A Other Related Work

Human Motion Synthesis. Human motion synthesis aims at generating natural motions consistent with any signal that describes the motion, such as action category (Guo et al., 2020; Petrovich et al., 2021), text (Guo et al., 2022; Tevet et al., 2023), music (Tseng et al., 2023; Alexanderson et al., 2023) and historical pose sequences (Mao et al., 2019; Li et al., 2020). With these guidance, earlier works (Ahuja & Morency, 2019; Ghosh et al., 2021) focus on learning a shared latent space for motion and conditions deterministically, limiting one-to-one mapping from condition to motion. Some recent works have put more emphasis on the promotion of diversity, and learned to model the distribution of motions based on the development of deep generative models, like Variational AutoEncoders (VAEs) (Kingma & Welling, 2013) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). BiHMP-GAN (Kundu et al., 2019), conditioned on a given starting sequence, uses the discriminator of GANs to regress the random vector for multiple probable predictions. (Cai et al., 2021) presents a VAE-based unified framework for generalized motion synthesis that covers motion prediction, completion, interpolation and recovery. Wang et al. (2022) involves the modeling of human-scene interaction, path planning and body movement, to implement motion generation in the given scene environment with target action sequence. ACTOR (Petrovich et al., 2021) and TEMOS (Petrovich et al., 2022) suggest employing a VAE to map action labels and texts into a variational distribution with transformer structure (Vaswani et al., 2017), respectively.

However, all the aforementioned methods primarily focus on fixed-size motion modeling, and do not take into consideration the aspect of motion framerate. As a result, higher-framerate details and lower-framerate global structures in datasets are disregarded. Moreover, these methods are constrained to generating motions of fixed sizes. In contrast, our proposed method addresses these limitations by incorporating available varied framerates into considerations and enabling the generation of motions at arbitrary framerates.

Motion Representation. Motion representation is very crucial for the subsequent synthesis task. There existing several motion representation on different datasets, such as (1) the classical SMPL-based motion parameters (Petrovich et al., 2021), (2) the redundant hand-crafted motion features (Guo et al., 2022), and (3) straight-forward joint positions (Mao et al., 2019). We opt for the first two representations in this paper. Particularly, The first one is widely used in motion capture, and the second one is mainly used in character animation. Following (Tevet et al., 2023; Chen et al., 2023), we employ the SPML parameters in action-to-motion task for a fair comparison, and redundant hand-crafted features in text-to-motion and unconditional tasks.

B More Details

B.1 Implementation

Our NeRM are decomposed into two stages, including INR and latent diffusion. For INR, the hidden layer size is fixed to 1,024. We use the pre-trained codebook of (Zhang et al., 2023a), which is trained by VQ-VAE (Van Den Oord et al., 2017) on the HumanML3D dataset. The codebook size is set to 512×512. The number of learnable query embeddings of codebook-coordinate attention is 256, and the dimension of each embedding is 128. We employ a frozen *CLIP-ViT-L-14* model as our text encoder for text descriptions, and a learnable embedding for action categories. The shape of latent codes $\boldsymbol{z}$ is set to 256, which is then injected into condition by concatenation for diffusion training and inference. Our models are trained with the AdamW optimizer using a fixed learning rate of 10^{-4}. Our batch size is set to 4,096 during the INR training stage and 64 during the diffusion training stage separately. Since INR requires learning a latent code for each training sample, we set its batch size large for efficient training. Besides, INR model was trained for 20,000 epochs and diffusion model was trained for 3,000 epochs. The number of diffusion steps is 1,000 during training while 50 during inference. The corresponding variances β_k in diffusion are scaled linearly from 8.5×10^{-4} to 0.012. We train our models under Pytorch on NVIDIA GeForce RTX 3090.

B.2 Auto-decoding of Latent Code

Unlike traditional auto-encoder whose latent code is produced by the encoder, we draw inspiration from DeepSDF Park et al. (2019) in 3D shapes and use an auto-decoder to learn the latent code

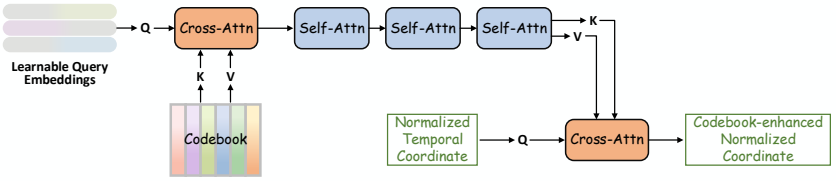

Figure 6: Detailed network architecture of Codebook-Coordinate Attention (CCA).

without an encoder. This encoder-less design avoids explicit modeling of raw motions, making arbitrary-framerate training feasible. To construct distributions where one can sample a representation to generate a new high-quality motion, we gain insight from Variational Auto-Encoder (VAE) that utilizes an encoder to infer a distribution from which representations of motions can be sampled, and employs a decoder to reconstruct the data from the representation. As our design does not contain an encoder, we model each motion by sampling from a learnable distribution with optimized parameters, i.e., mean μ and covariance Σ, leading to learnable parameters $\{\mu_i, \Sigma_i\}_{i=1}^{n}$ for all training dataset. When training, we sample a latent representation $\boldsymbol{z}_i$ from a distribution $\mathcal{N}(\mu_i, \Sigma_i)$. We then combine $\boldsymbol{z}_i$ with temporal coordinate encodings $\gamma(t)$, and input them into a shared MLP decoder f_θ. The parameters μ_i, Σ_i and θ are optimized simultaneously by minimizing the reconstruction loss between the generated and ground truth.

B.3 Synthesizing Long Sequences

Ideally, NeRM is possible to generate motions with arbitrary framerates s and durations l by setting appropriate temporal coordinates. However, either significant increase of s or l requires significant memory resources. Thus, we employ an iterative-synthesis approach to generate non-overlapping motion clips and assemble them into longer motions. By leveraging clip-based multi-framerate training conditioned on temporal coordinates, we learn continuous motion fields and ensure smooth transitions between motion segments. Notably, the framerate range we support is still limited by the original training data. The model is unlikely to learn motion patterns that exceed the highest framerate present in the training data.

C Details of Evaluation Metrics

C.1 Evaluation Metrics

Frechet Inception Distance (FID). FID is widely used for overall generative quality evaluation. FID is calculated by extracting features from 1,000 generated motions and real motions in test set. Instead of the inception neural network in image domain, we extract a deep representation of the motion with the evaluator network Guo et al. (2022), as suggested by Tevet et al. (2023).

R-Precision (Top-3). This metric is used for text-motion matching measurement. For each generation, its corresponding GT description and randomly selected 31 mismatched descriptions are gathered in a pool. Then the Euclidean distance between the motion feature and each text feature in the pool is calculated, where accuracy of Top 3 are picked. The GT description falling into the Top 3 candidates represents a successful retrieval.

MultiModality Dist. Like R-Precision, this metric also measures text-motion similarity. We calculate the average Euclidean distance between the motion feature of each generation and text feature of the corresponding description.

Diversity. Diversity measures generative variance across all motions. We randomly select motions from all generations, and put each of them in either of the two subsets $\{q_1, \ldots, q_{D_a}\}$ and $\{q'_1, \ldots, q'_{D_a}\}$, with the same size as $D_a = 300$, where q_i indicates a motion feature vector. Diversity can be expressed as:

$$\text{Diversity} = \frac{1}{D_a} \sum_{i=1}^{D_a} \|q_i - q'_i\|_2. \tag{7}$$

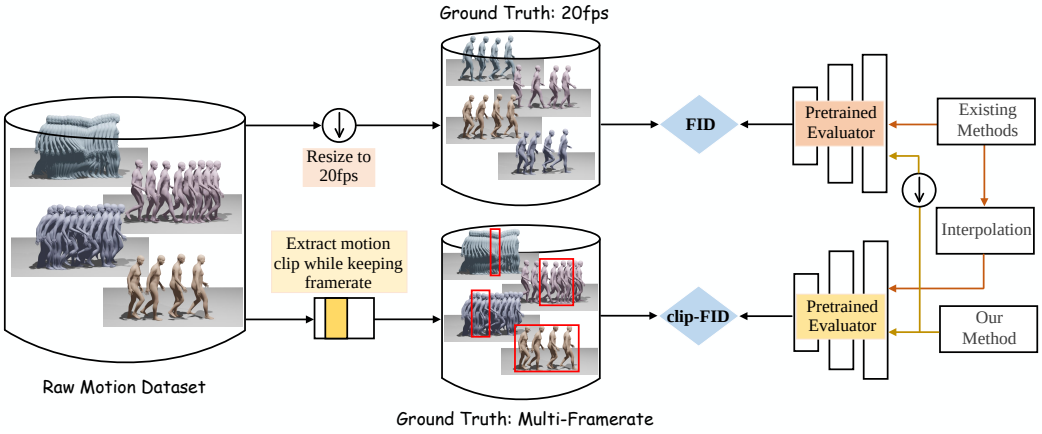

Figure 7: Comparisons of conventional FID and our clip-FID. FID evaluates global structure, but downsamples all human motions to a common 20fps which ignores high-framerate details. In contrast, clip-FID takes motion clips instead, thereby keeping the original framerates. We employ both metrics to validate the effectiveness of our method.

Table 4: Mean reconstruction errors of MLD and NeRM for motion of different framerates.

	Framerates (fps)				
Method / Metric	20	40	60	100	120
MLD / MPJPE	0.027	0.062	0.113	0.184	0.228
NeRM / MPJPE	**0.016**	**0.019**	**0.013**	**0.019**	**0.011**
MLD / MRE	0.074	0.105	0.194	0.254	0.387
NeRM / MRE	**0.041**	**0.036**	**0.034**	**0.035**	**0.038**

MultiModality (MM). MM measures the generated motions diversify within each condition (text or action). We randomly pick S text descriptions from all descriptions. Then the motions generated by the s-th description are randomly picked and put into one of the two subsets: $\{q_{s,1}, \ldots, q_{s,D_e}\}$ or $\{q'_{s,1}, \ldots, q'_{s,D_e}\}$, with subset size $D_e = 10$. MultiModality can be expressed as:

$$\text{MM} = \frac{1}{S \times D_e} \sum_{s=1}^{S} \sum_{i=1}^{D_e} \|q_{s,i} - q'_{s,i}\|_2. \tag{8}$$

Accuracy. We employ the pre-trained action recognition model to classify 1,000 generate motions. The obtained overall recognition accuracy demonstrates the correlation of the motion and its action label.

C.2 Motion Clip-based FID (clip-FID).

All the metrics above are designed for low-framerate data, and unable to be trained on original high-framerate data due to prohibitive memory requirements. Thus, we here present a new metric clip-FID that is aimed at better capturing the realism of details at high framerate by avoiding downsampling. As depicted in Figure 7, we show the comparison of standard FID and clip-FID, where FID evaluates global structure **(the coarse information of the entire motion is preserved)**, but downsamples all human motions to a common 20fps, ignoring high-framerate details; clip-FID takes motion clips extracted/cropped from motions **(the detailed information of the motion clip is preserved)**, keeping the original framerates. Note that we randomly sample clips (a short segment of motion) of size m from real motions at framerate s and center $\boldsymbol{v}$ to generate the corresponding clip. A lower value implies better high-framerate details.

Table 5: Ablation study on effectiveness of time encoding.

Simple	Codebook	Motion Reconstruction (MRE)					Motion Synthesis (clip-FID)				
		20	40	60	100	120	20	40	60	100	120
✗	✗	0.134	0.141	0.164	0.094	0.091	0.471	0.803	1.070	1.769	2.944
✓	✗	0.053	0.049	0.039	0.057	0.043	0.397	0.519	0.701	0.142	1.717
✗	✓	**0.041**	**0.036**	**0.034**	**0.035**	**0.038**	**0.389**	**0.493**	**0.680**	**0.903**	**1.315**

Table 6: Ablation study on effectiveness of Variational INRs.

Variational INRs	Motion Reconstruction (MRE)					Motion Synthesis (clip-FID)				
	20	40	60	100	120	20	40	60	100	120
✗	**0.032**	**0.030**	**0.031**	**0.036**	**0.027**	1.280	2.924	7.012	10.482	14.654
✓	0.041	0.036	0.034	0.035	0.038	**0.389**	**0.493**	**0.680**	**0.903**	**1.315**

D Network Architecture

Codebook-Coordinate Attention (CCA). Inspired by CoCo-NeRF (Yin et al., 2022), we apply CCA modulation to enrich the Fourier features of each coordinate. The detailed architecture is illustrated in Figure 6. Specifically, we employ one cross-attention block to learn dependency between learnable query embeddings $\mathcal{Q} = \{q_i\}_{i=1}^{M}$ and codebook prototypes $\mathcal{E} = \{e_i\}_{i=1}^{N}$. Then, the embeddings are fed into self-attention blocks with three layers to improve their feature representations further and obtain the final motion-relevant prototypes $\hat{\mathcal{Q}}$. Finally, we conduct once cross-attention operation between $\hat{\mathcal{Q}}$ and Fourier embeddings of each coordinate $\gamma(t)$. All attention modules are based on transformer (Vaswani et al., 2017) with 4 head Attention mechanism, Layer Normalization, Feed-Forward Network and GELU activation.

MLP Decoder. Similar to other neural representations (Ashkenazi et al., 2023), the NeRM decoder f_θ is constructed using a simple neural network architecture. It consists of a 9-layer MLP with ReLU activations and layer normalization. The hidden layer size remains constant across the network. We also incorporate residual connections within each layer to improve gradient flow. In contrast to previous approaches (Chen et al., 2023; Tevet et al., 2023) that rely on a transformer backbone, our choice of a simple decoder offers benefits in terms of inference speed for generating new motions.

Latent Denoiser. Different from the UNet-based architecture (Ronneberger et al., 2015) in latent diffusion model that designed for image synthesis, our latent denoiser ϵ_ϕ is built ViT backbone with long skip connections (Bao et al., 2023), which is more appropriate for time series data.

E Additional Experiments

Our NeRM consists of an INR model f_θ and a latent diffusion model ϵ_ϕ. We conduct additional experiments to evaluate the effectiveness of each component, as well as reconstruction capability. We also report time costs on inference and GPU memory on multi-framerate training dataset.

E.1 Reconstruction Capability

We first compare the reconstruction capability of MLD (Chen et al., 2023) and our NeRM with different framerates. Note that MLD learns latent representation via Variational AutoEncoder (VAE) based on transformer encoder. We randomly select 100 different motion sequences with their original framerate on HumanML3D dataset, and report the average reconstruction errors in Table 4. Since MLD cannot process multi-framerate dataset, we use spherical linear interpolation to generate higher-framerate motions for MLD. For evaluation metrics, we make use of the Mean Per Joint Position Error (MPJPE) (Mao et al., 2019), commonly used for image-based 3D human pose estimation. We also employ Mean Redundant Error (MRE) that measures the Euclidean distance between two

Table 7: Ablation study on decoder architecture.

MLP	Transformer	Motion Reconstruction (MRE)					Motion Synthesis (clip-FID)				
		20	40	60	100	120	20	40	60	100	120
✗	✓	**0.037**	**0.032**	0.037	**0.035**	**0.037**	**0.381**	**0.487**	**0.607**	**0.782**	**1.199**
✓	✗	0.041	0.036	**0.034**	**0.035**	0.038	0.389	0.493	0.680	0.903	1.315

Table 8: Ablation study on the dimension d and weight parameter λ_{KL} of latent representation.

d	128	256	512
MRE@20fps	0.0459	0.0412	**0.0407**
λ_{KL}	1e-5	1e-4	1e-3
MRE@20fps	**0.0410**	0.0412	0.0459

poses represented by redundant hand-crafted features. From the results in Table 4, we find that our NeRM with a simple MLP can achieve lower reconstruction error with 20 fps than MLD with complicated transformer backbone. When the framerate increases, the performance of MLD deteriorates significantly while NeRM still maintains very low reconstruction error.

E.2 Ablation Study

In this section, we provide ablation studies on the HumanML3D dataset to evaluate the effectiveness of network design.

Effectiveness of Time Encoding. Time encoding plays a critical role in the generalization. In Table 5, we evaluate the influence of different encoding manners, including No Time Encoding, Simple Time Encoding, and Codebook-Coordinate Encoding. Here, "Simple" means that we only use Fourier features to encode t like traditional NeRF (Mildenhall et al., 2021); and "Codebook" is our design of NeRM. From the table, we find that the model without time encoding achieves poor performance in both terms of motion reconstruction and motion synthesis. This can be attributed to "spectral bias" (Rahaman et al., 2019). In other words, INRs with simple MLP layers cannot learn high-frequency variations from motion data. Our codebook-based representation reaches the best performance, which confirms that the codebook is beneficial for feature representation of temporal coordinates.

Effectiveness of Variational INRs. We investigate the influence of variational INR by comparing a variational INR with a non-variational version. In this case, the latent code $\boldsymbol{z}$ of non-variational INR is obtained by optimizing the following:

$$\boldsymbol{z}_i^* = \arg\min_{\boldsymbol{z}_i} \|\hat{x}_{clip}^i - x_{clip}^i\|^2, \quad \text{for } i = 1, 2, \cdots, n \tag{9}$$

where $\hat{x}_{clip}^i = f_\theta(t_{\boldsymbol{v},s}, s)$. Comparison results are shown in Table 6. We observe that the reconstruction errors of non-variational INRs is smaller than variational ones, as non-variational INRs may overfit the motion sequence by powerful neural network. However, the realism of non-variational INRs is significantly improved by the variational ones. This suggests that the variational approach strongly regularizes the latent space and enhances the capability of sampling new motions.

Decoder Architecture. Furthermore, we compare the performance of MLP-based and transformer-based models. Table 7 shows that the self-attention mechanism can slightly improve the ability of motion synthesis. However, the latent representation learned by diffusion model is fed to the decoder of our variational INR, which implies that the complexity of the network greatly impacts the generation speed of new motions. Therefore, we pick MLP layers as our decoder architecture.

Effectiveness of latent representation. The latent representation $\boldsymbol{z}$ is a crucial variable in our NeRM. It acts as a bridge between the variational INR model f_θ and the diffusion model ϵ_ϕ. In our context, the quality of latent representation can be influenced by the dimension and the variational distribution $\mathcal{N}(\mu, \Sigma)$ of $\boldsymbol{z}$. Therefore, we investigate the impact of different dimensions d of $\boldsymbol{z}$ and weight λ_{KL} on the KL loss for motion reconstruction in Table 8. When the dimension d is set to

Table 9: Ablation study on effectiveness of time normalization.

Time Normalization	MRE@20fps	FID	Diversity
✗	0.118	0.958	9.892
✓	**0.041**	**0.389**	**9.547**

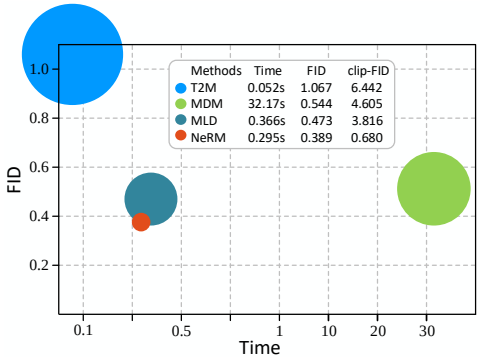

Figure 8: Average inference time (seconds) for generating one motion sequence. The circle size is proportional to the value of clip-FID. Bigger circle indicates worse performance of high-framerate details.

256, the motion reconstruction loss becomes significantly small, and higher dimension can slightly improve the reconstruction ability. For λ_{KL}, a higher weight results in a smoother latent space but increases the reconstruction error. We set λ_{KL} to a small value (i.e., 0.0001) like Chen et al. (2023).

Effectiveness of time normalization. In this section, we explore the effect of time normalization. Table 9 shows the results on HumanML3D dataset with 20 fps. Notably, '✗' indicates that we directly use the true temporal coordinates, i.e., $\{1, 2, \cdots, T\}$, instead of normalized temporal coordinates $t_{v,s}$. From the table, we find that time normalization plays a vital role in time encoding of implicit neural representation.

E.3 Inference Time

In Figure 8, we report average inference time on per sequence. As T2M (Guo et al., 2022) is built under VAEs, it uses the least time for generation; but under diffusion setting, we are the most time-saving. To be specific, due to the latent diffusion design, MLD (Chen et al., 2023) and our NeRM are much faster than MDM (Tevet et al., 2023). We are even faster than MLD in that we use simple MLP decoder of INR, rather than the transformer-based decoder in MLD. We yield the best FID for best global motion quality, and significant superiority in clip-FID (the target framerate is set to 60 fps), indicated by the smallest circle size. All of these experiments are conducted on NVIDIA GerForce RTX 3090.

E.4 Memory Burden

By padding zeros, current text-to-motion generative models are able to train motions with the same framerate s (20 fps) and different duration $2 \leq l \leq 9.8$ (seconds). When exploiting native framerates of motions, one possible solution is to padding zeros according to the maximum framerates (250 fps). However, this operation cannot capture accurate temporal dependency as fixed-framerate training. Another problem is that the dimensionality of padded motions needs to be much larger than what state-of-the-art diffusion models can be trained on. For example, MDM (Tevet et al., 2023) trains the diffusion model on motions where the maximum number of poses is 196 ($9.8s \times 20fps$), while we use motions of maximum size 2450 ($9.8s \times 250fps$). This imposes a substantial burden on GPU memory. Alternatively, we approach it from the perspective of implicit neural representation, which enables us to process *clips* (a short segment of motion) with size m ($m \ll 2450$), therein significantly reducing the memory burden. In addition, we find a representative latent code for each

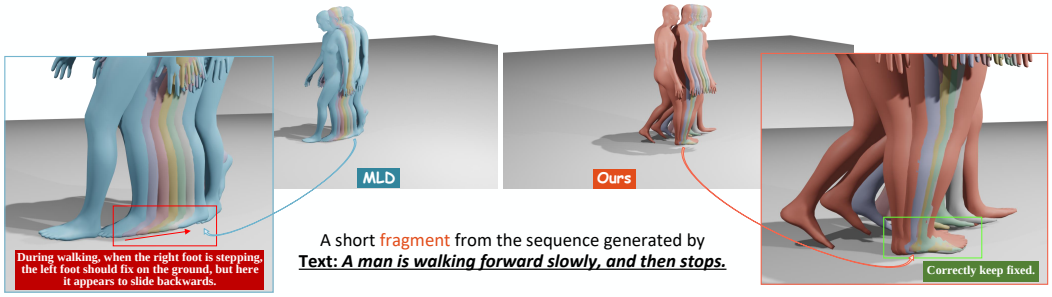

Figure 9: High-framerate generation (120 fps). We visualize a short fragment from the whole generation, with purple, pink, yellow, green, and grey sequentially indicating high-framerate motion changing. The entire generation can be found in our video.

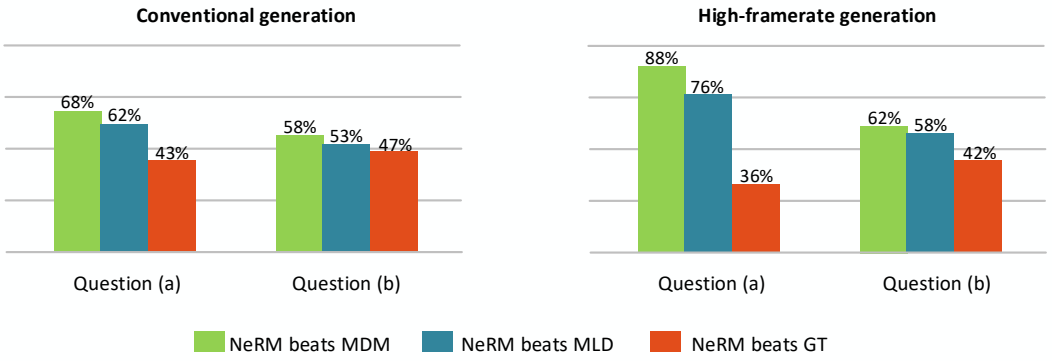

Figure 10: User study on HumanML3D dataset.

motion and learn the distribution in this low-dimensional latent space efficiently, which decouples the modeling of the distribution from varied-size human motions.

E.5 Visualization

A video is contained in our supplement. We provide 120 fps generation on two more examples, where our NeRM constantly outperforms MLD (Chen et al., 2023) in both terms of basic motion quality and high-framerate quality. Note that, as MLD cannot directly generate high-framerate motions, we use SLERP to interpolate them towards the target framerate (120 fps). We also select a short fragment from our video with full discussion shown in Figure 9. Additionally, for clearer observation on artifacts of baseline, we slow down the video by 6 times, so the motions may appear to be slower.

E.6 User Study

Human eyes are the ultimate evaluation for human motion performance. We asked 17 people over question (a) "Which of the two motions is more realistic" and (b) "Which of the two motions is more consistent with the given texts". In each *two-motion* pair, we provide one motion generated by NeRM and the other by baselines MDM (Tevet et al., 2023), MLD (Chen et al., 2023) and GT from HumanML3D. We randomly pick 8 cases for each question. We evaluate on conventional 20 fps generation and high-framerate generation (each of 120, 100, 60 and 40 fps has 2 cases). Shown in Figure 10, our NeRM gains more preference than baselines, and even comparable to GT motions.

F LIMITATIONS AND FUTURE WORK

Although our method has achieved promising performance for high-framerate motion synthesis, it still has the following challenges. (1) While our method currently supports generation with external conditions such as action labels or text prompts, it has limitations in incorporating fine-grained internal conditions such as keyframes or trajectory. A promising direction for future research is to design a more comprehensive framework (as exemplified by Karunratanakul et al. (2023); Zhang et al. (2023b)) that can simultaneously consider both external and internal conditions. By integrating these factors, we can achieve more precise and nuanced motion generation. (2) The quality of motion generated using INRs is highly dependent on the dataset. If the dataset lacks high-framerate data, the performance of generating high-framerate motions may not be optimal. Additionally, the model is unlikely to learn motion patterns that exceed the highest framerate present in the training data. (3) While our method demonstrates fast inference time, the training process can be relatively slow, particularly when dealing with the dataset containing numerous samples. This is because our method learns a latent code for each training sample. (4) Our method is designed for motion modeling and can not adapt to outputs with varying body shapes. Some recent works Mihajlovic et al. (2022) have employed INRs to model skinned articulated objects with specific poses. It would be interesting to explore the integration of our method with them.