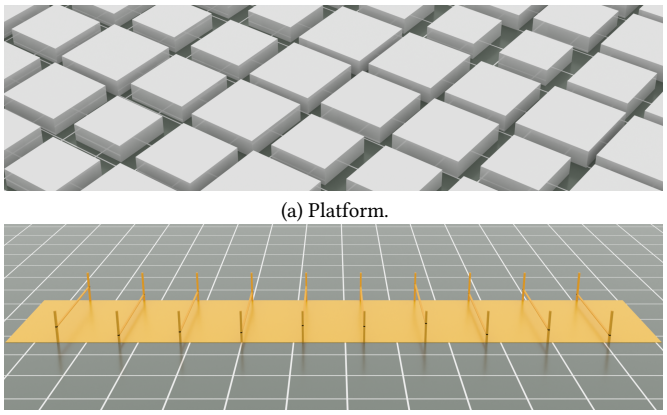
In the **Target Reaching** task, the character is required to navigate toward a specified 2D goal location in the environment. The

Table 9. In a comparison against CVAE across 15 starting poses with 32 rollouts each (20s duration), GPC demonstrates superior performance in both motion quality and diversity in the absence of external perturbations. It achieves lower Jerk and Acceleration values while simultaneously maintaining higher Root APD and Pose APD compared to the baseline. GPC exhibits greater robustness than the baseline, consistently maintaining higher survival rates as external perturbations increase.

	No Force Perturbation		**Push Impulse (2.4 m/s)**		**Push Impulse (9.8 m/s)**	
Metric	**GPC**	**CVAE**	**GPC**	**CVAE**	**GPC**	**CVAE**
Norm. Jerk ↓	**657 ± 147**	1072 ± 38	**1054 ± 32**	1172 ± 32	1458 ± 14	**1406 ± 33**
Norm. Foot Jerk ↓	1002 ± 184	**989 ± 89**	1551 ± 112	**1177 ± 66**	1918 ± 64	**1389 ± 68**
Mean Accel. (m/s^2) ↓	**4.33 ± 0.46**	6.14 ± 0.21	**5.91 ± 0.19**	6.23 ± 0.23	7.72 ± 0.14	**6.51 ± 0.09**
APD Root (m) ↑	**2.66**	2.40	3.55	**3.72**	**6.89**	6.01
APD Pose (m) ↑	**13.55**	12.22	17.90	**18.72**	**34.33**	29.82
Survival Rate (%) ↑	**82.1%**	79.4%	**68.1%**	44.4%	**52.5%**	3.1%

Table 10. Network architecture and grouping configuration of the generative controller. d_{model} represents the model dimension, N_{heads} indicates the number of attention heads, N_{layers} denotes the total transformer layers, and d_{ff} refers to the feed-forward network dimension.

Component	**Parameter**	**Value**
Transformer	d_{model}	1024
	N_{heads}	4
	N_{layers}	6
	d_{ff}	4096
	Activation	GELU
Token Grouping	G	5
	N_{token}	8
	Vocab size ($\lvert\mathcal{V}\rvert$)	$9^5 = 59,049$
Positional Encoding	Type	Learned



(a) Platform.

(b) Barrier.

Fig. 16. Scenes of HSI tasks.

task observation consists of a 2D target position expressed in the character's local coordinate frame, enabling the policy to remain invariant to global orientation. The target position is periodically reset to a random location every τ_{reset} seconds. This setup requires the controller to continuously replan and adapt its motion in response to changing goals, testing both responsiveness and stability under dynamic conditions. We reward proximity to a target location $\mathbf{p}^* \in \mathbb{R}^2$:

$$r_{\text{target}} = \begin{cases} 1 & \text{if } \|\mathbf{p} - \mathbf{p}^*\|_2 < \epsilon_{\text{prox}} \\ \exp\left(-\alpha_{\text{pos}} \|\mathbf{p} - \mathbf{p}^*\|_2\right) & \text{otherwise} \end{cases} \tag{10}$$

where $\mathbf{p}$ is the root position projected onto the ground plane, ϵ_{prox} is the proximity threshold, and α_{pos} controls the reward sharpness.

In the **Joystick Steering** task, we evaluate the controller's ability to simultaneously track a desired velocity and align with a specified facing direction. The observation includes a 2D target velocity vector along with a target heading direction, both defined in the character's local frame. At each timestep, the controller must generate actions that achieve the desired velocity while also orienting the body toward the commanded direction. This task introduces a coordination challenge between locomotion and orientation control, requiring the policy to balance multiple objectives in a temporally consistent manner. For velocity-conditioned locomotion, the agent receives a target heading direction $\hat{\mathbf{d}} \in \mathbb{R}^2$, target speed v^*, and target facing direction $\hat{\mathbf{f}} \in \mathbb{R}^2$. The reward combines directional velocity matching with heading alignment:

$$r_{\text{steer}} = w_{\text{dir}} \cdot r_{\text{dir}} + w_{\text{face}} \cdot r_{\text{face}} \tag{11}$$

where $w_{\text{dir}} = 0.7$ and $w_{\text{face}} = 0.3$. The directional reward penalizes both speed error and tangential drift:

$$r_{\text{dir}} = \begin{cases} 0 & \text{if } v_{\parallel} \le 0 \\ \exp\left(-\alpha_v \left[(v^* - v_{\parallel})^2 + \beta \|\mathbf{v}_{\perp}\|_2^2\right]\right) & \text{otherwise} \end{cases} \tag{12}$$

where $v_{\parallel} = \dot{\mathbf{p}} \cdot \hat{\mathbf{d}}$ is the velocity component along the target direction, $\mathbf{v}_{\perp} = \dot{\mathbf{p}} - v_{\parallel}\hat{\mathbf{d}}$ is the tangential velocity. The facing reward encourages alignment between the robot's heading $\hat{\mathbf{h}}$ and the target facing direction:

$$r_{\text{face}} = \max\left(0, \hat{\mathbf{h}} \cdot \hat{\mathbf{f}}\right) \tag{13}$$

In the **Trajectory Following** task, the controller must follow a predefined 2D path specified by a sequence of future waypoints. The observation provides the next 10 waypoints that span a 5-second horizon, giving the controller a short-term preview of the desired trajectory and requiring it to plan ahead. Success in this task demonstrates the model's ability to generate coherent motion by selecting motor skills, represented as discrete tokens, that align with path

Table 11. Task reward hyperparameters.

Task	Parameter	Symbol	Value
Target	Position scale	α_{pos}	0.42
	Proximity threshold	ϵ_{prox}	0.5 m
	Reset timer	τ_{reset}	6–8s
Joystick	Velocity scale	α_v	0.25
	Tangent weight	β	0.1
	Direction weight	w_{dir}	0.7
	Facing weight	w_{face}	0.3
Trajectory	Position Scale	α_{traj}	2.0

constraints. The adapted controller follows the target trajectory accurately while maintaining naturalistic behavior. Given a target position $\mathbf{p}^* \in \mathbb{R}^2$, the reward is as follows,

$$r_{\text{path}} = \exp\left(-\alpha_{\text{traj}} \|\mathbf{p} - \mathbf{p}^*\|_2\right) \tag{14}$$

$\mathbf{p}$ is the root position in the xy plane, α_{traj} is the scale of the proximity reward term. Reward hyperparameters are summarized in Table 11.

D.2. Human-Scene Interaction Tasks

We further introduce two Human-Scene Interaction (HSI) tasks to evaluate the controller in environments that require rich interaction with surrounding geometry. We train our generative controller with a mixed dataset of LAFAN1, Beyond, and Bones. These tasks, including Barrier and Platform, are formulated as target-reaching problems but involve structured scene elements that constrain feasible motion. To represent terrain observations, we use square heightmaps, from which terrain features are extracted via a lightweight 2D CNN encoder. Unlike standard locomotion settings, these tasks require the controller to avoid collisions and adapt its movement dynamically to the surrounding environment. Since the pretrained generative policy already encodes the dataset's skill distribution, a minimal proximity-based reward suffices across all human–scene interaction tasks. The reward is as follows,

$$r_{\text{target}} = \exp\left(-\alpha_{\text{pos}} \|\mathbf{p} - \mathbf{p}^*\|_2\right) \tag{15}$$

This reduces reward design across the entire task suite to a basic target-reaching objective, eliminating the need for complex task-specific reward shaping.

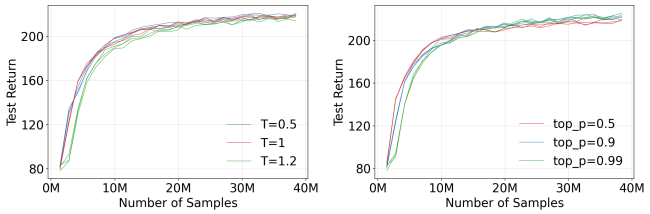
In the **Platform** task, the character must traverse a terrain of elevated platforms to reach the target location. We consider two terrain variants: a regular 2×2 grid of evenly spaced platforms (*platform-2x2*), and a larger, randomly generated layout (*platform-giant*), shown in Figure 16 (a). Platform sizes and inter-platform gaps in *platform-giant* are randomized, requiring the controller to adjust step length, timing, and balance to land safely on discontinuous surfaces. For the platform-giant terrain, gap widths are sampled uniformly between 0.7m and 1.7m. To introduce vertical complexity, neighboring platforms may differ in height by up to 0.3m. To ensure task feasibility and focus on local navigation, the target is spawned on a platform in the immediate vicinity of the character. In the **Barrier** task, the character encounters obstacles of varying heights along the path to the target. Barrier heights are randomized, and a gap is left beneath each obstacle, so the controller must invoke a distinct set of skills, ducking or crawling under each barrier, to pass through. This task evaluates the model's ability to adapt its motion to strict vertical constraints. Overall, these HSI tasks introduce scene constraints. Despite this increased complexity, our adapted controller retains the diverse and robust skills learned during generative pretraining while effectively handling scene constraints.

Table 12. Task Adaptation Configuration

Category	Parameter	Value
Adapter	Rank (r)	64
	Scaling (α)	128
Rollout	Top-p	0.9
	Temperature (T)	1.0

D.3. Task Adaptation Implementation

CoLA applies FiLM adaptation to DoRA low-rank matrices, making the adapter conditioning-dependent. The low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ have rank r and scaling factor α; $\mathbf{B}$ is zero-initialized so that the adapted model coincides with the frozen generative controller at the start of training. At rollout, we restrict the action support to the top-p mass of the frozen controller's output distribution, renormalize the adapted controller's distribution on this support, and draw a sample. Hyperparameters are reported in Table 12.



(a) Temperature sweep (b) Nucleus threshold sweep

Fig. 17. Return Curve under different Nucleus Sampling configurations during task adaptation. (a) Varying the sampling temperature T with a fixed nucleus threshold $p = 0.9$. (b) Varying the nucleus threshold p with a fixed temperature $T = 1.0$.

D.4. Ablation of Nucleus Sampling Configuration

We conduct an ablation study to evaluate how Nucleus Sampling configurations during task adaptation influence downstream performance. Using a target-reaching task, we compared a pair of default settings ($p = 0.9, T = 1.0$) against variations in temperature ($T \in \{0.5, 1.2\}$) and nucleus thresholds ($p \in \{0.5, 0.99\}$). As illustrated in Figure 17 (a), variations in temperature at a fixed $p = 0.9$ yielded broadly similar quantitative results. However, higher temperatures qualitatively degraded control by flattening the output distribution. This increased the probability of sampling low-probability codes, resulting in persistent jitter, with $T = 1.2$ performing worse than the default setting in qualitative results. Regarding threshold

Table 13. Comparison of GPC, CVAE, and MaskedMimic on success rate and final target distance, evaluated over 256 episodes.

Method	Success Rate (%) ↑	Final Target Dist. (m) ↓
MaskedMimic	89.92	0.44
CVAE	93.89	0.41
GPC (Ours)	**94.20**	**0.37**

variations in Figure 17 (b), settings of $p \geq 0.9$ achieved higher rewards than lower values. While low p configurations converged faster due to a restricted action space, they ultimately limited the agent's repertoire, suggesting that a narrower selection of codes constrains the essential skill sets required for optimal adaptive performance.

D.5. Comparison vs Continuous Priors: Downstream Tasks

We evaluate GPC against continuous-prior baselines on a goal-reaching downstream task, comparing to the CVAE-based priors of PULSE [Luo et al. 2023] and MaskedMimic [Tessler et al. 2024]. GPC and PULSE share the same downstream training schedule, where a task policy is trained with RL on top of a frozen prior. MaskedMimic is trained end-to-end with supervised learning conditioned on randomly masked future joint positions, and at inference is provided with target root positions at a fixed height. As shown in Table 13, GPC and CVAE achieve comparable success rates (0.942 and 0.939, respectively), while MaskedMimic lags behind at 0.899. GPC further achieves the lowest final target distance (0.37m) among the three methods. The performance gap of MaskedMimic stems primarily from a mismatch between the goal-reaching task setup, which conditions on a fixed root height, and its training distribution, in which no real motion exhibits a constant root height. This distributional shift induces unnatural behaviors that hinder task performance.