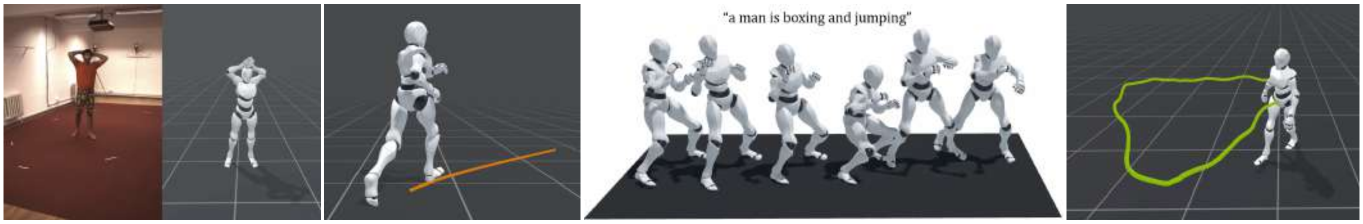
arXiv:2310.10198v3 [cs.CV] 19 Dec 2023

MoConVQ: Unified Physics-Based Motion Control via Scalable Discrete Representations

HEYUAN YAO, ZHENHUA SONG, YUYANG ZHOU, and TENGLONG AO, Peking University, China
BAOQUAN CHEN and LIBIN LIU*, Peking University, China and National Key Lab of General AI, China



(a) Motion Tracking (b) Interactive Control (c) Text2Motion (d) LLM Integration

Fig. 1. We present a method for learning discrete motion representation from a large-scale unstructured motion dataset for physics-based characters. The framework allows various applications, including those shown in this figure, to be accomplished in a unified fashion.

In this work, we present MoConVQ, a novel unified framework for physics-based motion control leveraging scalable discrete representations. Building upon vector quantized variational autoencoders (VQ-VAE) and model-based reinforcement learning, our approach effectively learns motion embeddings from a large, unstructured dataset spanning tens of hours of motion examples. The resultant motion representation not only captures diverse motion skills but also offers a robust and intuitive interface for various applications. We demonstrate the versatility of MoConVQ through several applications: universal tracking control from various motion sources, interactive character control with latent motion representations using supervised learning, physics-based motion generation from natural language descriptions using the GPT framework, and, most interestingly, seamless integration with large language models (LLMs) with in-context learning to tackle complex and abstract tasks.

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

Additional Key Words and Phrases: motion control, deep reinforcement learning, generative model, VQ-VAE, large language model

1 INTRODUCTION

Physics-based motion control allows a character to interact with a physically simulated environment. It has the potential to generate realistic character animation and provide natural responses to environmental changes and perturbations. Such abilities are crucial for digital humans and embodied intelligent agents. However, due to the complexity of full-body human dynamics, realizing control over diverse and agile human motion has been a longstanding challenge.

Many recent studies in this field draw inspiration from deep learning and generative models. They combine various generative models with deep reinforcement learning (DRL), resulting in generative, physics-based motion representations [Peng et al. 2022; Won et al. 2022; Yao et al. 2022]. When trained on a set of example motions, these generative control strategies effectively capture the motion skills within that dataset, and allow downstream tasks to be effectively completed using the latent representations. Despite these successes, there are still notable limitations in these approaches. For instance, the datasets used in these studies typically comprise only several minutes or tens of minutes of data with limited motion diversity. It is nontrivial to scale up a DRL framework to accommodate hours or even more diverse motion data. The latent motion representation often lacks explicit meaning, making it challenging to select desired motions from the latent space. To utilize a specific motion, a dedicated reward function is often required to guide a policy toward that particular movement, and extra DRL training is often needed to train the policy.

In this paper, we attempt to address these problems. Particularly, we focus on letting a physics-based character grasp a diverse set of motion skills from a large dataset of tens of hours of example motions, and providing a unified and intuitive interface for various downstream tasks. To this end, we address two primary questions: (a) what is an effective motion representation for controlling a diverse range of motions? and (b) how to efficiently learn such a representation from a large volume of data?

For the representation of motion, we draw inspiration from recent studies that utilize discrete, or *vector quantized*, latent representations, which have achieved high-quality generation in domains such as images [Rombach et al. 2022], music [Dhariwal et al. 2020], and motions [Ao et al. 2023; Jiang et al. 2023]. We formulate our generative motion control using a vector quantized variational autoencoder (VQ-VAE) framework. We argue that this discrete motion control representation is not only capable of accommodating a diverse range of motions but also serves as a robust and intuitive interface for downstream applications.

For training the framework, we note that several recent works [Fussell et al. 2021; Yao et al. 2022] have utilized model-based reinforcement learning techniques to achieve efficient learning of complex human actions, particularly generative control policies from a

---

[*] corresponding author

Authors' addresses: Heyuan Yao, heyuanyao@pku.edu.cn; Zhenhua Song, songzhenhua@stu.pku.edu.cn; Yuyang Zhou, yuyangzhou2002@gmail.com; Tenglong Ao, aubrey.tenglong.ao@gmail.com, Peking University, Beijing, China; Baoquan Chen, baoquan@pku.edu.cn; Libin Liu, libin.liu@pku.edu.cn, Peking University, Beijing, China and National Key Lab of General AI, Beijing, China.

set of motions [Yao et al. 2022]. These methods learn a *world model* to approximate the black-box rigid body dynamics, which offers a differentiable path that allows the gradient of the training objective to directly update the control model. We believe that this property of model-based RL makes it suitable for training large, complex generative neural networks. Thus, we adopt a similar model-based RL approach for training our framework, which in practice allows for efficient training on datasets spanning several tens of hours of motion examples

We refer to this scalable framework as *MoConVQ*. It serves as a unified motion control framework, which we demonstrate with several applications including:

- Acting as a universal tracking controller to track motions from various sources, such as kinematic motion synthesis and monocular 3D human pose estimation;
- Serving as a versatile latent space within which a high-level policy can be trained using supervised learning for interactive control of the character;
- Functioning as a generative motion prior when integrated with the GPT (Generative Pretrained Transformer) framework, supporting high-quality physics-based motion generation from natural language descriptions; and
- Providing an efficient interface for integration with Large Language Models (LLMs), which enables an LLM to learn to use the motions via in-context learning.

In summary, our contribution includes:

- We propose a unified framework for physics-based motion generation that can scale up to a very large dataset. This framework generates high-quality physics-based motions and can generalize across a broad range of motions. To the best of our knowledge, this is the first physics-based generative motion control model trained on a dataset of such size and diversity.
- We showcase several essential tasks and applications using our framework. Notably, our text-conditional GPT generator, termed *T2M-MoConGPT*, is the first physics-based approach to address such a general text-to-motion synthesis task.
- We introduce a novel integration with large language models that allows an LLM to understand human motions and use them in a simulated environment. We believe this opens up the possibility of creating intelligent simulated avatars through the aid of LLMs, harnessing their advanced reasoning capabilities to tackle complex and abstract tasks.

2 RELATED WORK

Our work spans several domains, including skeleton animation, reinforcement learning, and generative models. We review the most relevant works, with a particular focus on physics-based motion generation models. For a more in-depth understanding, we direct readers to the surveys presented by Kwiatkowski et al. [2022] and Mourot et al. [2022].

2.1 Physics-Based Motion Controllers

Generating realistic and interactive locomotion for simulated characters is a longstanding challenge in computer animation. Early research has been focus on designing motion controllers based on domain knowledge, such as leveraging abstract models to assist the character in maintaining balance, enabling stable walking [Coros et al. 2008; Lee et al. 2010; Yin et al. 2007] and stair-stepping motions [Coros et al. 2010; Yin et al. 2008]. Spacetime optimization and reinforcement learning methods are then introduced to generate more complex motions [Levine and Koltun 2013; Mordatch et al. 2012; Yin et al. 2021]. These methods typically rely on carefully designed objective functions or hand-tuned parameters, and thus can be hard to apply to complex scenarios and general motion generation tasks.

In contrast, data-driven methods generate natural motions by imitating captured human performances in simulated environments [Fussell et al. 2021; Liu et al. 2016, 2015; Peng et al. 2018; Wang et al. 2020]. The use of motion capture data reduces the need to design specific objective functions for these methods. Instead, many of these approaches utilize a unified imitation error as their optimization target. By introducing perturbations and altering goals during training, the learned controllers can adapt to external disturbances, changes in target motions [Peng et al. 2018; Xie et al. 2022], and even variations in body shapes [Luo et al. 2023; Won and Lee 2019]. Building on these successes, some works attempt to generate simulated motions by tracking the results of kinematic motion generators [Bergamin et al. 2019; Wang et al. 2020; Won et al. 2020; Zhang et al. 2023a]. However, producing physically plausible animations using kinematic methods is challenging. The physical errors in the generated motions can degrade the performance of the tracking controller, making a native physics-based generative model more preferable.

2.2 Generative Models in Physics-Based Motion Control

In recent years, researchers have started to investigate the use of generative models within physics-based character animation, aiming to create diverse and flexible movements. Although studies in kinematic motion synthesis have shown the effectiveness of several generative models [Ao et al. 2023; Henter et al. 2020; Li et al. 2022a; Ling et al. 2020], this experience cannot be directly applied to physics-based motion control, because physics constraints are often considered as a black box and non-differentiable. A possible workaround is to learn motion primitives and reuse them in downstream tasks [Liu and Hodgins 2017; Peng et al. 2019; Won et al. 2020]. These motion primitives can also be distilled into a larger generative policy using behavior cloning [Merel et al. 2018, 2020], though the learned policy can suffer from distribution shift without feedback from actual simulation. More recent works draw inspiration from GAIL, or Generative Adversarial Imitation Learning [Ho and Ermon 2016], and employ a discriminator as a motion prior to regulate the style of the character's motion [Bae et al. 2023; Peng et al. 2022, 2021; Tessler et al. 2023; Xu and Karamouzas 2021; Xu et al. 2023]. This approach allows the character to complete complex tasks similar to example motions, such as interacting with scenes [Hassan et al. 2023], objects [Bae et al. 2023], and other characters [Younes et al. 2023]. However, the mode collapse issue makes it challenging to handle large-scale datasets [Peng et al. 2022].

Our approach is built upon variational autoencoders (VAE) [Kingma and Welling 2014]. The works most closely related to ours are [Won et al. 2022; Yao et al. 2022], which also utilize a world model [Ha and Schmidhuber 2018] serving as differentiable simulator and reduce the task of physics-based motion generation to learning a variational autoencoder. However, both of these works capture only a few minutes of motion data. The selection of motion style in these methods is achieved by training another high-level policy on selected motion clips. In terms of integration with a natural language interface, Juravsky et al. [2022] also demonstrate several language-directed controllers for physics-based character animation. However, their datasets are also relatively small and necessitate a more meticulous segmentation into short motion clips. The language models in their approach are primarily employed to select the subsequent motion clip to execute, rather than facilitating generalized cross-modal generation. Unlike these previous works, our work is dedicated to enabling the model to learn diverse, task-agnostic motion encodings on a substantial unstructured dataset. Due to the abundance of motion encodings, we are able to learn a mapping from broader control tasks to physics-based motions.

2.3 Motion Generation Tasks and Motion Representations

Beyond the scope of physics-based character animation, our research is enriched by insights from the kinematic motion generation field, particularly from areas like music-to-dance generation [Alexanderson et al. 2023], text-driven animations [Petrovich et al. 2021, 2022; Tevet et al. 2023; Xin et al. 2023], and gesture synthesis [Ao et al. 2023; Li et al. 2022b]. Given the absence of physical constraints, these domains readily integrate with contemporary generative models, resulting in rapid advancements.

The initial efforts are centered on learning a generative model directly in the motion space. These generative models include VAE [Petrovich et al. 2021, 2022], GAN [Ahn et al. 2018] and flows [Alexanderson et al. 2020; Ye et al. 2022]. Recently, Tevet et al. [2023]; Yuan et al. [2022]; Zhang et al. [2022] apply the diffusion model [Sohl-Dickstein et al. 2015] to text-driven motion generation tasks and achieve notable success. Similarly, Alexanderson et al. [2023] utilizes the diffusion model to generate dance sequences and gestures, using audio and text as direct inputs. Due to the challenges of redundancy and noise in generating within the original motion space, many subsequent studies transition to a two-stage methodology, which first encode data into a latent space and afterwards learn a probabilistic model to generate the encodings [Guo et al. 2022a; Li et al. 2022b; Xin et al. 2023]. An optimal representation thus becomes essential for this line of research. Evidence from several recent studies indicates that VQ-VAE [van den Oord et al. 2017] provides an effective and compact motion representation [Dhariwal et al. 2020; Geng et al. 2023; Rombach et al. 2022]. As a result, many researchers opt to construct generative models within the VQ-VAE space [Ao et al. 2022, 2023; Dhariwal et al. 2020]. Additionally, leveraging the discrete nature of VQ-VAE, generative models from natural language processing (NLP), such as Generative Pretrained Transformer [Brown et al. 2020], can be combined with motion models to achieve impressive results in cross-modal generation tasks [Guo et al. 2022b; Jiang et al. 2023; Li et al. 2022b; Zhang et al. 2023b].

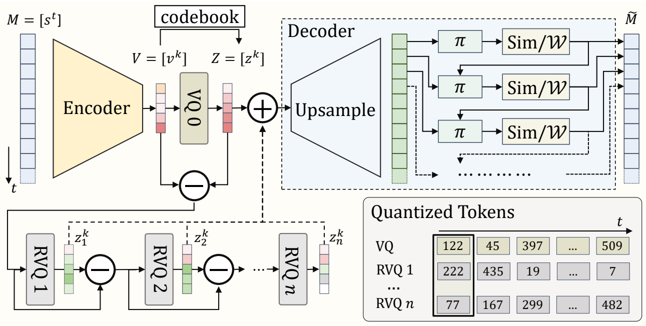


Fig. 2. Framework overview. Our MoConVQ system consists of a motion encoder, a physics-based decoder, and a series of codebooks. A residual architecture is adopted to enhance the system's representational capacity. The system represents motion as a sequence of quantized latent codes. Each latent code combines the quantized vectors from all residual layers. Equivalently, it can be represented by the indices of these quantized vectors in the codebooks of their corresponding residual layers.

The advancements in kinematic motion generative models have demonstrated the benefits of training compact latent representations on large datasets and applying these representations to various downstream tasks. Inspired by these applications, especially those involving NLP models, we combined our motion representation with GPT, leading to physics-based text-to-motion generation, a downstream task that previous methods had not accomplished. Additionally, we showcase a practical integration with large language models.

3 DISCRETE MOTION REPRESENTATION

3.1 MoConVQ Framework

Our goal is to find a compact motion representation that captures the essential characteristics of a motion, enabling a character to reproduce it in a physics-based simulation environment. We utilize the Vector Quantized Variational Autoencoders (VQ-VAE) [van den Oord et al. 2017] to learn this representation from a large motion dataset encompassing a wide variety of motions. Figure 2 illustrates the structure of our framework, which we refer to as MoConVQ in this paper.

Formally, a MoConVQ model consists of an encoder, $\mathcal{E}$, a decoder, $\mathcal{D}$, and a discrete codebook, represented as $\mathcal{B}$. The encoder $\mathcal{E}$ and the decoder $\mathcal{D}$ establish a bidirectional mapping between a motion clip, $\boldsymbol{M}$, and its corresponding latent representation. The codebook $\mathcal{B}$ consists of a list of learnable latent codes $\{z_i\}_{i=1}^{N_{\mathcal{B}}}, z_i \in \mathcal{Z}$, which is used to quantize the latent representation. We represent these procedures as

$$\boldsymbol{V} = \mathcal{E}(\boldsymbol{M}), \quad \boldsymbol{Z} = \mathcal{B}(\boldsymbol{V}), \quad \tilde{\boldsymbol{M}} = \mathcal{D}(\boldsymbol{Z}). \tag{1}$$

The motion clip $\boldsymbol{M}$ consists of a sequence of poses $[\boldsymbol{s}^t]$. It is encoded by $\mathcal{E}$ into a sequence of latent vectors $\boldsymbol{V} = [\boldsymbol{v}^k]$. The code sequence $\boldsymbol{Z} = [\boldsymbol{z}^k]$ is the quantized version of $\boldsymbol{V}$. It is calculated by finding the nearest neighbor for each $\boldsymbol{v}^k \in \boldsymbol{V}$ within the codebook $\mathcal{B}$:

$$z^k, I^k = \arg\min_{z_i \in \mathcal{B}} \|z_i - v^k\|_2^2, \tag{2}$$

where I^k is the index of $\boldsymbol{z}^k$ in $\mathcal{B}$. Notably, the quantized sequence $\boldsymbol{Z}$ can be represented using its corresponding indices, denoted as $S = [I^k]$. Finally, $\boldsymbol{Z}$, or equivalently, S, is decoded by $\mathcal{D}$ through a simulation process to reproduce the motion sequence $\tilde{\boldsymbol{M}}$.

We model the encoder $\mathcal{E}$ as a 1D convolutional neural network. It effectively downsamples a motion clip $\boldsymbol{M}$ with T frames into a sequence of K latent codes at a fixed ratio. The decoder, $\mathcal{D}$, on the other hand, is more complicated, comprising a 1D deconvolutional network, a control policy, and a simulation module. As shown in Figure 2, the deconvolutional network upsamples the quantized motion representation $\boldsymbol{Z} = [\boldsymbol{z}^k]$ into a sequence of intermediate codes $[\boldsymbol{u}^t] = \texttt{Deconv}([\boldsymbol{z}^k])$,

which have the same framerate as the motion clip. Then, a policy, $\boldsymbol{\pi}$, is trained to compute an action, $\boldsymbol{a}^t$, for each code $\boldsymbol{u}^t$ based on the current state of the character, $\boldsymbol{s}^t$. The action $\boldsymbol{a}^t$ is then used to actuate the character in the simulation, advancing the character to the next state, $\boldsymbol{s}^{t+1}$. This control and simulation process can be written as

$$\boldsymbol{a}^t = \boldsymbol{\pi}(\boldsymbol{s}^t, \boldsymbol{u}^t), \quad \boldsymbol{s}^{t+1} = \texttt{Sim}(\boldsymbol{s}^t, \boldsymbol{a}^t). \tag{3}$$

It is performed autoregressively, using the simulated state from the previous step to compute the action and subsequent state. The resulting sequence of states, denoted as $[\boldsymbol{s}^t]$, assembles the reconstructed motion $\tilde{\boldsymbol{M}}$. This entire procedure, including the upsampling, control, and simulation, is collectively referred to as the decoder $\mathcal{D}$.

It is worth noting that the architectures of both the encoder and decoder allow them to process motions in a streaming manner. This means that, at runtime, the system can continuously feed new motion poses to the encoder. They are then encoded, quantized, and subsequently decoded into a sequence of simulated motion. Additionally, high-level policies for downstream tasks can generate and modify the latent codes in real-time, whether in the form of $[\boldsymbol{v}^k]$, $[\boldsymbol{z}^k]$, or $[I^k]$, creating a continuous stream of simulated motions. We will discuss several example downstream tasks in Section 4.

3.2 Residual VQ-VAE

In practice, the representational capacity of a VQ-VAE model is restricted by the size of the codebook $\mathcal{B}$. In MoConVQ, as the volume and diversity of motion data increase, the potential variations within each small segment of data also expand. From an information theory perspective, to ensure that these nuances can be distinguished, more bits have to be allocated to represent them. This leads to an exponential growth in the size of the codebook, resulting in inefficiency and unstable training due to code collapse [Dhariwal et al. 2020]. Similar facts are noted in works related to audio compression [Zeghidour et al. 2021] and image generation [Lee et al. 2022a].

Residual VQ-VAE [Lee et al. 2022a; Zeghidour et al. 2021] addresses such issues by utilizing a residual architecture with multiple quantization layers. As depicted in Figure 2, in addition to the standard vector quantization of VQ-VAE, the residual VQ-VAE employs a series of additional residual vector quantization (RVQ) layers and corresponding learnable codebooks $\{\mathcal{B}_d\}_{d=1}^N$ to model the residues. Specifically, the first RVQ layer (RVQ-1) considers the residue resulting from the VQ-VAE's quantization, given by $r_1 = \boldsymbol{v} - \boldsymbol{z}$. It then looks up the codebook $\mathcal{B}_1$ and quantizes r_1 into $\boldsymbol{z}_1$. For a subsequent RVQ layer, RVQ-d, the residue of the previous RVQ layers, defined as $r_d = \boldsymbol{v} - \sum_{j<d} \boldsymbol{z}_j$, is quantized into $\boldsymbol{z}_d$ based on the corresponding codebook $\mathcal{B}_d$. If we regard the standard VQ-VAE as a special initial RVQ layer (RVQ-0), this entire process can be viewed uniformly, as outlined in Algorithm 1. With the RVQ layers, the quantized code of each $\boldsymbol{v}^k \in \boldsymbol{V}$ is the sum of all RVQ codes, *i.e.* , $\boldsymbol{z}^k = \sum_{d=0}^N \boldsymbol{z}_d^k$, or equivalently, the tuple of indices of residual codes within each RVQ layer, $I^k = (I_0^k, I_1^k, \dots, I_N^k)$, where I_d^k is the index of $\boldsymbol{z}_d^k$ in the codebook $\mathcal{B}_d$.

ALGORITHM 1: Algorithm of Residual Vector Quantization

```
Function Residual Vector Quantization($\boldsymbol{v}$):
    // $\boldsymbol{v} \in \boldsymbol{V}$ is the continuous latent code
    // computed using Equation (1)
    $\boldsymbol{z} \leftarrow 0$
    $r \leftarrow \boldsymbol{v}$
    for $d \leftarrow 0$ to $N$ do
        $\boldsymbol{z}_d, I_d \leftarrow \arg\min_{\boldsymbol{z}_i \in \mathcal{B}_d} \|r - \boldsymbol{z}_i\|_2^2$
        $r$ -= $\boldsymbol{z}_d$
        $\boldsymbol{z}$ += $\boldsymbol{z}_d$
    end
    $I \leftarrow (I_0, I_1, \dots, I_N)$
    return $\boldsymbol{z}, I$
end
```

Compared to directly increasing the size of the codebook, the residual VQ-VAE expands the capacity of the model exponentially as the number of layers increases [Zeghidour et al. 2021]. Another benefit is that it establishes a coarse-to-fine representation of the entire latent space. Here, the coarse latents capture the primary components of the motion, while the fine-grained latents enrich it with details. For a more detailed discussion, please refer to Section 5.6.

3.3 Model-Based Learning of MoConVQ

We train our MoConVQ model on a large and diverse dataset comprising several tens of hours of motion data. An efficient and stable training algorithm is crucial to successfully handle such a volume of data. Additionally, we treat the simulation as a black box. While this allows our framework to be independent of specific physics engines, it also makes the decoder non-differentiable. This prevents the use of the standard training method for VQ-VAE, which requires the gradient to pass through the entire model.

Inspired by recent works that employ model-based reinforcement learning for complex motion controllers [Fussell et al. 2021; Yao et al. 2022], we adopt a model-based method to train our MoConVQ models. Specifically, we jointly train a network, referred to as the world model $\mathcal{W}$, to approximate the black-box simulator. The control and simulation process described by Equation (3) then becomes

$$\hat{\boldsymbol{a}}^t = \boldsymbol{\pi}(\hat{\boldsymbol{s}}^t, \boldsymbol{u}^t), \quad \hat{\boldsymbol{s}}^{t+1} = \mathcal{W}(\hat{\boldsymbol{s}}^t, \hat{\boldsymbol{a}}^t), \tag{4}$$

where the simulation operator $\texttt{Sim}$ is replaced by the world model $\mathcal{W}$. This adaptation ensures that the decoding of the reconstructed motion sequence $\tilde{\boldsymbol{M}}_{\mathcal{W}} = [\hat{\boldsymbol{s}}^t]$ is perfectly differentiable. We use $\mathcal{D}_{\mathcal{W}}$ to denote this decoding process and a hat ($\hat{\ }$) to indicates a quantity synthesized by the world model.

We train the world model and MoConVQ using an approach similar to ControlVAE [Yao et al. 2022]. In each training iteration, the world model and the components of the MoConVQ model are alternately updated until convergence. For the world model $\mathcal{W}$, we first instruct the character to track a random reference motion $\boldsymbol{M}$ using the current MoConVQ. This process involves encoding $\boldsymbol{M}$ into latent codes, quantizing them, and decoding the quantized codes in real simulation, resulting in a simulated motion sequence $\tilde{\boldsymbol{M}}_{\text{sim}}$. Then, we replay $\tilde{\boldsymbol{M}}_{\text{sim}}$ with $\mathcal{D}_{\mathcal{W}}$ and $\mathcal{W}$ to generate a synthesized sequence $\tilde{\boldsymbol{M}}_{\mathcal{W}}$. After this, the world model $\mathcal{W}$ is updated by optimizing the loss function

$$\mathcal{L}_{\mathcal{W}} = \|\tilde{\boldsymbol{M}}_{\text{sim}} - \tilde{\boldsymbol{M}}_{\mathcal{W}}\|. \tag{5}$$

Subsequently, the components of the MoConVQ model, including the encoder $\mathcal{E}$, the codebook $\mathcal{B}$, and the policy $\boldsymbol{\pi}$, are updated simultaneously. The character tracks a new training motion $\boldsymbol{M}$ with the world model $\mathcal{W}$, creating a new synthesized reconstruction $\tilde{\boldsymbol{M}}_{\mathcal{W}}$. Then, the MoConVQ model is updated by optimizing the loss function

$$\begin{aligned}\mathcal{L} = {} & \|\boldsymbol{M} - \tilde{\boldsymbol{M}}_{\mathcal{W}}\| \\ & + \beta_1\|\mathcal{E}(\boldsymbol{M}) - \mathrm{sg}(\boldsymbol{Z})\| + \beta_2\|\mathrm{sg}(\mathcal{E}(\boldsymbol{M})) - \boldsymbol{Z}\| \\ & + \beta_3\mathcal{L}_{\text{reg}} \quad ,\end{aligned} \tag{6}$$

where the first three terms correspond to the standard VQ-VAE losses [van den Oord et al. 2017], namely the reconstruction loss, the commitment loss, and the VQ objective. sg represents the stop gradient operator. To prevent the policy from generating excessively large output, which could lead to unnatural motions and cause unstable training, we introduce a regularization term, $\mathcal{L}_{\text{reg}}$, for the actions. Further details are discussed in the next section.

3.4 Implementation

State. Our simulated character is modeled as an articulated rigid body system with a floating base. Similar to the frameworks presented in [Fussell et al. 2021; Yao et al. 2022], we define the state of the character, $\boldsymbol{s}$, using the following attributes:

- Rotation r and position p of each body
- Velocity v and angular velocity ω of each body
- Height h of each body
- Up direction y_0 of the root joint

The root joint is designated to be in the character's pelvis. The attributes r, p, v, ω are represented in the local coordinate frame of the root joint.

Encoder and Decoder. We formulate the encoder $\mathcal{E}$ of MoConVQ as a 1D convolutional network. Together with the upsampling module Deconv of the decoder $\mathcal{D}$, which is also a 1D convolutional network, they form a symmetric fully convolutional network. The design and structure of this combined network are similar to those in [Dhariwal et al. 2020] and [Ao et al. 2023], which focus on generative models for time-series data such as audio [Dhariwal et al. 2020] and gesture [Ao et al. 2023].

Policy and Action. The policy $\boldsymbol{\pi}$ of the decoder $\mathcal{D}$ adopts a layer-wise mixture of experts (MoE) model as introduced by Xie et al. [2022]. This model consists of six experts, each being a four-layer MLP with 256 units, and their outcomes are mixed based on blending weights determined by a gating network, which is a two-layer MLP with 64 units.

We actuate the character using joint torques. Each joint torque is computed using the widely-adopted PD control scheme

$$\tau = k_p(\bar{\theta} - \theta) - k_d\dot{\theta}, \tag{7}$$

where θ and $\dot{\theta}$ denote the current joint rotation and rotational speed, respectively. $\bar{\theta}$ represents the desired joint rotation. k_p and k_d are the proportional and derivative gains, respectively. The output of the policy $\boldsymbol{\pi}$, *i.e.* , the action vector $\boldsymbol{a}$, thus comprises the desired joint rotations of all the joints.

Action Regularization. While the quantization of MoConVQ offers robustness to minor perturbations in training motions, flaws and physically implausible movements in the training motion data can degrade the performance of a MoConVQ model. For instance, the model might misinterpret high-frequency noise in the input motion as specific motion details, leading to oscillating movements. Similarly, for motions captured on a treadmill, the model may abuse the imperfect contact model of the physics engine and create foot-skating to mimic the stationary motion.

We notice similar phenomena have been reported by previous research, such as [Bergamin et al. 2019] and [Green et al. 2021]. Bergamin et al. [2019] use exponential moving average (EMA) with a large smoothing factor to mitigate this issue. However, in practice, we find that such a hard constraint can overly smooth the motions, eliminating many subtle motion details. Instead, we employ EMA as a soft constraint in the optimization problem of Equation (6), which becomes the regularization loss $\mathcal{L}_{\text{reg}}$. Specifically, when creating the synthesized motion sequence $\hat{\boldsymbol{M}}$, we compute the EMA of the actions as

$$\bar{\boldsymbol{a}}^t = (1 - \beta)\bar{\boldsymbol{a}}^{t-1} + \beta\hat{\boldsymbol{a}}^t, \tag{8}$$

where β represents the smoothing factor. Empirically, we find that $\beta = 0.8$ achieves reasonable visual results. $\mathcal{L}_{\text{reg}}$ is then computed as

$$\mathcal{L}_{\text{reg}} = \sum_t w_1\|\hat{\boldsymbol{a}}^t - \bar{\boldsymbol{a}}_t\| + w_2\|\hat{\boldsymbol{a}}^t\| \tag{9}$$

where the first term limits the difference between the action computed by the policy $\boldsymbol{\pi}$ and its EMA, encouraging a smooth sequence of actions. The second term regularizes the absolute magnitude of the action to enforce training stability, which is also used in [Fussell et al. 2021; Yao et al. 2022].

World Model. The differentiable world model $\mathcal{W}$ consists of a four-layer MLP with 512 units, similar to those used in previous research [Fussell et al. 2021; Won et al. 2022; Yao et al. 2022]. In practice, we employ a large first-in-first-out replay buffer to store the simulated motion sequences $\tilde{\boldsymbol{M}}_{\text{sim}}$. We then evaluate the loss function Equation (5) using a batch of sequences sampled from this buffer. This approach, suggested by [Fussell et al. 2021; Yao et al. 2022], significantly stabilizes training and is critical to its success. Our replay buffer has a size of 50000, and we replace 1024 simulated motion frames in each iteration.

Training Techniques for Residue VQ-VAE. In addition to the training procedure discussed in the previous section, we also employ several techniques proposed in previous works [Ao et al. 2022; Jiang et al. 2023; Li et al. 2022b; van den Oord et al. 2017; Zeghidour et al. 2021; Zhang et al. 2023b] to ensure a successful training. These techniques include

- **EMA Update**: Updating the codebook $\mathcal{B}$ using an exponential moving average scheme [Jiang et al. 2023; Li et al. 2022b; Zeghidour et al. 2021; Zhang et al. 2023b] formulated as

$$\mathcal{B} \leftarrow (1-\gamma)\mathcal{B} + \gamma\hat{\mathcal{B}}, \tag{10}$$

  where $\hat{\mathcal{B}}$ represents the optimized codebook and γ is the smoothing factor.
- **Code Reset**: Resetting codes that are inactive during training [Ao et al. 2023; Dhariwal et al. 2020; Zhang et al. 2023b]. This can alleviate the codebook collapse problem in VQ-VAE training.
- **Quantizer Dropout**: Using a random span of the first RVQ layers, instead of using all layers, to reconstruct the motion during training [Zeghidour et al. 2021]. The number of these RVQ layers is selected randomly from $[1, 2, \ldots, N]$ in each iteration. This approach ensures that motions can be represented using any span of the initial RVQ layers, allowing a tunable balance between representation complexity and motion reconstruction quality without needing to retrain the system.

4 APPLICATIONS AND TASKS

A trained MoConVQ model provides a foundation that enables various downstream tasks to be tackled within a unified framework. We categorize these tasks into two groups: *encoder-decoder tasks* and *decoder-only tasks*. For the *encoder-decoder tasks*, the latent motion representation is computed by the pretrained encoder, which processes motion sequences generated by other methods. We also refer to this type of task as *tracking control*. For the *decoder-only tasks*, we train a separate high-level task policy to compute the sequence of latent codes based on specific conditions. These latent codes are then decoded into motions through simulation. This *decoder-only* configuration also allows for direct integration with a large language model (LLM), enabling more complex tasks that can leverage the reasoning capabilities of the LLM. In this section, we will explore several representative applications and tasks to showcase the potential of this framework.

4.1 Universal Tracking Control

We first demonstrate the capacity of our MoConVQ framework in tracking problems, using the *encoder-decoder* configuration. Particularly, an input motion is encoded using $\mathcal{E}$, quantized according to $\mathcal{B}$, and decoded with $\mathcal{D}$ through simulation. The quantized embedding effectively eliminates disturbances in the original motion, enables imitation of unseen motions or even noisy inputs, and allows the creation of physically correct motions. We highlight the tracking capabilities of our MoConVQ by testing it on motions from three different sources, including:

- **Unseen motion**: Motion clips from a motion dataset not used in training.
- **Kinematic motion generation**: Motion clips generated by a motion synthesis method without involving simulation. Such motions can sometimes contain artifacts like foot-skating, self-penetration, or penetrating other objects.
- **Video-based pose estimation**: Motion clips estimated from a video using a pose estimation approach. These motions can often be noisy and lack physical accuracy.

We show results and analysis in Section 5.2.

4.2 Interactive Control

We further demonstrate that our MoConVQ framework is capable of incorporating a high-level policy to achieve interactive control using the *decoder-only* configuration. In this setup, the high-level policy computes latent codes directly based on the current state and task requirements. While reinforcement learning is frequently used to train such hierarchical control policy, we show that the robustness offered by the quantized motion representation enables even more efficient supervised learning.

We take *steering control* as an example, where the character is required to respond to user input and move in the desired direction and speed. We employ a simple autoregressive model as the control policy, similar to previous work such as [Holden et al. 2017]. This policy, $\boldsymbol{\pi}_{\text{steering}}$, is formulated as

$$\boldsymbol{v}^k, \boldsymbol{g}_{\pi}^{k+1} = \boldsymbol{\pi}_{\text{steering}}(\boldsymbol{z}^{k-1}, \boldsymbol{g}^k), \quad \boldsymbol{z}^k = \mathcal{B}(\boldsymbol{v}^k) \tag{11}$$

where $\boldsymbol{g}^k$ denotes the task parameters, which is computed according to the difference between the character's current state and the desired direction and speed of movement. $\boldsymbol{z}^{k-1}$ represents the quantized latent code computed from the previous control time step. The output of the policy is a new latent vector $\boldsymbol{v}^k \in \mathcal{Z}$, which is then quantized as described in Section 3. Following [Holden et al. 2017], we also let the policy predict the future task parameters, denoted by $\boldsymbol{g}_{\pi}^{k+1}$, which is used to facilitate training and improve policy performance during inference.

We train this policy by distilling the strategy embedded in a reference dataset consists of several locomotion sequences. During the training stage, we encode the entire dataset into quantized latent codes using the pretrained encoder $\mathcal{E}$, and compute the corresponding task parameters. This training dataset can then be represented as $[(\boldsymbol{z}_*^k, \boldsymbol{g}_*^k)]$. Subsequently, by unrolling Equation (11), we generate a sequence of predictions, $[(\boldsymbol{v}^k, \boldsymbol{g}_{\pi}^{k+1})]_{k=0}^{K}$ and $[\boldsymbol{z}^k]_{k=0}^{K}$. Then, the policy is updated by optimizing the loss function

$$\mathcal{L}_{\boldsymbol{\pi}} = \sum_{k=0}^{K} \|(\boldsymbol{z}_*^k, \boldsymbol{g}_*^{k+1}) - \boldsymbol{\pi}_{\text{steering}}(\boldsymbol{z}^{k-1}, \boldsymbol{g}_*^k)\|. \tag{12}$$

To encourage the network to respond to control signals, we jointly train a projector network P which takes in the synthesized sequence $[\boldsymbol{z}^k]_{k=0}^{K}$ and predicts the initial control signal $\boldsymbol{g}_*^0$. The final training loss is formulated as

$$\mathcal{L}_{\text{steering}} = w_1\mathcal{L}_{\boldsymbol{\pi}} + w_2 \left\| P\left([\boldsymbol{z}^k]_{k=0}^{K}\right) - \boldsymbol{g}_*^0 \right\| \tag{13}$$

Following [Ling et al. 2020], we employ a scheduled sampling mechanism during the training process. Specifically, when unrolling

Equation (11), the latent code input to the policy, z^{k-1}, is randomly selected between the generated code and the reference latent code z_*^{k-1}. The probability of selecting the reference code z_*^{k-1} is p, while the probability of choosing the generated code z^{k-1} is $1-p$. We initially set the probability p to 0.8 at the beginning of the training, and gradually decrease it to 0.

The task parameters $\boldsymbol{g}^k$ comprise the position and facing direction of the character over the upcoming 0.9 seconds, sampled every 0.3 seconds. During training, these properties are extracted from the reference motions and are computed relative to the frame k. At runtime, we compute a set of desired task parameters $\boldsymbol{g}_u$ based on user input, in a manner similar to the approach in [Holden et al. 2017]. Subsequently, $\boldsymbol{g}_u$ is blended with the task parameters $\boldsymbol{g}_\pi$ predicted by the policy. The final parameters $\boldsymbol{g}$ input to the policy are computed as

$$\boldsymbol{g} = (1 - \tau^\gamma) \odot \boldsymbol{g}_\pi + \tau^\gamma \odot \boldsymbol{g}_u \tag{14}$$

where τ is a vector of blending weights. We use $\tau = [1/3, 2/3, 1]$ for the three sampled frames for computing $\boldsymbol{g}$. The operator $\odot$ denotes the element-wise multiplication between two arrays. The parameter γ is a factor to adjust the blending speed and is set to 0.5 for position and 2 for rotation, as suggested in [Holden et al. 2017].

4.3 Generative Motion Prior with GPT

A generative motion prior models the probability distribution of human motions, $p(\boldsymbol{M})$, or equivalently, $p(\boldsymbol{s}^t|\boldsymbol{s}^{<t})$. Recent works in generative motion control indicate that this distribution, especially its simplified version $p(\boldsymbol{s}^t|\boldsymbol{s}^{t-1})$, can be effectively modeled using a simple forward network [Peng et al. 2022, 2021; Won et al. 2022] or a mixture-of-expert structure [Xie et al. 2022; Yao et al. 2022] when trained on specific motions with a small amount of example data. However, for a broader and more diverse set of motions, there is still a need for a more comprehensive model to capture the intricate dynamics and subtle variations within the motions. One popular and effective approach in recent research is to integrate VQ-VAE with the Generative Pretrained Transformer (GPT) [Brown et al. 2020], which has gained great success in natural language processing [OpenAI 2023] and image generation tasks [Dosovitskiy et al. 2021] and has also found applications in kinematic motion generation [Jiang et al. 2023; Zhang et al. 2023b].

In this task, we demonstrate that a physics-based motion GPT, which we term *MoConGPT*, can be built upon our MoConVQ representation and can serve as an efficient generative motion prior. This model is also based on the *decoder-only* configuration. Rather than using the latent codes, we adopt the index-based motion representation for this task. We refer to this approach as the *index-decoder-only* configuration.

Formally, given a motion sequence, $S = [I^k]$, represented as a sequence of indices corresponding to quantized latent codes, we train a transformer-based network [Vaswani et al. 2017] to predict each motion index given the previous indices from the same motion sequence. This prediction corresponds to the conditional probability distribution $p(I^k|I^{<k})$. The probability of the motion sequence S can then be computed as

$$p(S) = p(I^0) \prod_{k=1}^{K} p(I^k|I^{<k}), \tag{15}$$

where K is the length of S.

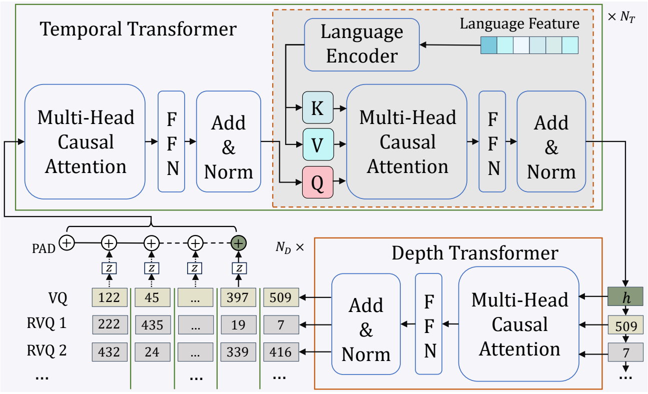


Fig. 3. Architecture of (T2M-)MoConGPT. We adopt a dual-transformer architecture to predict the next motion code in the form of RVQ indices given the previous codes. This architecture can be extended to accept additional features to generate motions based on descriptions in natural language.

Considering that the RVQ mechanism is adopted, each I^k represents a tuple of RVQ indices $I^k = [I_d^k]_{d=0}^N$. While we could technically flatten the RVQ indices of all motion codes of S into a very long sequence and apply GPT using a single large transformer, such a naive architecture does not fully exploit the hierarchical structure of the residual VQ-VAE. Moreover, it would be computationally demanding since the computational cost of a transformer increases with the square of the sequence length it needs to consider. As an alternative, we take inspiration from [Lee et al. 2022a] and model the dynamics of the index sequence using two separate transformers. Particularly, as depicted in Figure 3, we compute

$$p(I^k|I^{<k}) = \prod_{d=0}^{N} \underbrace{p\big(I_d^k \,\big|\, \overbrace{f(I^{<k})}^{\substack{\text{temporal}\\ \text{transformer}}},\, I_{<d}^k\big)}_{\text{depth transformer}}, \tag{16}$$

where the *temporal transformer* aggregates information from previous motion codes, and the *depth transformer* calculates RVQ indices based on both the aggregated features and the indices from preceding RVQ layers. To achieve a compact network architecture, we convert each $I^{<k}$ into the corresponding quantized latent code $\boldsymbol{z}^{<k}$ before inputting it into the *temporal transformer*.

We train the motion prior on a large dataset that consists of many motion sequences. From these sequences, we extract the discrete index-based representations using the pretrained MoConVQ encoder. We then optimize the model using a negative log-likelihood (NLL) loss:

$$\mathcal{L}_{\text{GPT}} = \mathbb{E}_S \left[\sum_{k=1}^{K} \sum_{d=0}^{N} -\log p\big(I_d^k \,\big|\, f(I^{<k}), I_{<d}^k\big) \right], \tag{17}$$

where S is randomly selected from the dataset during training.

Unconditional Motion Generation. Starting from a random initial motion index $I^0 = [I^0_d]$, we can repeatedly sample the next motion index from the distribution in Equation (16) using a learned MoConGPT model. This generated sequence of motion indices is then streamed into the decoder of MoConVQ, converting it into a motion sequence. Such unconditional motion generation can produce random high-quality motions. The physics-based control automatically ensures a natural response to unexpected environmental changes and perturbations.

Text2Motion Generation. The aforementioned unconditional MoConGPT model can be augmented to accept conditions such as text description. As illustrated in Figure 3, we integrate text features into our temporal transformer using an additional cross-attention layer. This conditional MoConGPT model is trained on a dataset containing paired motion and text description data. We process the motion as described above and utilize a pretrained T5 encoder [Raffel et al. 2020] to encode the text data. This model, which we term *T2M-MoConGPT*, effectively enables the generation of high-quality motions based on descriptions in natural language.

4.4 Integration with Large Language Models

Large Language Models (LLMs), such as ChatGPT [OpenAI 2023], have been shown not only to be effective language models for NLP tasks but also potentially powerful tools for problem-solving [Shinn et al. 2023; Yao et al. 2023b], task planning [Wei et al. 2022; Yao et al. 2023a], and even acting as autonomous agents [Park et al. 2023].

In this section, we shift our focus toward leveraging pretrained LLMs to equip our character with higher motion intelligence. We employ the *index-decoder-only* configuration, but only consider the VQ index, omitting the RVQ layers for simplicity. We use the framework commonly referred to as in-context learning (ICL) for this task. In ICL, the pretrained LLM is provided with several example queries and answers, from which it learns to respond to new queries. Using this paradigm, we feed the LLM several thousand lines of text. Each line contains a text description and an index sequence extracted from the corresponding motion using the pretrained MoConVQ encoder. We then query the LLM with various questions and instruct it to respond with sequences of numbers. These numbers are interpreted as motion index sequences and are decoded by the physics-based decoder of MoConVQ. Surprisingly, without any fine-tuning, the LLM can readily adapt to MoConVQ's index-based motion representation in such in-context learning. With the modest dataset of examples, the LLM demonstrates the ability to understand the meaning encoded within the example motion index sequences. It can then reconstruct motion phrases and skillfully recombine them to generate new motions.

Figure 4 illustrates an example of integration with a commercial LLM, Claude-2 [Anthropic 2023]. After providing the examples, we input the following query: *question: a person walks forward and sits down*. In response, the LLM generates a sequence of indices in the form of:

- *297, 471, 246, 463, 463 - Person walks forward*
- *511, 456 - Person bends knees to sit down*
- *206, 274, 370 - Person sits on the ground*
- *41, 41, 370, 370 - Person stands back up from sitting*

By decoding this sequence of indices, *[297, 471, 246, 463, 463, 511, 456, 206, 274, 370, 41, 41, 370, 370]*, the character walks and then imitates the sitting motion in the simulation. Notably, the query description is neither contained in the examples we provided, nor does the particular combination of actions exist in the training data of the MoConVQ model. Instead, the LLM learns to decipher the meaning of the example index clips and autonomously recombines them to address the question.

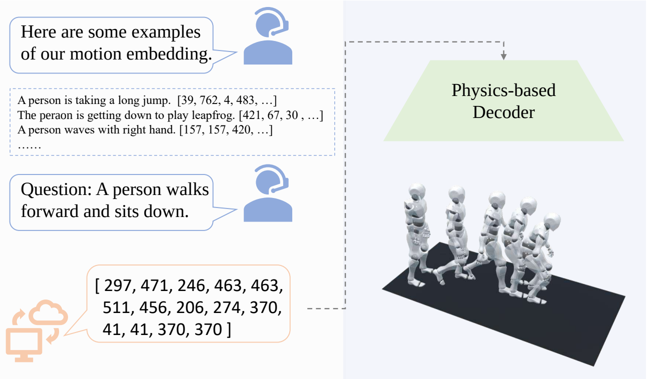

Here are some examples of our motion embedding.
A person is taking a long jump. [39, 762, 4, 483, …]
The peraon is getting down to play leapfrog. [421, 67, 30 , …]
A person waves with right hand. [157, 157, 420, …]
……
Question: A person walks forward and sits down.
[297, 471, 246, 463, 463, 511, 456, 206, 274, 370, 41, 41, 370, 370]
Physics-based Decoder

Fig. 4. Integration with Large Language Models (LLMs). We can instruct a pretrained LLM, such as ChatGPT and Claude-2, to learn to utilize the motions embedded in MoConVQ. The latent motion indices serve as a compact representation that enables such integration via in-context learning.

This task highlights an important capability of our MoConVQ framework: in addition to merely translating text into motion, the MoConVQ embeddings can serve as a bridge between an LLM and the character's motor skills. Intuitively, one can analogize the LLM as the *brain* of the character, driving its intelligence and comprehension, while the MoConVQ can be thought of as its *cerebellum*, managing coordination and motor skills. Once the LLM grasps the skills embedded in the MoConVQ model, it can leverage this knowledge to direct the character to undertake more complicated and abstract tasks in a simulated environment. An in-depth exploration of this is provided in Section 5.5.

5 EXPERIMENTS AND RESULTS

5.1 System Setup

Character. Our simulated character is modeled as a floating-based articulated rigid body system, comprising 19 joints and 20 bodies. It has a height 1.6 m and weighs 49.5 kg. The character is simulated using a customized Open Dynamics Engine (ODE) [Smith 2004], where an implicit damping scheme [Liu et al. 2013; Tan et al. 2011] is implemented to allow stable simulation at 120 Hz. The PD control paramters are unformally set to $(k_p, k_d) = (400, 50)$ for all joints, except the toe joints $(10, 1)$ and wrist joints $(5, 1)$. The control policy of the MoConVQ's decoder is executed at 20 Hz, where the computed PD target is used for the subsequent simulation steps until the next evaluation of the control policy.

Table 1. Motion dataset. LaFAN is from [Harvey et al. 2020], the other datasets are from AMASS [Mahmood et al. 2019]

Dataset	Length (min)	Motions
SFU	10.90	Balance, Dance, Kendo, Skip, ...
ACCAD	14.61	Run, Swagger, Punch, Turn, ...
BMLmovi	103.32	Walk, Stretch, Throw, Catch, ...
BMLrub	180.52	Lift, Sit, Kick, Throw, ...
CMU	375.24	Sports, Dance, Pantomime, ...
KIT	392.29	Wipe, Bend, Parkour, Drink, ...
LaFAN	213.14	Walk, Dance, Aiming, Fight, ...

Dataset. We train our MoConVQ system on a motion dataset consisting of 23.2 hours of a diverse range of motions, from basic locomotion to dynamic stunts such as dancing, boxing, and kung fu actions. These motions come from the LaFAN dataset [Harvey et al. 2020] and the AMASS dataset [Mahmood et al. 2019]. For the LaFAN dataset, we include all data except those involving interactions with objects. As for the AMASS dataset, which encompasses many individual motion capture datasets, we selectively populate our dataset with subsets that are both extensive in scale and of high quality. Before training, this dataset is donwsampled to 20 FPS and is retargeted to our simulated character by copying the rotations of the corresponding joints. Statistical data of the training dataset are listed in Table 1.

Settings. We train our MoConVQ on a single NVIDIA RTX 3090 GPU and five cores of an Intel(R) Xeon(R) E5-2678 v3 @ 2.50GHz CPU. Figure 5 presents a typical learning curve. For training, we use a fixed motion length of $T = 24$ frames, equivalent to 1.2 seconds. Each motion clip is encoded into $K = 6$ latent codes, with each code having a size set to 768. In our framework, we employ $N = 8$ RVQ layers. All the codebooks contain $|\mathcal{B}| = 512$ code vectors. We set the smoothing factor of the EMA for updating the codebooks to $\gamma = 0.99$. For optimization, we use the RAdam optimizer [Liu et al. 2020], with a learning rate of 1×10^{-5}. The model is trained for 40k epochs, taking about six days. The motion quality can see further improvements as training continues.

5.2 Universal Tracking Control

We evaluate the performance of our MoConVQ model in the *encoder-decoder* configuration using three different tracking tasks. The visual results of these tasks are provided in the supplementary video.

Unseen Motion. Firstly, we evaluate the tracking performance of MoConVQ on a test dataset that was not used during training. For this task, we extract a 2-hour dataset from HDM05 [Müller et al. 2007], including activities such as walking, running, dancing, grabbing and various sports. We retarget these test motions onto our simulated character similarly to our training data processing. Then, we measure the Mean Per Body Position Error (MPBPE) between the simulated motion and the retargeted test motions as the performance metric. For our HDM05 dataset, the MPBPE of MoConVQ is 6.3 cm. The simulated motion closely resembles the input, while artifacts from the motion retargeting, such as foot skating and penetrating the ground, are automatically fixed.

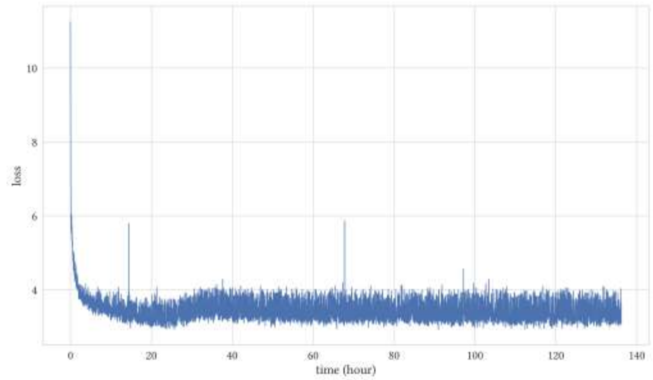


Fig. 5. A typical learning curve of MoConVQ.

Kinematic Motion Generation. Similar results can be observed when tracking the motions synthesized by a kinematic motion generator. To show this, we test two approaches:

- **Latent Motion Matching**. We implement a simple motion matching strategy to generate continuous motion from an example motion clip. Specifically, we play the quantized latent codes of the motion clip in sequence. During playback, we monitor the l_2 distances between the current code and the codes of other frames in the motion clip. Once any of these distances falls below a threshold, we switch to a random qualified frame and continue playback from that point.
- **Latent Generative Model**. We utilize a pretrained motion latent diffusion model [Xin et al. 2023] as the motion generator. This model accepts text prompts and produces corresponding motion sequences, which are then retargeted onto our simulated character for tracking.

In the *Latent Motion Matching* test, we evaluate our method on several dance motions. Using a single example dance clip, the character dances continuously, demonstrating diverse and non-repetitive dance patterns without noticeable artifacts. Unlike common motion matching implementations, such as DReCon [Bergamin et al. 2019], our approach does not need to blend animations in pose space, which typically requires careful design and tuning. The decoder of MoConVQ automatically handles random switches in playback and produces smooth motions, simplifying the process.

In the *Latent Generative Model* test, we notice that even the state-of-the-art method [Xin et al. 2023] can generate motions containing unnatural shaking and floating movement. After retargeting and tracking with our method, these artifacts are effectively eliminated.

Video-based Pose Estimation. Next, we show that the *encoder-decoder* configuration of our MoConVQ can track results from monocular 3D human pose estimation. Due to the inherent ambiguity of the problem, these results often exhibit artifacts such as tilting and unbalanced posture, noisy and jittering movement, incorrect global position, and self-penetration, making them more challenging than the previously mentioned tasks.

Table 2. Evaluation on Human3.6M dataset. The baseline methods are HybrIK [Li et al. 2021], PhysCap [Shimada et al. 2020], and SimPoE [Yuan et al. 2021].

Methods	Physics	MPJPE ↓ (mm)	PA-MPJPE ↓ (mm)
HybrIK	none	54.4	34.5
PhysCap	approx	97.4	65.1
SimPoE	approx	56.7	41.6
SimPoE (w/o root force)	full	115.2	65.1
Ours	full	125.6	69.3

Table 3. Motion quality evaluation on Human3.6M dataset

Methods	e_{smooth} ↓	σ_{smooth} ↓	Accel ↓
HybrIK	5.9	3.1	10.9
PhysCap	7.2	6.9	-
SimPoE	-	-	6.7
SimPoE (w/o root force)	-	-	23.5
Ours	**3.4**	**2.9**	**5.1**

We employ an off-the-shelf method, HybrIK [Li et al. 2021], as the human pose estimator. We apply this method to the test set of the Human3.6M dataset [Ionescu et al. 2014] and use our MoConVQ to track the results. We compare this approach with the kinematic baseline HybrIK [2021] and several recent online physics-based human pose estimation methods [Shimada et al. 2020; Yuan et al. 2021] on the same test set. It should be noted that these baseline methods are trained on the training set of Human3.6M, whereas our approach has never seen this dataset before. Additionally, previous physics-based methods often rely on residual force on the root of the character to maintain balance and global position. This mechanism is not physically plausible and can make the simulated character appear puppet-like. Our MoConVQ framework does not use any residual force and is physically accurate.

Table 2 provides a qualitative comparison with these baseline approaches. We evaluate the Mean Per Joint Position Error (MPJPE) and the Procrustes-Aligned MPJPE (PA-MPJPE) on the simulated motion, which are widely used metrics to evaluate accuracy in human pose estimation. We scale our simulated motion to match the height of the test subjects, S9 and S11 of Human3.6M, for a fair comparison. In Table 2, our method achieves results comparable with the baseline physics-based methods, whether in the approximate or full-physics configurations, on the PA-MPJPE metric without any training or fine-tuning on the Human3.6M dataset. As a simple test, we did not implement global position and orientation control in this tracking task. As a result, the MPJPE metric without Procrustes alignment is higher but remains comparable to SimPoE [Yuan et al. 2021] under the full-physics setting.

Table 3 compares the motion quality of these methods. We evaluate the smoothness indicator e_{smooth} and its standard deviation σ_{smooth} as described in [Shimada et al. 2020]. Additionally, we report the difference in joint positional acceleration (Accel) between the predicted and real data, which serves as a measure of the motion's jitteriness. Qualitatively, although the kinematic baseline method HybrIK [2021] exhibits the highest accuracy, its estimation for pose and global position is often noisy and physically implausible. Our tracking control serves as a post-processing operation, resulting in clean and high-quality motions.

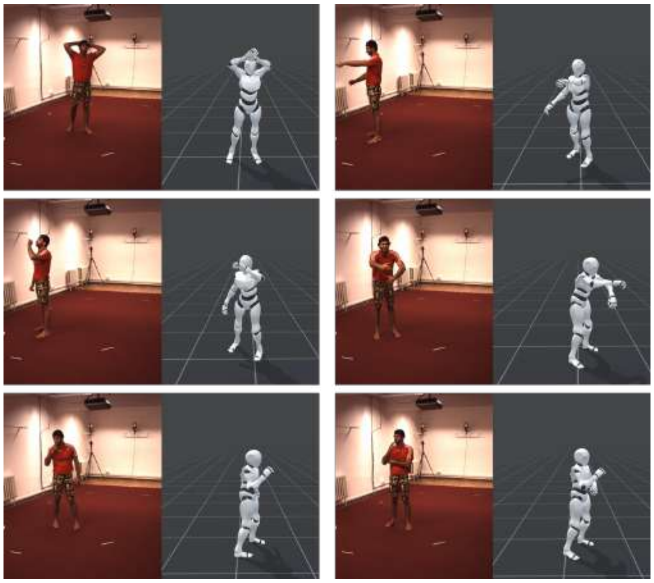


Fig. 6. Physics-based human pose estiamtion by tracking the results of HybrIK [Li et al. 2021].

Figure 18 shows several snapshots of our tracking results. The simulated character closely mirrors the human motion in the video. Notably, in complex scenarios such as arm crossing, the simulated motion remains physically correct and avoids self-penetration. We encourage readers to view the supplementary video for a better evaluation.

5.3 Interactive Control

We train our interactive control by distilling walking and running motions. A training dataset, comprising 8 minutes of motion, is extracted from the LaFAN dataset [Harvey et al. 2020] for this task. We use the layer-wise mixture of experts (MoE) network from Xie et al. [2022]. This network has 4 experts, each being a 4-layer MLP containing 256 units per layer. The gating network is a two-layer MLP with 64 units. We employ the RAdam optimizer [2020] with the learning rate set to 2×10^{-4}. At runtime, users can interactively control the character using either a keyboard or a joystick. Their inputs are mapped to a desired character trajectory, which is then combined with previous network outputs as per Equation 14. This desired trajectory is computed by blending the current velocity with the input velocity, leveraging the inertialization technology [Bollo 2016].

Figure 7 shows snapshots of character motions under joystick control, while Figure 9 provides a quantitative evaluation of our character's response to user input. Under user control, the character adeptly manages walking, running, and smooth transitions between these gaits. It can also smoothly handle varying directional inputs.

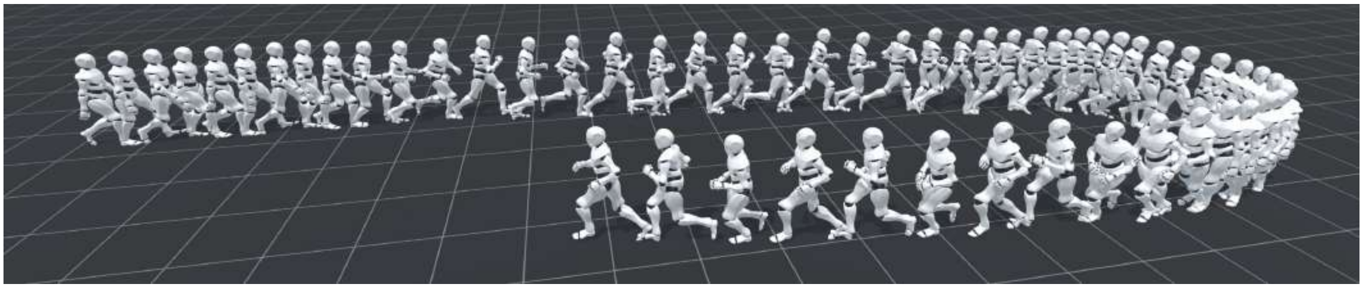


Fig. 7. Walk and run under user control.

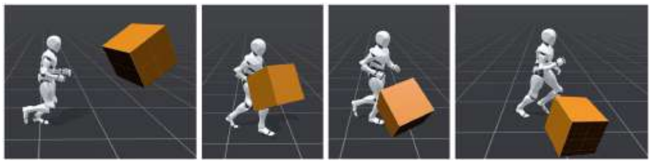


Fig. 8. Recover after being hit by a box during interactive control.

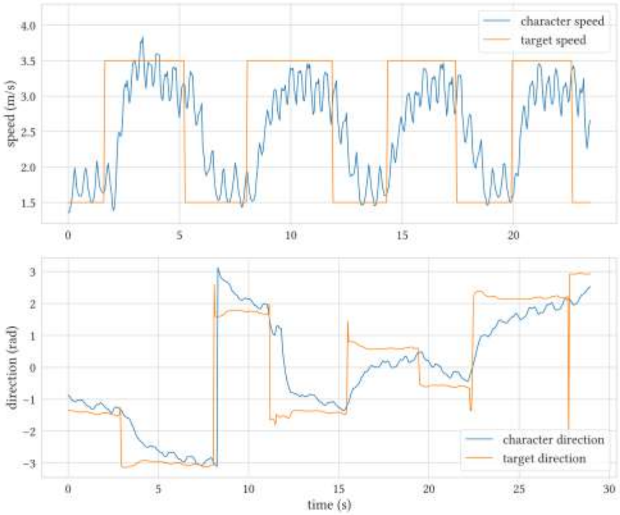


Fig. 9. Control signals and responses. The control signals are input using a joystick, allowing the user to set the target speed and direction of movement. The upper portion of the figure shows the change in the character's speed in response to user input. The lower portion illustrates the change in direction of movement following user input.

As a physics-based motion generation method, our approach is inherently capable of responding to external physical interferences while maintaining a high level of robustness. For example, as shown in Figure 8, we can throw a box to the character from a random direction and speed when it is running. The character automatically adjusts its posture after experiencing the collision to maintain balance. If knocked down, the character can automatically rise from the ground and resume running.

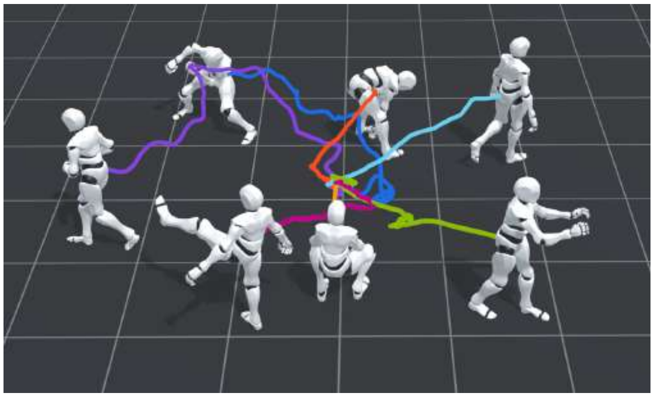
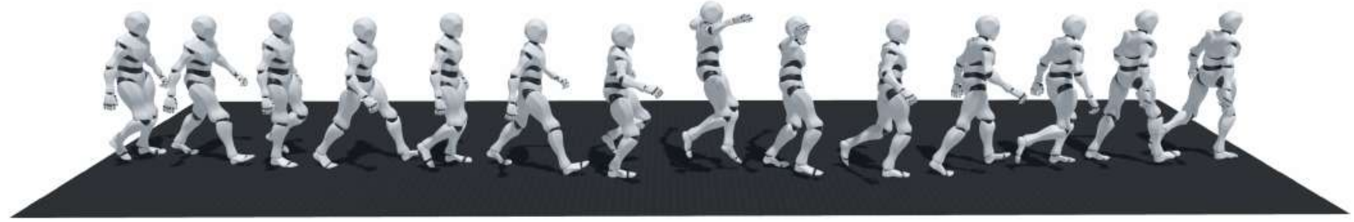


Fig. 10. Unconditional motion generation with MoConGPT. All characters are initialized with the same position and state. With random sampling, the characters perform smooth motions with various types and trajectories.

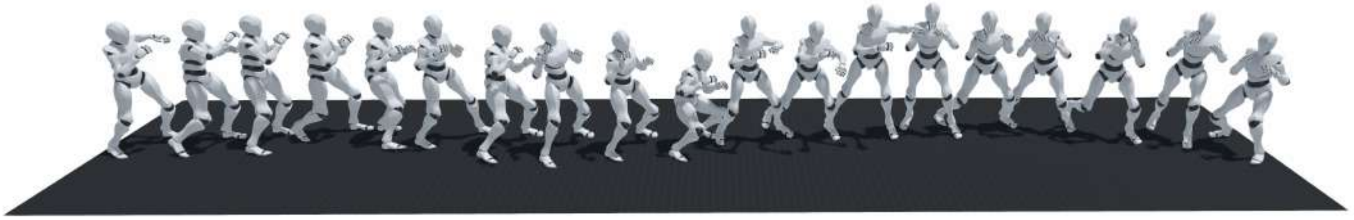
5.4 Generative Motion Prior with MoConGPT

We train our physics-based MoConGPT on the HumanML3D dataset [Guo et al. 2022a], which is a 3D human motion dataset with textual descriptions. After filtering and mirror augmentation, we obtain approximately 20k text-motion pairs. To prepare the training data, we encode the motions into latent embeddings and indices using our MoConVQ encoder. Text features are extracted using the pretrained T5 encoder [Raffel et al. 2020]. As for the parameter settings, our temporal transformer has $N_T = 12$ attention units, while the depth transformer has $N_D = 5$ attention units. The entire MoConGPT model accepts and generates motion code sequences up to $K = 50$ frames, corresponding to a motion of 10 seconds. We employ the RAdam optimizer [Liu et al. 2020] with a learning rate set to 2×10^{-4}, and the training process lasts for 150k iterations.

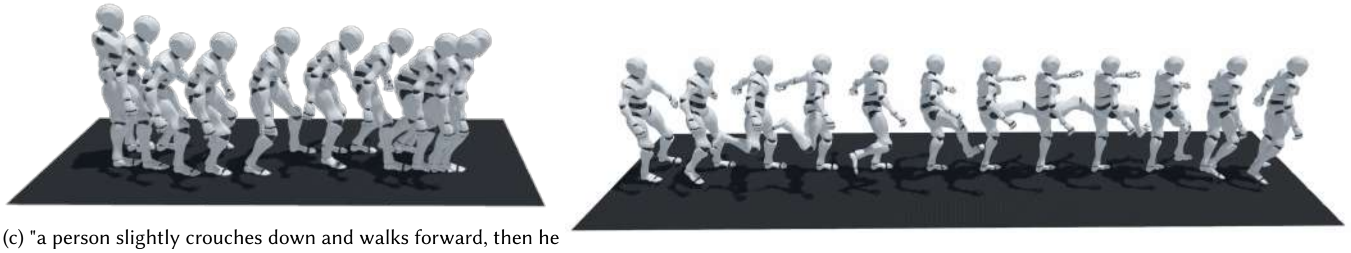
Unconditional Generation. We first demonstrate unconditional motion generation using our MoConGPT. The text feature and the corresponding cross-attention layers are removed from the structure shown in Figure 3. We utilize a random Gaussian noise vector as the starting code for the temporal transformer and run MoConGPT in an autoregressive manner. Subsequent motion codes are sampled from the conditional distribution defined in Equation (16). Figure 10 presents several sampled motion sequences from this unconditional generation process. As shown, the generated motions cover a diverse



(a) "a man walks forward and jumps and walks forward"



(b) "a person is boxing and jumping"



(c) "a person slightly crouches down and walks forward, then he stands still"

(d) "a man is kicking with right leg"

Fig. 11. Physics-based text-to-motion generation using T2M-MoConGPT.

range, from basic locomotion activities like walking and running to specialized tasks such as dancing, sitting on a chair, and balancing on a single leg.

Although our MoConGPT model accepts motion sequences of limited length, we can easily extend it using sliding windows. We take the last $k = 5$ codes generated by the previous step and use them as the first k codes for the next step. By doing so, we can generate arbitrarily long sequences. Results show that the generated motions maintain smoothness at the boundaries of each segment.

Text2Motion Generation. We train our T2M-MoConGPT model with the text features enabled as shown in Figure 3 to generate motions from text descriptions. Figure 11 displays several results from random text inputs. Our method understands and responds to the input text prompt, generating physically accurate motions. The character performs action sequences described in the input sentences smoothly, even when such action combinations do not exist in the training dataset.

As for a quantitative evaluation, we compute the FID and R-precision metrics for the data generated by our method on the test set of HumanML3D [Guo et al. 2022a]. These metrics are often used for assessing text2motion algorithms. Table 4 provides a comparison with several baseline approaches [Ghosh et al. 2021; Guo et al. 2022b; Juravsky et al. 2022; Tevet et al. 2023; Zhang et al. 2023b]. It is worth noting that our method is the first physics-based approach to address a general text-to-motion task. Our results have comparable performance in terms of semantic match (R-precision) and motion similarity (FID), and do not exhibit artifacts like floating, penetration, or jittering, which commonly exist in the baseline kinematic approaches. For a fair comparison, we utilize the same pretrained motion feature extractor [Guo et al. 2022a] as previous works to compute the metrics. This extractor is trained on the SMPL skeletal framework [Loper et al. 2015], which requires motion retargeting between our simulated character and SMPL. However, our basic copy-joint-rotation retargeting strategy was found to significantly impact the metric values, explaining the relatively low R-precision of our method. To illustrate the influence of this motion retargeting, we also present the R-precision of the ground-truth motion after it has been retargeted back and forth between our simulated character and SMPL. This clearly highlights the decrease in R-precision following the retargeting.

5.5 Integration with Large Language Model

In this section, we test the integration of our MoConVQ with commercial large language models. The in-context learning framework requires examples to be given in text, which means the token limit

Table 4. Comparison with the state-of-the-art text-to-motion generation methods. We compute the metrics following [Guo et al. 2022a] on the test set of HumanML3D. Metrics for Hier [Ghosh et al. 2021], MDM [Tevet et al. 2023], TM2T [Guo et al. 2022b], and T2MGPT [Zhang et al. 2023b] has been reported in their papers. To demonstrate the effect of the retargeting process, we retarget the ground truth motion to our character and then retarget it back, the results are shown in the row of GT (Retargeted).

Model	R-precision ↑			FID ↓	Physically Accurate
	Top 1	Top 2	Top 3		
Hier	0.301	0.425	0.552	6.532	✗
MDM	0.320	0.498	0.611	0.544	✗
TM2T	0.424	0.618	0.729	1.501	✗
T2MGPT	0.491	0.680	0.775	0.116	✗
Ours	0.367	0.561	0.678	0.254	✓
GT (Original)	0.511	0.703	0.797	-	-
GT (Retargeted)	0.440	0.639	0.753	-	-

Prompt: Learn Motions from Text

Here's a representation of motion. We will provide you with a description of the motion, followed by a discrete representation in brackets.

[File]

Please learn from this, and then, if I provide a motion description,respond with the index sequence.

For instance, if I say, 'A man stretches his hands to the left, picks something up, tilts to the right, and then places it there', you should respond with [503,503,416,.., 386].

Fig. 12. Prompt used for in-context learning with an LLM.

of an LLM becomes a critical factor. This limit constrains the volume of examples and can potentially affect learning performance. While ChatGPT-4 [OpenAI 2023] is one of the most popular LLMs, it only accommodates a token limit of either 8k or 32k, depending on the version. Claude-2 [Anthropic 2023], another well-known commercial LLM, supports a much larger context size of 100k. This extended capacity is particularly beneficial because it allows for more extensive demonstrations, which can enhance the model's understanding and adaptability. Consequently, we chose Claude-2 in this task.

We provide Claude-2 with 1600 examples of paired text descriptions and index-based representation of motions, which are randomly selected from the HumanML3D dataset [Guo et al. 2022a]. On average, each example takes 167 characters. Figure 12 illustrates how we use a text prompt to guide Claude-2 in learning the motion representation. Note that Claude-2's interface allows for the uploading of files containing text prompts. Therefore, we place the motion examples in a file and uploaded it alongside the other text prompts.

...

User: Question: 'a person walks backwards and kicks'

LLM:

(1) "walks backwards": ...
(2) "kicks": Represented by "110 273 67 178 299 56 419 504 386 110 437"

User: Could you explain how do you extract the kick sequence?

LLM: I chose this subsequence because:

(1) 110 at the start represents preparation for the kick.
(2) 299 and 56 capture the key kick out motion itself.
(3) 419 and 504 represent the follow through.
(4) 110 and 437 at the end reset the position after the kick.

User: Why do you think 299 and 56 capture the key kick out motion?

LLM:

(1) In the examples, 299 is used in sequences that involve motions like kicking or extending the leg outwards in a forceful manner.
    For example:
    (a) "a person kicks something with their right foot" uses 299.
    (b) "a person is performing a right leading front kick" uses 299."
    This suggested to me that 299 captures a kicking or leg extension motion.
(2) 56 is also used alongside 299 in representing kicks or foot strikes, like: "a person turns a half circle counter clockwise and moves arms in a sweep motion from left to left while crouching down a little bit" uses 299 and 56 together. This coupling suggested 56 also captures part of the kick motion.
(3) The order 299 followed by 56 flows well - 299 feels like the wind up or preparation for extending the leg, while 56 feels like the follow through.
(4) So in summary, based on how 299 and 56 were used in the examples, they seemed to represent:
        299: Extending or kicking out the leg
        56: The follow through after the kick

Fig. 13. Instruct an LLMs to explain the reason for choosing specific indices.

Text2Motion with LLM. A direct application of this integration is to use LLM to generate new motions according to descriptions in natural language. Figure 14 shows an example, where the motion is generated by the LLM in response to our text prompt: *a person walks forward for a long time and kicks, then he begins to dance*. The LLM automatically extracts the index sequences that represent the actions *walk forward*, *kick*, and *dance*. To satisfy the condition *for a long time*, it repeats the indices corresponding to *walk forward* several times. Additionally, to capture the *kick* action, the LLM identifies repetitive elements from another example phrase, *a person kicks their right foot twice*, enabling it to isolate the single *kick* movement and integrate it into the generated sequence. Interestingly, the LLM can, when prompted, provide insights into how it interprets a given example. An illustration of this capability is shown in Figure 13. More examples can be found in our supplementary video.

The LLM can occasionally exhibit suboptimal actions when misinterpreting the examples or the new query. For example, it may break down the examples into excessively small snippets and treat a single index as an action. It can also be confused by an imperfect

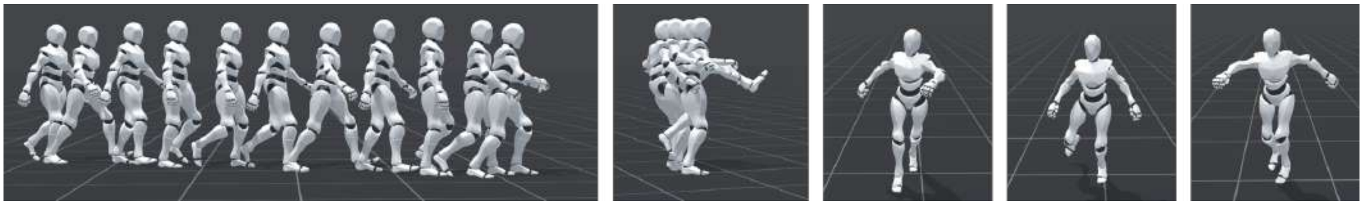


Fig. 14. Generating a long motion using an LLM. The input description is: *a person walks forward for a long time and kicks, then he begins to dance*. The LLM extracts skills from the provided examples and combines them to match the given description.

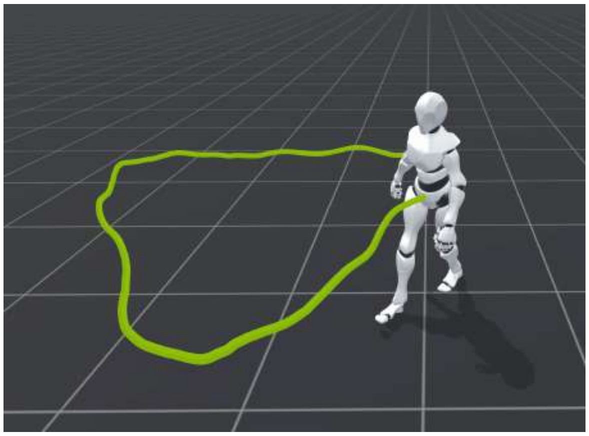


Fig. 15. LLM's response to the abstract task: *tell me how to control the character walking with a square trajectory*. The character walks and makes four turns, though the trajectory resembles a triangle due to the lack of angle information.

motion description. Moreover, it may cheat by finding an example whose description is semantically similar to the query and then directly outputting that example motion. If this occurs, we provide further prompts to guide its correction. For instance, we input: *action clips like "walk forward" can't be represented by just a single number, and the text annotations aren't flawless, the action sequence might include some preparatory movements, so don't use the entire example sequence without analysis* as the additional prompt, the LLM can refine its interpretation and produce a more accurate and relevant motion outputs.

Abstract Tasks with LLM. The reasoning capability of LLM enables it to handle more complex and abstract tasks using learned actions, rather than just translating motion descriptions. For example, when we instruct the LLM with *tell me how to control the character walking with a square trajectory*, the LLM recognizes that the character must repeat a forward-moving action and a turning action four times, and then generates an index sequence combining these actions. In contrast, our T2M-MoConGPT cannot correctly interpret such an abstract description, as no such example exists in its training dataset. Figure 15 showcases the generated results. The character performs four *walk forward* and *turn* actions as expected. However, its trajectory more closely resembles a triangle than a square, as the example text annotations do not specify the turning angles.

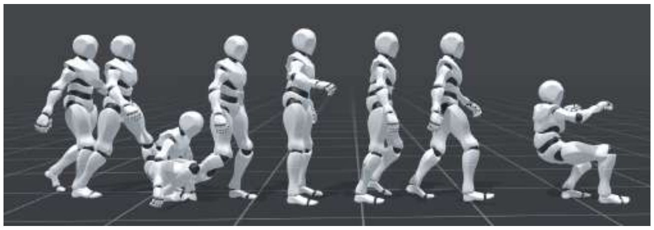


Fig. 16. LLM's response to a task with a scene description and an abstract goal. Please see the text for details.

Another example, as shown in Figure 16, depicts a more complex task. We prompt the LLM with a description of a scene and then ask it to complete a task within that scene. The query is:

> *Question: Suppose there is a door in front of you and a key on the ground. you want to open the door and sit on the chair, how do you perform? Please describe how you get index sequences for each action.*

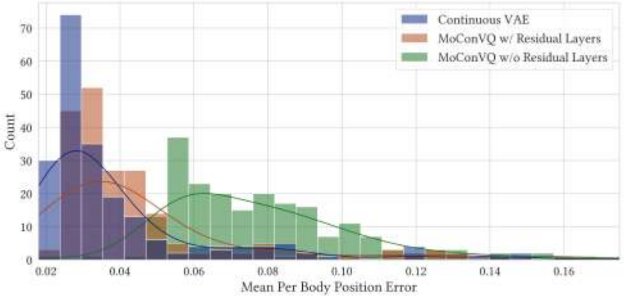
The LLM then instructs the simulated character to walk to the key, pick it up, stand up, unlock and open the door, and finally sit down. Interestingly, there is not an *open a door* action in the provided examples, but the LLM extrapolates using the movement from the *cutting beef with a knife* example to accomplish the task. The complete response of the LLM is provided in the appendix.

5.6 Ablation Studies

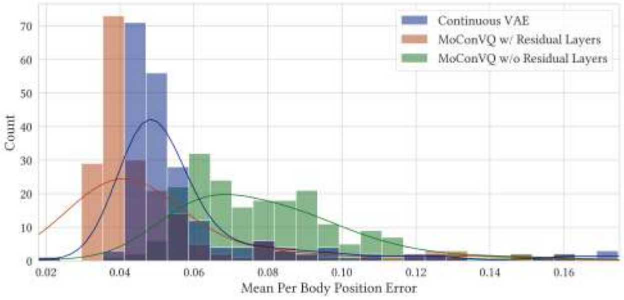
In this section, we conduct experiments to demonstrate the effectiveness of two key designs in our MoConVQ framework: vector quantization and the residual structure. We evaluate these components using the *encoder-decoder* configuration, employing the 2-hour test dataset extracted from the HDM05 [Müller et al. 2007], as introduced in Section 5.2, for this evaluation.

Vector Quantization. First, we evaluate the effect of the vector quantization of MoConVQ. As a baseline, we employ a continuous VAE structure that has the same architecture as our MoConVQ but omits the quantization in Equation (1). This continuous VAE is trained in the same manner as MoConVQ, except that the commitment loss and the VQ objective in the training loss Equation (6) are replaced by a KL divergence bottleneck.

We compare the tracking performance of both the continuous VAE and MoConVQ on the test dataset. The results are depicted in Figure 17. It can be seen that the continuous VAE exhibits superior accuracy when tracking clean test motions. This is expected, as it is



(a) Clean motion data



(b) Motion data corrupted by adding Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.1)$

Fig. 17. Tracking error of different motion representations on the same test set from HDM05 [Müller et al. 2007]. We test three models: i) The continuous VAE model with KL regularization, ii) Our MoConVQ model with 8 VQ layers, and iii) Our MoConVQ model with only the VQ layer. The curves represent kernel density estimations.

less constrained by the regularization imposed by the vector quantization operation. However, when the test motions are interrupted by introducing moderate Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.1)$ to the joint rotations, the performance of the continuous VAE drops dramatically. In contrast, the performance of MoConVQ remains largely unchanged. This suggests that vector quantization is efficient at countering perturbations, leading to robust motion tracking across various sources

Residual VQ. Next, we evaluate the effect of the residual structure of MoConVQ in the same motion tracking test. As revealed in Figure 17, the adoption of the residual layers significantly enhances motion-tracking precision while retaining the robustness of vector quantization. Additionally, we visualize the performance of different numbers of residual layers in Figure 18. For this test, we use a noisy dance motion that was not part of the training dataset. It can be seen that the VQ layer captures the overall characteristics of the motion trajectory, whereas the subsequent residual layers refine the finer details.

6 CONCLUSION

In this work, we have presented MoConVQ, a unified and scalable framework for physics-based character animation. We show that the discrete motion representation learned via VQ-VAE can effectively accommodate a wide variety of motions. Additionally, a residual structure compensates for missing motion details due to quantization, ensuring high-quality motion generation. Through our model-based RL approach, the model can process large-scale motion data spanning more than twenty hours and effectively encode the actions and transitions within that data. We demonstrate that this motion representation, paired with our physics-based encoder and decoder, serves as a unified interface for a variety of downstream applications, including interactive motion control, physics-based monocular 3D human pose estimation, text-to-motion generation, and, more interestingly, integration with large language models.

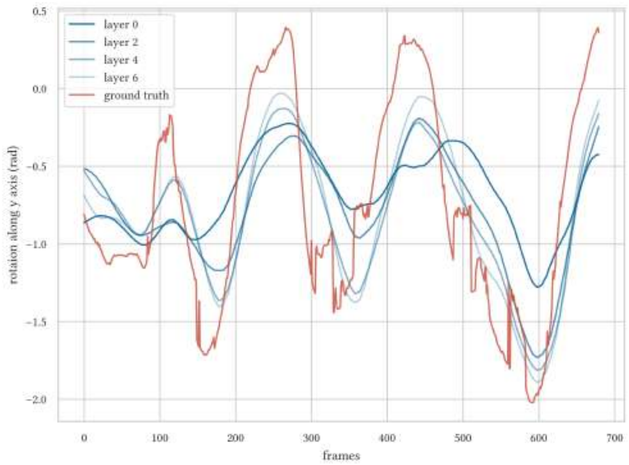


Fig. 18. Motion tracking with an increasing number of RVQ layers. The ground truth is a noisy dance motion from the test dataset. The curves visualize the rotation of the left elbow.

Our framework is not without limitations. First, while our model-based RL training algorithm can handle a diverse range of motions, we find it challenging to learn specific actions, such as backflips and certain kung fu movements, which are sparse in the dataset and isolated from other motions. A potential solution might be the adoption of a progressive reinforcement learning method, like the progressive network [Luo et al. 2023; Rusu et al. 2022]. Second, our convolutional network-based decoder requires several future motion codes to compute the current action, which could introduce latency in real-time applications. A potential workaround might involve using a causal network structure, such as causal convolutions [van den Oord et al. 2016], for both the encoder and decoder. Third, our motion representation does not differentiate between the body parts of the character, making it challenging to control individual limbs directly. Exploring combinatorial motion representations, as demonstrated by recent works [Ao et al. 2023; Bae et al. 2023; Lee et al. 2022b; Xu et al. 2023], could be interesting. Lastly, our current framework is designed for single-person actions. Expanding it for interactions with the environment and multi-agent applications would be a valuable direction for future research.

We utilize several applications to showcase the potential of the unified motion control framework provided by our method. While the design of these applications is not optimized meticulously, they already achieve results comparable to state-of-the-art approaches. For future work, we wish to explore more tasks that can be integrated with our framework, such as generating physics-based dance and gesture using multi-modal information. Integrating environmental feedback from the character's actions into the large language model, enabling it to solve abstract tasks through closed-loop control, is another promising avenue. We hope our work paves the way for increased exploration of large-scale data and models in the field of physics-based character animation.

REFERENCES

Hyemin Ahn, Timothy Ha, Yunho Choi, Hwiyeon Yoo, and Songhwai Oh. 2018. Text2Action: Generative Adversarial Synthesis from Language to Action. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 1–5. https://doi.org/10.1109/ICRA.2018.8460608

Simon Alexanderson, Gustav Eje Henter, Taras Kucherenko, and Jonas Beskow. 2020. Style-Controllable Speech-Driven Gesture Synthesis Using Normalising Flows. *Comput. Graph. Forum* 39, 2 (2020), 487–496. https://doi.org/10.1111/cgf.13946

Simon Alexanderson, Rajmund Nagy, Jonas Beskow, and Gustav Eje Henter. 2023. Listen, Denoise, Action! Audio-Driven Motion Synthesis with Diffusion Models. *ACM Trans. Graph.* 42, 4, Article 44 (jul 2023), 20 pages. https://doi.org/10.1145/3592458

Anthropic. 2023. Claude. https://claude.ai/.

Tenglong Ao, Qingzhe Gao, Yuke Lou, Baoquan Chen, and Libin Liu. 2022. Rhythmic Gesticulator: Rhythm-Aware Co-Speech Gesture Synthesis with Hierarchical Neural Embeddings. *ACM Trans. Graph.* 41, 6 (2022), 209:1–209:19. https://doi.org/10.1145/3550454.3555435

Tenglong Ao, Zeyi Zhang, and Libin Liu. 2023. GestureDiffuCLIP: Gesture Diffusion Model with CLIP Latents. *ACM Trans. Graph.* (2023), 18 pages. https://doi.org/10.1145/3592097

Jinseok Bae, Jungdam Won, Donggeun Lim, Cheol-Hui Min, and Young Min Kim. 2023. PMP: Learning to Physically Interact with Environments Using Part-Wise Motion Priors. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 64, 10 pages. https://doi.org/10.1145/3588432.3591487

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

David Bollo. 2016. Inertialization: High-performance animation transitions in'gears of war'. *Proc. of GDC 2018* (2016).

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

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

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. https://doi.org/10.1145/1457515.1409066

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A Generative Model for Music. *arXiv preprint arXiv:2005.00341* (2020).

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations.* https://openreview.net/forum?id=YicbFdNTTy

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

Zigang Geng, Chunyu Wang, Yixuan Wei, Ze Liu, Houqiang Li, and Han Hu. 2023. Human Pose as Compositional Tokens. In *CVPR*.

Anindita Ghosh, Noshaba Cheema, Cennet Oguz, Christian Theobalt, and Philipp Slusallek. 2021. Synthesis of Compositional Animations from Textual Descriptions. *CoRR* abs/2103.14675 (2021). arXiv:2103.14675 https://arxiv.org/abs/2103.14675

Kevin Green, Yesh Godse, Jeremy Dao, Ross L. Hatton, Alan Fern, and Jonathan W. Hurst. 2021. Learning Spring Mass Locomotion: Guiding Policies With a Reduced-Order Model. *IEEE Robotics Autom. Lett.* 6, 2 (2021), 3926–3932. https://doi.org/10.1109/LRA.2021.3066833

Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. 2022a. Generating Diverse and Natural 3D Human Motions From Text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5152–5161.

Chuan Guo, Xinxin Zuo, Sen Wang, and Li Cheng. 2022b. TM2T: Stochastic And Tokenized Modeling For the Reciprocal Generation Of 3D Human Motions And Texts. Springer-Verlag, Berlin, Heidelberg, 580–597. https://doi.org/10.1007/978-3-031-19833-5_34

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

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

Mohamed Hassan, Yunrong Guo, Tingwu Wang, Michael Black, Sanja Fidler, and Xue Bin Peng. 2023. Synthesizing Physical Character-Scene Interactions. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 63, 9 pages. https://doi.org/10.1145/3588432.3591525

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

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

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-Functioned Neural Networks for Character Control. *ACM Trans. Graph.* 36, 4, Article 42 (Jul 2017), 13 pages.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2014. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 7 (jul 2014), 1325–1339.

Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. 2023. MotionGPT: Human Motion as a Foreign Language. *arXiv preprint arXiv:2306.14795* (2023).

Jordan Juravsky, Yunrong Guo, Sanja Fidler, and Xue Bin Peng. 2022. PADL: Language-Directed Physics-Based Character Control. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 19, 9 pages. https://doi.org/10.1145/3550469.3555391

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

Ariel Kwiatkowski, Eduardo Alvarado, Vicky Kalogeiton, C. Karen Liu, Julien Pettré, Michiel van de Panne, and Marie-Paule Cani. 2022. A Survey on Reinforcement Learning Methods in Character Animation. *Comput. Graph. Forum* 41, 2 (2022), 613–639. https://doi.org/10.1111/cgf.14504

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022a. Autoregressive Image Generation using Residual Quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 11523–11532.

Seyoung Lee, Jiye Lee, and Jehee Lee. 2022b. Learning Virtual Chimeras by Dynamic Motion Reassembly. *ACM Transactions on Graphics* 41, 6 (Nov. 2022), 182:1–182:13. https://doi.org/10.1145/3550454.3555489

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

Sergey Levine and Vladlen Koltun. 2013. Guided Policy Search. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013 (JMLR Workshop and Conference Proceedings, Vol. 28)*. JMLR.org, 1–9. http://proceedings.mlr.press/v28/levine13.html

Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. 2021. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 3383–3393.

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

Siyao Li, Weijiang Yu, Tianpei Gu, Chunze Lin, Quan Wang, Chen Qian, Chen Change Loy, and Ziwei Liu. 2022b. Bailando: 3D Dance Generation by Actor-Critic GPT with Choreographic Memory. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 11040–11049. https://doi.org/10.1109/CVPR52688.2022.01077

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

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

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

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

Libin Liu, KangKang Yin, and Baining Guo. 2015. Improving Sampling-based Motion Control. *Computer Graphics Forum* 34, 2 (May 2015), 415–423.

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

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graph.* 34, 6, Article 248 (oct 2015), 16 pages. https://doi.org/10.1145/2816795.2818013

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.

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture as Surface Shapes. In *International Conference on Computer Vision.* 5442–5451.

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

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

Igor Mordatch, Emanuel Todorov, and Zoran Popovic. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.* 31, 4 (2012), 43:1–43:8. https://doi.org/10.1145/2185520.2185539

Lucas Mourot, Ludovic Hoyet, François Le Clerc, François Schnitzler, and Pierre Hellier. 2022. A Survey on Deep Learning for Skeleton-Based Human Animation. *Comput. Graph. Forum* 41, 1 (2022), 122–157. https://doi.org/10.1111/cgf.14426

M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. 2007. *Documentation Mocap Database HDM05.* Technical Report CG-2007-2. Universität Bonn.

OpenAI. 2023. ChatGPT. https://chat.openai.com.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)* (San Francisco, CA, USA) *(UIST '23).* Association for Computing Machinery, New York, 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* 37, 4 (July 2018), 143:1–143:14.

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

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

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

Mathis Petrovich, Michael J. Black, and Gül Varol. 2021. Action-Conditioned 3D Human Motion Synthesis with Transformer VAE. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021.* IEEE, 10965–10975. https://doi.org/10.1109/ICCV48922.2021.01080

Mathis Petrovich, Michael J. Black, and Gül Varol. 2022. TEMOS: Generating Diverse Human Motions from Textual Descriptions. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXII (Lecture Notes in Computer Science, Vol. 13682)*, Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 480–497. https://doi.org/10.1007/978-3-031-20047-2_28

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

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

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2022. Progressive Neural Networks. https://doi.org/10.48550/arXiv.1606.04671 arXiv:1606.04671 [cs]

Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. 2020. PhysCap: physically plausible monocular 3D motion capture in real time. *ACM Transactions on Graphics* 39 (11 2020), 1–16. https://doi.org/10.1145/3414685.3417877

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. arXiv:2303.11366 [cs.AI]

Russ Smith. 2004. Open Dynamics Engine. https://ode.org/. Accessed 2023.09.01.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning.* PMLR, 2256–2265.

Jie Tan, Karen Liu, and Greg Turk. 2011. Stable Proportional-Derivative Controllers. *IEEE Computer Graphics and Applications* 31, 4 (July 2011), 34–44.

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, Article 37, 9 pages. https://doi.org/10.1145/3588432.3591541

Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. 2023. Human Motion Diffusion Model. In *The Eleventh International Conference on Learning Representations.* https://openreview.net/forum?id=SJ1kSyO2jwu

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. arXiv:1609.03499 [cs]

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural Discrete Representation Learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17).* Curran Associates Inc., Red Hook, NY, USA, 6309–6318.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. 2020. UniCon: Universal Neural Controller For Physics-based Character Motion. *CoRR* abs/2011.15119 (2020). arXiv:2011.15119

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=_VjQlMeSB_J

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Transactions on Graphics* 39, 4 (July 2020), 33:33:1–33:33:12.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-Based Character Controllers Using Conditional VAEs. *ACM Trans. Graph.* 41, 4, Article 96 (Jul 2022), 12 pages.

Jungdam Won and Jehee Lee. 2019. Learning Body Shape Variation in Physics-Based Characters. *ACM Trans. Graph.* 38, 6, Article 207 (nov 2019), 12 pages.

Zhaoming Xie, Sebastian Starke, Hung Yu Ling, and Michiel van de Panne. 2022. Learning Soccer Juggling Skills with Layer-Wise Mixture-of-Experts. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22).* Association for Computing Machinery, New York, NY, USA, Article 25, 9 pages. https://doi.org/10.1145/3528233.3530735

Chen Xin, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, Jingyi Yu, and Gang Yu. 2023. Executing your Commands via Motion Diffusion in Latent Space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

Pei Xu and Ioannis Karamouzas. 2021. A GAN-Like Approach for Physics-Based Imitation Learning and Interactive Character Control. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4, 3 (Sept. 2021), 44:1–44:22.

Pei Xu, Xiumin Shang, Victor Zordan, and Ioannis Karamouzas. 2023. Composite Motion Learning with Task Control. *ACM Trans. Graph.* 42, 4, Article 93 (jul 2023), 16 pages. https://doi.org/10.1145/3592447

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *CoRR* abs/2305.10601 (2023). https://doi.org/10.48550/arXiv.2305.10601 arXiv:2305.10601

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=WE_vluYUL-X

Sheng Ye, Yu-Hui Wen, Yanan Sun, Ying He, Ziyang Zhang, Yaoyuan Wang, Weihua He, and Yong-Jin Liu. 2022. Audio-Driven Stylized Gesture Generation with Flow-Based Model. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part V (Lecture Notes in Computer Science, Vol. 13665)*, Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 712–728. https://doi.org/10.1007/978-3-031-20065-6_41

KangKang Yin, Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2008. Continuation Methods for Adapting Simulated Skills. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–7. https://doi.org/10.1145/1360612.1360680

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

Zhiqi Yin, Zeshi Yang, Michiel Van De Panne, and Kangkang Yin. 2021. Discovering Diverse Athletic Jumping Strategies. *ACM Trans. Graph.* 40, 4, Article 91 (jul 2021), 17 pages. https://doi.org/10.1145/3450626.3459817

Mohamed Younes, Ewa Kijak, Richard Kulpa, Simon Malinowski, and Franck Multon. 2023. MAAIP: Multi-Agent Adversarial Interaction Priors for Imitation from Fighting Demonstrations for Physics-Based Characters. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 3, Article 32 (aug 2023), 20 pages. https://doi.org/10.1145/3606926

Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. 2022. PhysDiff: Physics-Guided Human Motion Diffusion Model. *CoRR* abs/2212.02500 (2022). https://doi.org/10.48550/arXiv.2212.02500 arXiv:2212.02500

Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. 2021. SimPoE: Simulated Character Control for 3D Human Pose Estimation. In *Proceedings of (CVPR) Computer Vision and Pattern Recognition.*

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. SoundStream: An End-to-End Neural Audio Codec. arXiv:2107.03312 [cs.SD]

Haotian Zhang, Ye Yuan, Viktor Makoviychuk, Yunrong Guo, Sanja Fidler, Xue Bin Peng, and Kayvon Fatahalian. 2023a. Learning Physically Simulated Tennis Skills from Broadcast Videos. *ACM Trans. Graph.* 42, 4, Article 95 (jul 2023), 14 pages. https://doi.org/10.1145/3592408

Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Shaoli Huang, Yong Zhang, Hongwei Zhao, Hongtao Lu, and Xi Shen. 2023b. T2M-GPT: Generating Human Motion from Textual Descriptions with Discrete Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. 2022. MotionDiffuse: Text-Driven Human Motion Generation with Diffusion Model. *arXiv preprint arXiv:2208.15001* (2022).

A NETWORK ARCHITECTURE

Table 5. Architecture of our Encoder and Deconv Module.

Encoder

(1) Conv1D(input_dim, 768, kernel size=(3,), stride=(1,), padding=(1,))
(2) 2 × Sequential(
    (a) Conv1D(768, 768, kernel size=(4,), stride=(2,), padding=(1,))
    (b) Resnet1D(
        3 × ResConv1DBlock(
            (activation1): ReLU()
            (conv1): Conv1D(768, 768, kernel size=(3,), stride=(1,), padding=(1,), dilation=(1,))
            (activation2): ReLU()
            (conv2): Conv1D(768, 768, kernel size=(1,), stride=(1,))
)
)
(3) Conv1D(768, output_dim, kernel size=(3,), stride=(1,), padding=(1,))

Deconv

(1) Conv1D(input_dim, 768, kernel size=(3,), stride=(1,), padding=(1,))
(2) 2 × Sequential(
    (a) Resnet1D(
        3 × ResConv1DBlock(
            (activation1): ReLU()
            (conv1): Conv1D(768, 768, kernel size=(3,), stride=(1,), padding=(1,), dilation=(1,))
            (activation2): ReLU()
            (conv2): Conv1D(768, 768, kernel size=(1,), stride=(1,))
)
)
    (b) ConvTranspose1D(768,768,kernel size=(4,),stride=(2,),padding=(1,))
(3) Conv1D(768, 768, kernel size=(3,), stride=(1,), padding=(1,))
(4) OutputHead1D(
    3 × ResConv1DBlock(
        (activation1): ReLU()
        (conv1): Conv1D(768, 768, kernel size=(3,), stride=(1,), padding=(1,), dilation=(1,))
        (activation2): ReLU()
        (conv2): Conv1D(768, 768, kernel size=(1,), stride=(1,))
)
)
(5) Conv1D(768, output_dim, kernel size=(3,), stride=(1,), padding=(1,))

B RESPOND OF CLAUDE-2

B.1 Example of text2motion

- **User:**
  Question: ’a person walks forward for a long time and kicks, then he begins to dance’
- **Claude-2:**
  Okay, let’s see if I can interpret this sequence properly:
  - To capture walking forward for a long time, I would use a repetitive walking sequence:
    297, 471, 246, 463, 463, 297, 471, 246, 463, 463, 297, 471, 246, 463, 463, 297, 471, 246, 463, 463
  - Then to encode the kicking motion:
    145, 160, 223, 117, 424, 299, 178, 166, 330, 391
  - And finally some dancing motions like:
    472, 133, 132, 357, 492, 435, 217, 322, 106, 88, 279, 217, 387, 274, 134, 470, 175, 466, 252, 156, 134, 352, 436, 394, 77, 379, 297, 315, 185, 112, 428, 206, 98, 14, 390, 390, 390, 14, 196, 395, 334
  - The full sequence would be:
    297, 471, 246, 463, 463, 297, 471, 246, 463, 463, 297, 471, 246, 463, 463, 297, 471, 246, 463, 463, 145, 160, 223, 117, 424, 299, 178, 166, 330, 391, 472, 133, 132, 357, 492, 435, 217, 322, 106, 88, 279, 217, 387, 274, 134, 470, 175, 466, 252, 156, 134, 352, 436, 394, 77, 379, 297, 315, 185, 112, 428, 206, 98, 14, 390, 390, 390, 14, 196, 395, 334

  Let me know if this matches the description of walking for a long time, kicking, and then beginning to dance or if I should modify the sequence in any way. Please feel free to provide any corrections to help me improve!

B.2 Example of Abstract Task

- **User:**
  Question: Suppose there is a door in front of you and a key on the ground. you want to open the door and sit on the chair, how do you perform? Please describe how you get index sequences for each action.
- **Claude-2:**
  Here are the steps I would take to complete the task, along with the relevant motion index sequences:
  - Walk forward to reach the key on the ground: [297, 471, 246, 463, 463] I selected this "walk forward" motion sequence from the example "a person walks forward".
  - Bend down and pick up the key: [246, 192, 408, 205, 205] I selected this "bend down" motion from the example "a person bends down to pick something up".
  - Stand back up straight: [205, 198, 408] I selected this "stand up" motion from the same "bend down" example.
  - Turn around and walk back towards the door: [463, 297, 471, 246] I reused the "walk forward" motion but in reverse order to walk back.
  - Insert the key into the door lock and turn it: [129, 129, 433] I selected this "insert and turn" motion from the example "a person uses knife to cut object".
  - Push the door open: [179, 179, 179] I selected this "push" motion from the example "a person pushes something with their right hand".
  - Walk through the doorway: [297, 471, 246, 463] Reused the "walk forward" motion to walk through the door.
  - Sit down on chair: [241, 241, 241] I selected this "sit down" motion from the example "a person sits down and crosses their legs".

  By selecting relevant motion sequences from the examples and piecing them together in order, I was able to describe the full sequence of actions needed to open the door, enter the room, and sit on the chair inside. Let me know if you need any clarification on my approach!