MMM: Generative <u>M</u>asked <u>M</u>otion <u>M</u>odel

Ekkasit Pinyoanuntapong, Pu Wang, Minwoo Lee

University of North Carolina at Charlotte

{epinyoan, pwang13, Minwoo.Lee}@uncc.edu

Chen Chen

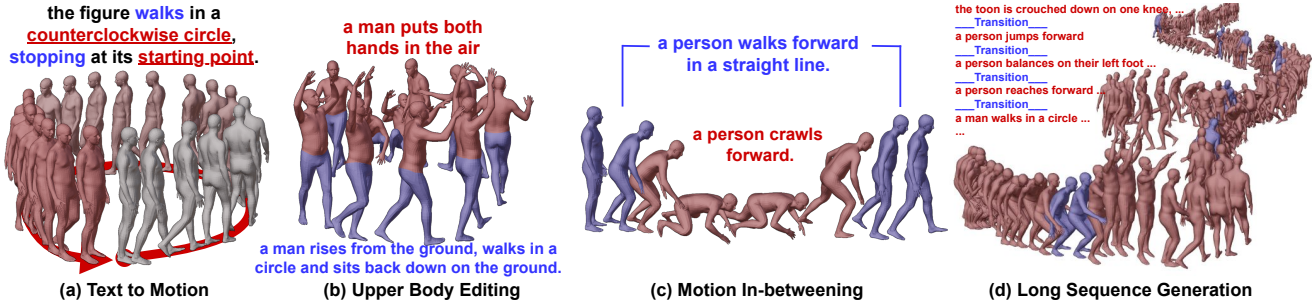University of Central Florida

chen.chen@crcv.ucf.edu

Figure 1. MMM can generate precise human motions given fine-grained textual descriptions while enabling motion editing applications. Blue frames represent conditioned motion input and red frames are the generated motion. Motion in-betweening can be performed by filling the gaps between keyframes or major motion points, conditioned on text or without conditions. Upper body part editing is done by fixing the lower body motion generated by one text prompt and altering the upper body motion according to another input prompt. An arbitrary long motion sequence can be generated according to a story (i.e., a sequence of text prompts), where MMM generates the motion for each prompt (red frames), while "hallucinating" the nature and smooth motion transitions (blue frames) between neighboring prompts (without being explicitly trained on motion transition datasets)

Abstract

Recent advances in text-to-motion generation using diffusion and autoregressive models have shown promising results. However, these models often suffer from a trade-off between real-time performance, high fidelity, and motion editability. To address this gap, we introduce MMM, a novel yet simple motion generation paradigm based on Masked Motion Model. MMM consists of two key components: (1) a motion tokenizer that transforms 3D human motion into a sequence of discrete tokens in latent space, and (2) a conditional masked motion transformer that learns to predict randomly masked motion tokens, conditioned on the precomputed text tokens. By attending to motion and text tokens in all directions, MMM explicitly captures inherent dependency among motion tokens and semantic mapping between motion and text tokens. During inference, this allows parallel and iterative decoding of multiple motion tokens that are highly consistent with fine-grained text descriptions, therefore simultaneously achieving high-fidelity and high-speed motion generation. In addition, MMM has innate motion editability. By simply placing mask tokens in the place that needs editing, MMM automatically fills the gaps while guaranteeing smooth transitions between editing and non-editing parts. Extensive experiments on the HumanML3D and KIT-ML datasets demonstrate that MMM surpasses current leading methods in generating high-quality motion (evidenced by superior FID scores of 0.08 and 0.429), while offering advanced editing features such as body-part modification, motion in-betweening, and the synthesis of long motion sequences. In addition, MMM is two orders of magnitude faster on a single mid-range GPU than editable motion diffusion models. Our project page is available at https://exitudio.github.io/MMM-page/.

1. Introduction

Text-driven human motion generation has recently become an emerging research focus due to the semantic richness and user-friendly nature of natural language descriptions, with its broad applications in animation, film, VR/AR, and robotics. However, generating high-fidelity motion that precisely aligns with text descriptors is challenging because of inherent differences between language and motion data distributions. To address this challenge, three predominant methods have been proposed, including (1) language-motion latent space alignment, (2) conditional diffusion model, and (3) conditional autoregressive model.

In the first method, text descriptions and motion se-

quences are projected into separate latent spaces, forcibly aligned by imposing distance loss functions such as cosine similarity and KL losses [1, 12, 22, 23, 34, 41]. Due to unavoidable latent space misalignment, this method falls short of achieving high-fidelity motion generation, which requires the synthesized motion to accurately reflect the fine-grained textural descriptions. To this end, conditional diffusion and autoregressive models are proposed recently [15, 17, 19, 35, 38, 43, 44, 47]. Instead of brutally forcing latent space alignment, these models learn a probabilistic mapping from the textural descriptors to the motion sequences. However, the improvement in quality comes at the cost of motion generation speed and editability.

Motion-space diffusion models learn text-to-motion mapping by applying diffusion processes to raw motion sequences conditioned on text inputs [17, 35, 38, 44]. The use of the raw motion data supports partial denoising on certain motion frames and body parts, naturally supporting semantic motion editing, such as motion inpainting and body part editing. However, the redundancy in raw data usually leads to high computational overhead and thus slow motion generation speed. A recent latent-space motion diffusion model [4] accelerates motion generation speed by compressing raw motion data into a single latent embedding. Nonetheless, this embedding hides rich temporal-spatial semantics present in the original motion data, hindering effective motion editing.
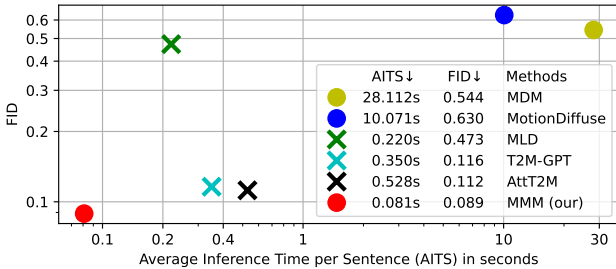

Figure 2. The motion generation quality (FID score) and speed (AITS) comparisons between MMM and SOTA methods on HumanML3D dataset. The model closer to the origin is better. MMM achieves the best FID score (0.08) and the highest speed (0.081 AITS), while preserving motion editability. "◯" represents editibility and "×" otherwise. All tests are performed on a single NVIDIA RTX A5000.

Compared with motion diffusion models, motion autoregressive models further improve motion generation fidelity by modeling temporal correlations within motion sequences [15, 43, 47]. Following a training paradigm similar to large language models such as GPT [2], motion autoregressive models learn to predict and generate the next motion token conditioned on the text token and previously generated motion tokens. This process, known as autoregressive decoding, contributes to improved coherence and accuracy in motion generation. However, this sequential and unidirectional decoding approach not only results in significant de-

lays in motion generation but also makes motion editing an extremely challenging, if not impossible, task.

As mentioned above, the existing text-driven motion generation models suffer from a trade-off between real-time performance, high fidelity, and motion editability. To address this critical issue, we introduce MMM, a novel motion generation paradigm based on conditional **M**asked **M**otion **M**odel. During the training phase, MMM follows a two-stage approach. In the first stage, a motion tokenizer is pretrained based on the vector quantized variational autoencoder (VQ-VAE) [37]. This tokenizer converts and quantizes raw motion data into a sequence of discrete motion tokens in latent space according to a motion codebook. A large-size codebook is learned to enable high-resolution quantization that preserves the fine-grained motion representations. In the second stage, a portion of the motion token sequence is randomly masked out, and a conditional masked transformer is trained to predict all the masked motion tokens concurrently, conditioned on both the unmasked ones and input text.

By attending to motion and text tokens in all directions, MMM explicitly captures inherent correlation among motion tokens and semantic mapping between motion and text tokens. This enables text-driven parallel decoding during inference, where in each iteration, the model concurrently and progressively predicts multiple high-quality motion tokens that are highly consistent with text descriptions and motion dynamics. This feature allows MMM to simultaneously achieve high-fidelity and high-speed motion generation. In addition, MMM has innate motion editability. By simply placing mask tokens in the place that needs editing, MMM automatically fills the gaps while ensuring smooth and natural transitions between editing and non-editing parts. Our contributions are summarized as follows.

- We introduce the generative masked motion model, a novel yet simple text-to-motion generation paradigm. This model departs from diffusion and autoregressive models that currently dominate motion generation tasks.
- We conduct extensive qualitative and quantitative evaluations on two standard text-to-motion generation datasets, HumanML3D [12] and KIT-ML [24]. It is shown in Figure 2 that our model outperforms current state-of-the-art methods in both motion generation quality and speed.
- We demonstrate that our model enables fast and coherent motion editing via three tasks: motion in-between, upper body modification, and long sequence generation.

2. Related Work

Text-driven Motion Generation. Early text-to-motion generation methods are mainly based on distribution alignment between motion and language latent spaces by applying certain loss functions, such as Kullback-Leibler (KL) divergences and contrastive losses. The representative meth-
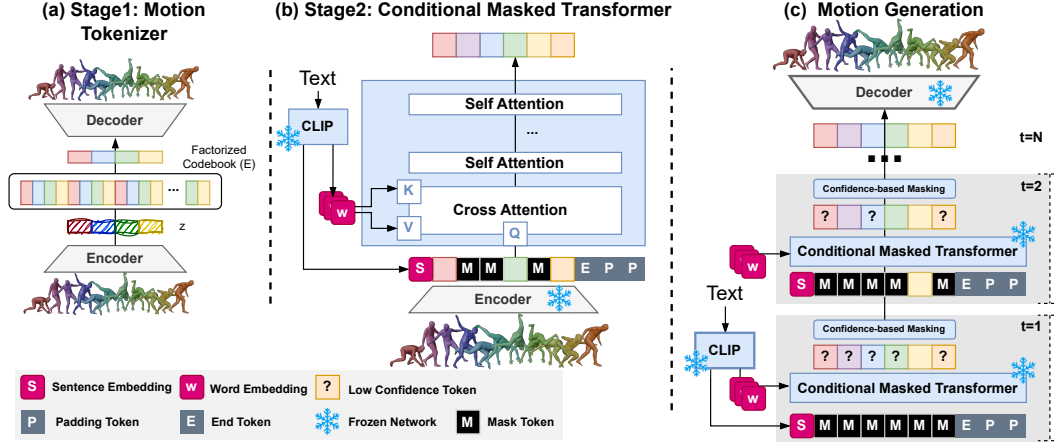
(a) Stage1: Motion Tokenizer

(b) Stage2: Conditional Masked Transformer

(c) Motion Generation

S Sentence Embedding	**W** Word Embedding	**?** Low Confidence Token
P Padding Token	**E** End Token	❄ Frozen Network **M** Mask Token

Figure 3. Overall architecture of MMM. **(a) Motion Tokenizer** transforms the raw motion sequence into discrete motion tokens according to a learned codebook. (b) **Conditional Masked Transformer** learns to predict masked motion tokens, conditioned on word and sentence tokens obtained from CLIP text encoders. **(c) Motion Generation** starts from an empty canvas and the masked transformer concurrently and progressively predicts multiple high-confidence motion tokens.

ods include Language2Pose [1], TEMOS [22], T2M [12], MotionCLIP [34], TMR [23] and DropTriple [41]. Due to the significant variations between text and motion distributions, these latent space alignment approaches generally lead to unsatisfied motion generation quality.

Since denoising diffusion models [33][13] have demonstrated notable success in vision generation tasks, [14, 21, 29, 32], diffusion models have been adopted for motion generation, where MDM [35], MotionDiffuse [44] and FRAME [17] are recent attempts. However, these methods directly process diffusion models in raw and redundant motion sequences, thus resulting in extremely slow during inference. In our experiments, MDM [35] takes 28.11 seconds on average to generate a single motion on Nvidia RTX A5000 GPU. Inspired by pixel-based latent diffusion models [28], motion-space diffusion models MLD [4] mitigates this issue by applying diffusion process in low-dimensional motion latent space. Nonetheless, relying on highly compressed motion embedding for speed acceleration, MLD struggles to capture fine-grained details, greatly limiting its motion editability.

Inspired by the success of autoregressive models in language and image generations, such as GPT [2], DALL-E [27] and VQ-GAN [7, 39, 42], autoregressive motion models, T2M-GPT [43], AttT2M [47] and MotionGPT [15], have been recently developed to further improve motion generation quality. However, these autoregressive models utilize the causal attention for unidirectional and sequential motion token prediction, limiting its ability to model bidirectional dependency in motion data, increasing the training and inference time, and hindering the motion editability. To address these limitations, we aim to exploit masked motion modeling for real-time, editable and high-fidelity motion generation, drawing inspiration from the success of BERT-like masked language and image model-

ing [3, 5, 6, 8, 25, 45, 46].

Motion Editing. MDM [35], FLAME [17], and Fg-T2M [38] introduce text-to-motion editing, demonstrating body part editing for specific body parts and motion in-between for temporal interval adjustments. They achieve this by adapting diffusion inpainting to motion data in both spatial and temporal domains. PriorMDM [30] uses a pre-trained model from MDM [35] for three forms of motion composition: long sequence generation (DoubleTake), two-person generation (ComMDM), and fine-tuned motion control (DiffusionBlending). OmniControl [40] controls any joints at any time by combining spatial and temporal control together. EDGE [36] introduces editing for dance generation from music tasks with similar editing capabilities, including body part editing, motion in-betweening, and long sequence generation. GMD [16] employs the concept of editing spatial parts and temporal intervals to guide the position of the root joint (pelvic) in order to control the motion trajectory. Recently, COMODO [31] controls motion trajectory by integrating an RL-based control framework with the inpainting method. However, all current approaches utilize a diffusion process directly on motion space, which is slow and impractical for real-time applications.

3. Method

Our goal is to design a text-to-motion synthesis paradigm that significantly improves synthesis quality, accelerates generation speed, and seamlessly preserves editability. Towards this goal, our paradigm, as depicted in Figure 3, consists of two modules: motion tokenizer (Section 3.1) and conditional masked motion transformer (Section 3.2). Motion tokenizer learns to transform 3D human motion into a sequence of discrete motion tokens without losing rich motion semantic information. Conditional masked motion

transformer is trained to predict randomly masked motion tokens conditioned on the pre-computed text tokens. During inference, masked motion transformer allows parallel decoding of multiple motion tokens simultaneously, while considering the context from both preceding and succeeding tokens.

3.1. Motion Tokenizer

The objective of the first stage is to learn a discrete latent space by quantizing the embedding from encoder outputs $\mathbf{z}$ into the entries or codes of a learned codebook via vector quantization, as shown in Figure 3(a). The objective of vector-quantization is defined as

$$L_{VQ} = \| \operatorname{sg}(\mathbf{z}) - \mathbf{e} \|_2^2 + \beta \| \mathbf{z} - \operatorname{sg}(\mathbf{e}) \|_2^2, \qquad (1)$$

where $\operatorname{sg}(\cdot)$ is the stop-gradient operator, β is the hyperparameter for commitment loss, and $\mathbf{e}$ is a codebook vector from codebook $\mathbf{E}$ ($\mathbf{e} \in \mathbf{E}$). The closet Euclidean distance of the embedding $\mathbf{z}$ and codebook vector index is calculated by $i = \operatorname{argmin}_j \| \mathbf{z} - \mathbf{E}_j \|_2^2$. To preserve the fine-grained motion representations, we adopt a large codebook with a size of 8192 to reduce the information loss during embedding quantization. Our experiments show that large codebook size with a suitable code embedding dimension can lead to improved motion generation quality. However, using large codebook can aggravate codebook collapse, where the majority of tokens are assigned to just a few codes, rendering the remaining codebook inactive. To boost codebook usage, we adopt the factorized code, which decouples code lookup and code embedding to stabilize large-size codebook learning [42]. In addition, moving averages during codebook update and resetting dead codebooks are also adopted, which are often employed for enhancing codebook utilization in VQ-VAE and VQ-GAN [7, 39]. These schemes together serve to transform 3D human motion into a sequence of discrete motion tokes in a robust and efficient manner.

3.2. Conditional Masked Motion Model

Text-conditioned Masked Transformer. During training, the motion tokens are first obtained by passing the output of the encoder through the vector quantizer. The motion token sequence, text embeddings, and special-purpose tokens serve as the inputs of a standard multi-layer transformer. Specifically, we obtain sentence embeddings from the pre-trained CLIP model [26] to capture the global relationships between the entire sentence and motion. The sentence embedding is then prepended to the motion tokens. Due to the nature of self-attention in transformers, all motion tokens are learned in relation to the sentence embedding. In addition, we obtain word embeddings from the same CLIP model [26] to capture the local relationships between each word and motion through cross-attention $= \operatorname{softmax}\left(Q_{motion} K_{\text{word}}^T / \sqrt{D} \right) V_{\text{word}}$ [47].

[MASK], [PAD], and [END] are learnable special-purpose tokens. During training, the [MASK] token is used to represent input corruption, and the model learns to predict the actual tokens in place of [MASK]. During inference, [MASK] tokens not only serve as placeholders for motion token generation but also prove useful in various tasks by placing the [MASK] tokens where editing is required, as detailed in Section 4. [PAD] token is used to fill up shorter motion sequences, allowing computation of batches with multiple sequences of varying lengths. Lastly, the [END] token is appended to the input tokens after the last token, providing the model with an explicit signal of the motion's endpoint.

Training Strategy and Loss. The motion token sequence Y is represented as $Y = [e_i]_{i=1}^L$, where L denotes the sequence length. We randomly mask out $r \times L$ tokens and replace them with learnable [MASK] tokens, where r is the masking ratio following a uniform distribution truncated between α and 1. Then, the original motion token sequence Y is updated with [MASK] tokens to form the corrupted motion sequence $Y_{\overline{\mathbf{M}}}$. This corrupted sequence along with text embedding W are fed into a text-conditioned masked transformer to reconstruct input motion token sequence with reconstruction probability or confidence equal to $p\left(y_i \mid Y_{\overline{\mathbf{M}}}, W\right)$. The objective is to minimize the negative log-likelihood of the predicted masked tokens conditioned on text:

$$\mathcal{L}_{\text{mask}} = - \mathbb{E}_{\mathbf{Y} \in \mathcal{D}} \left[\sum_{\forall i \in [1, L]} \log p\left(y_i \mid Y_{\overline{\mathbf{M}}}, W\right) \right]. \qquad (2)$$

Inference via Parallel Decoding. To decode the motion tokens during inference, we initiate the process by inputting all [MASK] tokens, representing an empty canvas then progressively predict more tokens per iteration. Next, iterative parallel decoding is performed, where in each iteration, the transformer masks out the subset of motion tokens that the model is least confident about and then predicts these masked tokens in parallel in the next iteration. The number of masked tokens n_M is determined by a masked scheduling function, *i.e.* a decaying function of the iteration t. The decaying function is chosen because at early iterations, there is high uncertainty in the predictions and thus the model begins with a large masking ratio and only keeps a small number of tokens with high prediction confidence. As the generation process proceeds, the masking ratio decreases due to the increase of context information from previous iterations. We experiment both linear $n_M = L((T - t)/T)$ [8] and cosine function $n_M = L \cos(\frac{1}{2}\pi t/T)$ [3], where cosine function yields better performance. To obtain the final n_M, the length of the generated motion sequence L should also be available. To address this issue, we adopt a pretrained predictor that estimates the motion sequence length based on the input text [12].
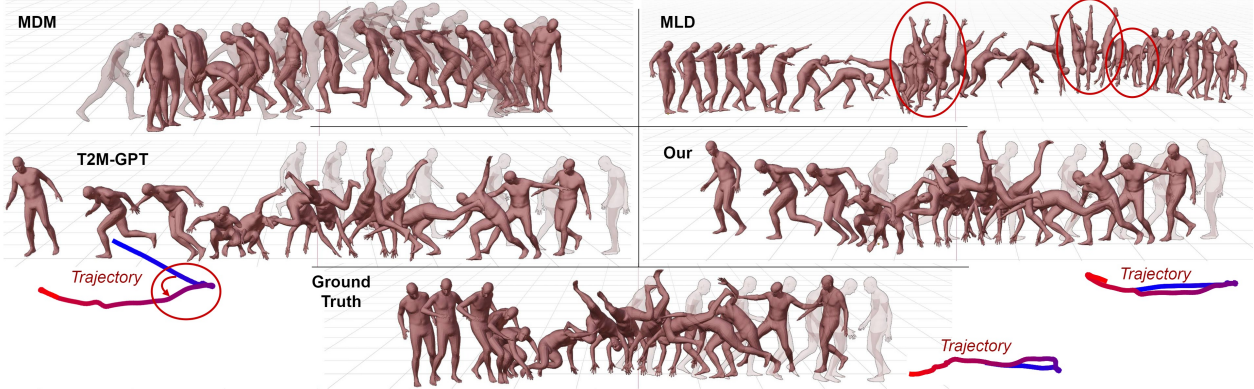
Figure 4. Motion Editing. (Left) Motion in-betweening. (Middle) Long Sequence Generation. (Right) Upper Body Editing. "M" refers to mask token. "T" is text conditioned tokens and "L" denotes lower body part conditioned tokens.

4. Motion Editing

As illustrated in Figure 4, the advantage of our masked motion modeling lies in its innate ability to edit motion.

Motion In-betweening. Thanks to its mask bidirectional decoding nature, we can easily place the [MASK] tokens wherever editing is needed regardless of past and future context. As an important editing task, the motion in-between involves interpolating or filling the gaps between keyframes or major motion points to create a smooth, continuous 3D animation. Since the model has already learned all possible random temporal masking combinations during training, motion in-between can be accomplished without any additional training.

Long Sequence Generation. Due to the limited length of motion data in the available HumanML3D [12] and KIT [24] datasets, where no sample exceeds a duration of 10 seconds, generating arbitrarily long motions poses a challenge. To address this, we use the trained masked motion model as a prior for long motion sequence synthesis without additional training. Particularly, given a story that consists of multiple text prompts, our model first generates the motion token sequence for each prompt. Then, we generate transition motion tokens conditioned on the end of the previous motion sequence and the start of the next motion sequence. Diffusion methods such as DoubleTakes in PriorMDM [30] require multiple steps (up to 1,000 steps) to generate transition and average the spatial differences between the previous and next motion before generating the transition motion. Our approach only needs a single step to generate realistic and natural transition motions.

Upper Body Editing. To enable body part editing, we pretrain the upper and lower body part tokenizers separately, each with its own encoders and decoders. The embedding size of each motion token is half of the regular full-body embedding size. In the second stage, the upper and lower tokens are concatenated back to form full body embeddings. Therefore, the embedding size and the conditional masked transformer remain unchanged. Ideally, we can train the transformer by predicting the masked upper body tokens, conditioned on the text for upper body motion along with the lower body tokens. However, the generated motions are inconsistent with the text. To address this prob-

lem, we introduce random [MASK] tokens into lower body part motion sequence via light corruptions so that the transformer can better learn the spatial and temporal dependency of the whole body motions. Thus, the training loss can be written as:

$$\mathcal{L}_{\text{up}} = - \mathop{\mathbb{E}}_{\mathbf{Y} \in \mathcal{D}} \left[\sum_{\forall i \in [1, L]} \log p \left(y_i^{up} \mid Y_{\overline{\mathbf{M}}}^{up}, Y_{\overline{\mathbf{M}}}^{down}, W \right) \right] \quad (3)$$

where $Y_{\overline{\mathbf{M}}}^{up}$ denotes the upper tokens with mask and $Y_{\overline{\mathbf{M}}}^{down}$ is the lower tokens with mask. It is important to note that the [MASK] tokens of the lower part remain unchanged throughout all iterations.

5. Experiments

In this section, we present comparisons to evaluate our models on both quality and time efficiency. In Section 5.1, following the standard evaluation from [12] across multiple datasets, we observe that our method consistently outperforms the current state-of-the-art methods. Moreover, in Section 5.2, evaluating with the time efficiency metric from MLD [4], our method exhibits shorter inference times, both on average and with respect to motion lengths.

Datasets. We conduct experiments on two standard datasets for text-to-motion generation: HumanML3D [12] and KIT Motion-Language (KIT-ML) [24] and follow the evaluation protocol proposed in [12]. **KIT Motion-Language (KIT-ML)** [24] contains 3911 motion sequences and 6,278 text descriptions, with an average of 9.5 words per annotation. One to four textual descriptions are provided for each motion clip, with an average description length of around 8 sentences. Motion sequences are selected from KIT [24] and CMU [41] datasets but have been downsampled to a rate of 12.5 frames per second (FPS). The dataset is split into training, validation, and test sets, with respective proportions of 80%, 5%, and 15%. **HumanML3D** [12] is currently the largest 3D human text-motion dataset, covers a diverse array of everyday human actions, including activities like exercising and dancing. The dataset comprises 14,616 motion sequences and 44,970 text descriptions. The entirety of the textual descriptions consists of 5,371 unique words. The motion sequences are originally from AMASS [20] and HumanAct12 [10]. Each sequence is adjusted to 20 frames per second (FPS) and trimmed to a 10-second duration, resulting in durations ranging from 2 to 10 seconds. Each motion clip is paired with at least three corresponding descriptions, with the average description length being approximately 12 words. Similar to KIT, the dataset is split into training, validation, and test sets, with respective proportions of 80%, 5%, and 15%.

Evaluation Metrics. The embeddings of textual descriptions and motion are encoded by pre-trained models from [12] for evaluation metrics as proposed by [12] with

Figure 5. Qualitative comparison of state-of-the-art methods with textual description: **"a person walks forward then turns completely around and does a cartwheel."** MDM is one of the most representative motion-space diffusion models. MLD is the first and SOTA latent-space motion diffusion model. T2M-GPT is the first and SOTA autoregressive motion model. Top-left: MDM [35] does not execute cartwheel motion. Top-right: MLD [4] generates unrealistic motion and lacks a complete cartwheel motion. Middle-left: the trajectory of T2M-GPT [43] is not **"completely around"**. Middle-right: our method generates realistic motion and trajectory compared to the ground truth on the bottom. Trajectories start from blue and end in red.
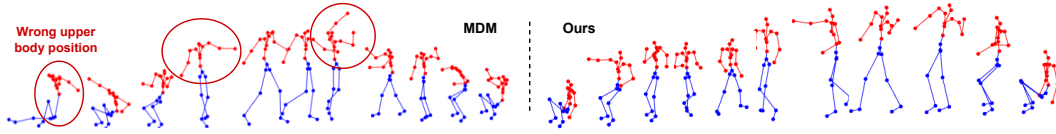
Figure 6. Qualitative comparison of **upper body editing**, generating upper body part based on the text "a man throws a ball" conditioned on lower body part of "a man rises from the ground, walks in a circle and sits back down on the ground."
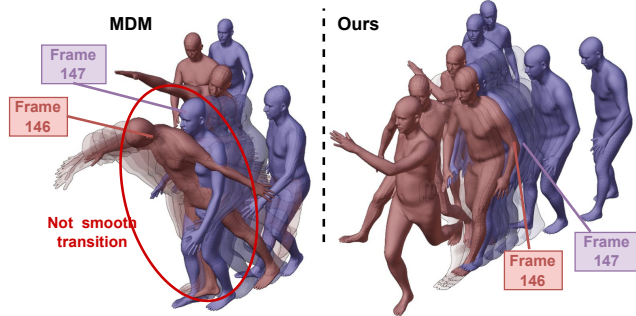
Figure 7. Qualitative comparison of **motion in-betweening**, generating 50% motion in the middle (frame 50-146) based on the text "a man throws a ball" conditioned on first 25% and last 25% of motion of "a person walks backward, turns around and walks backward the other way.". Compared with MDM, MMM achieves smoother and more natural transitions between the conditioned and generated motions (at frames 146 and 147)

five metrics. **R-precision** and **Multimodal Distance (MM-Dist)** measure how well the generated motions align with the input prompts. Top-1, Top-2, and Top-3 accuracy are reported for R-Precision. **Frechet Inception Distance (FID)** measures the distribution distance between the generated and ground truth motion features. **Diversity** is calculated by averaging Euclidean distances of random samples from 300 pairs of motion, and **MultiModality (MModality)** represents the average variance for a single text prompt by computing Euclidean distances of 10 generated pairs of motions.

5.1. Comparison to State-of-the-art Approaches

We evaluate our methods with state-of-the-art approaches [4, 9, 11, 12, 18, 22, 35, 38, 43, 44, 47] on HumanML3D [12] and KIT-ML [24]. We maintain the same architecture and hyperparameters for the evaluation on both datasets.

Quantitative comparisons. Following [12], we report the average over 20 repeated generations with 95% confidence interval. Table 1 and Table 2 present evaluations on HumanML3D [12] and KIT-ML [24] dataset respectively, in comparison to state-of-the-art (SOTA) approaches. Our method consistently performs best in terms of FID and Multimodal Distance. For the R-Precision and Diversity metric, our method still shows competitive results when compared to SOTA methods. On HumanML3D [12], as shown in Table 1, our method outperforms all SOTA methods in most of the metrics. Specifically, our method excels in Top-1, Top-2, and Top-3 R-Precision, FID, Multimodal Distance, and Diversity metrics. It is worth noting that the HumanML3D [12] dataset is significantly larger than the KIT-ML [24] dataset, which suggests that our method tends to perform better with more data.

Qualitative comparison. Figure 5 shows visual result of the generated motion by textual description, "a person walks forward then turns completely around and does a cartwheel.", compared to state-of-the-art models. MDM [35] fails to generate a cartwheel motion. MLD [4] flips in an unrealistic motion and lacks a complete cartwheel mo-

Table 1. **Comparison of text-conditional motion synthesis on HumanML3D [12] test set.** For each metric, we repeat the evaluation 20 times and report the average with 95% confidence interval. The right arrow ($\rightarrow$) indicates that the closer the result is to real motion, the better. Red and Blue indicate the best and the second best result. § reports results using ground-truth motion length.

Methods	R-Precision $\uparrow$			FID $\downarrow$	MM-Dist $\downarrow$	Diversity $\rightarrow$	MModality $\uparrow$
	Top-1 $\uparrow$	Top-2 $\uparrow$	Top-3 $\uparrow$				
Real	$0.511^{\pm.003}$	$0.703^{\pm.003}$	$0.797^{\pm.002}$	$0.002^{\pm.000}$	$2.974^{\pm.008}$	$9.503^{\pm.065}$	-
VQ-VAE	$0.503^{\pm.003}$	$0.698^{\pm.003}$	$0.793^{\pm.002}$	$0.075^{\pm.001}$	$3.027^{\pm.008}$	$9.697^{\pm.076}$	-
Hier [9]	$0.301^{\pm.002}$	$0.425^{\pm.002}$	$0.552^{\pm.004}$	$6.523^{\pm.024}$	$5.012^{\pm.018}$	$8.332^{\pm.042}$	-
TEMOS§ [22]	$0.424^{\pm.002}$	$0.612^{\pm.002}$	$0.722^{\pm.002}$	$3.734^{\pm.028}$	$3.703^{\pm.008}$	$8.973^{\pm.071}$	$0.368^{\pm.018}$
TM2T [11]	$0.424^{\pm.003}$	$0.618^{\pm.003}$	$0.729^{\pm.002}$	$1.501^{\pm.017}$	$3.467^{\pm.011}$	$8.589^{\pm.076}$	$2.424^{\pm.093}$
T2M [12]	$0.455^{\pm.003}$	$0.636^{\pm.003}$	$0.736^{\pm.002}$	$1.087^{\pm.021}$	$3.347^{\pm.008}$	$9.175^{\pm.083}$	$2.219^{\pm.074}$
MDM§ [35]	$0.320^{\pm.005}$	$0.498^{\pm.004}$	$0.611^{\pm.007}$	$0.544^{\pm.044}$	$5.566^{\pm.027}$	$9.559^{\pm.086}$	$2.799^{\pm.072}$
MotionDiffuse§ [44]	$0.491^{\pm.001}$	$0.681^{\pm.001}$	$0.782^{\pm.001}$	$0.630^{\pm.001}$	$3.113^{\pm.001}$	$9.410^{\pm.049}$	$1.553^{\pm.042}$
MLD§ [4]	$0.481^{\pm.003}$	$0.673^{\pm.003}$	$0.772^{\pm.002}$	$0.473^{\pm.013}$	$3.196^{\pm.010}$	$9.724^{\pm.082}$	$2.413^{\pm.079}$
Fg-T2M§ [38]	$0.492^{\pm.002}$	$0.683^{\pm.003}$	$0.783^{\pm.002}$	$0.243^{\pm.019}$	$3.109^{\pm.007}$	$9.278^{\pm.072}$	$1.614^{\pm.049}$
M2DM§ [18]	$0.497^{\pm.003}$	$0.682^{\pm.002}$	$0.763^{\pm.003}$	$0.352^{\pm.005}$	$3.134^{\pm.010}$	$9.926^{\pm.073}$	$3.587^{\pm.072}$
T2M-GPT [43]	$0.491^{\pm.003}$	$0.680^{\pm.003}$	$0.775^{\pm.002}$	$0.116^{\pm.004}$	$3.118^{\pm.011}$	$9.761^{\pm.081}$	$1.856^{\pm.011}$
AttT2M [47]	$0.499^{\pm.003}$	$0.690^{\pm.002}$	$0.786^{\pm.002}$	$0.112^{\pm.006}$	$3.038^{\pm.007}$	$9.700^{\pm.090}$	$2.452^{\pm.051}$
MMM§ (ours)	$0.515^{\pm.002}$	$0.708^{\pm.002}$	$0.804^{\pm.002}$	$0.089^{\pm.005}$	$2.926^{\pm.007}$	$9.577^{\pm.050}$	$1.226^{\pm.035}$
MMM (ours)	$0.504^{\pm.003}$	$0.696^{\pm.003}$	$0.794^{\pm.002}$	$0.080^{\pm.003}$	$2.998^{\pm.007}$	$9.411^{\pm.058}$	$1.164^{\pm.041}$

Table 2. **Comparison of text-conditional motion synthesis on KIT-ML [24] test set.** For each metric, we repeat the evaluation 20 times and report the average with 95% confidence interval. The right arrow ($\rightarrow$) indicates that the closer the result is to real motion, the better. Red and Blue indicate the best and the second best result. § reports results using ground-truth motion length.

Methods	R-Precision $\uparrow$			FID $\downarrow$	MM-Dist $\downarrow$	Diversity $\rightarrow$	MModality $\uparrow$
	Top-1 $\uparrow$	Top-2 $\uparrow$	Top-3 $\uparrow$				
Real	$0.424^{\pm.005}$	$0.649^{\pm.006}$	$0.779^{\pm.006}$	$0.031^{\pm.004}$	$2.788^{\pm.012}$	$11.080^{\pm.097}$	-
VQ-VAE	$0.392^{\pm.006}$	$0.606^{\pm.006}$	$0.736^{\pm.006}$	$0.641^{\pm.014}$	$3.047^{\pm.012}$	$11.075^{\pm.113}$	-
Hier [9]	$0.255^{\pm.006}$	$0.432^{\pm.007}$	$0.531^{\pm.007}$	$5.203^{\pm.107}$	$4.986^{\pm.027}$	$9.563^{\pm.072}$	-
TEMOS§ [22]	$0.353^{\pm.006}$	$0.561^{\pm.007}$	$0.687^{\pm.005}$	$3.717^{\pm.051}$	$3.417^{\pm.019}$	$10.84^{\pm.100}$	$0.532^{\pm.034}$
TM2T [11]	$0.280^{\pm.005}$	$0.463^{\pm.006}$	$0.587^{\pm.005}$	$3.599^{\pm.153}$	$4.591^{\pm.026}$	$9.473^{\pm.117}$	$3.292^{\pm.081}$
T2M [12]	$0.361^{\pm.006}$	$0.559^{\pm.007}$	$0.681^{\pm.007}$	$3.022^{\pm.107}$	$3.488^{\pm.028}$	$10.72^{\pm.145}$	$2.052^{\pm.107}$
MDM§ [35]	$0.164^{\pm.004}$	$0.291^{\pm.004}$	$0.396^{\pm.004}$	$0.497^{\pm.021}$	$9.191^{\pm.022}$	$10.85^{\pm.109}$	$1.907^{\pm.214}$
MotionDiffuse§ [44]	$0.417^{\pm.004}$	$0.621^{\pm.004}$	$0.739^{\pm.004}$	$1.954^{\pm.064}$	$2.958^{\pm.005}$	$11.10^{\pm.143}$	$0.730^{\pm.013}$
MLD§ [4]	$0.390^{\pm.008}$	$0.609^{\pm.008}$	$0.734^{\pm.007}$	$0.404^{\pm.027}$	$3.204^{\pm.027}$	$10.80^{\pm.117}$	$2.192^{\pm.071}$
Fg-T2M§ [38]	$0.418^{\pm.005}$	$0.626^{\pm.004}$	$0.745^{\pm.004}$	$0.571^{\pm.047}$	$3.114^{\pm.015}$	$10.93^{\pm.083}$	$1.019^{\pm.029}$
M2DM§ [18]	$0.416^{\pm.004}$	$0.628^{\pm.004}$	$0.743^{\pm.004}$	$0.515^{\pm.029}$	$3.015^{\pm.017}$	$11.417^{\pm.97}$	$3.325^{\pm.37}$
T2M-GPT [43]	$0.402^{\pm.006}$	$0.619^{\pm.005}$	$0.737^{\pm.006}$	$0.717^{\pm.041}$	$3.053^{\pm.026}$	$10.86^{\pm.094}$	$1.912^{\pm.036}$
AttT2M [47]	$0.413^{\pm.006}$	$0.632^{\pm.006}$	$0.751^{\pm.006}$	$0.870^{\pm.039}$	$3.039^{\pm.021}$	$10.96^{\pm.123}$	$2.281^{\pm.047}$
MMM§ (ours)	$0.404^{\pm.005}$	$0.621^{\pm.005}$	$0.744^{\pm.004}$	$0.316^{\pm.028}$	$2.977^{\pm.019}$	$10.910^{\pm.101}$	$1.232^{\pm.039}$
MMM (ours)	$0.381^{\pm.005}$	$0.590^{\pm.006}$	$0.718^{\pm.005}$	$0.429^{\pm.019}$	$3.146^{\pm.019}$	$10.633^{\pm.097}$	$1.105^{\pm.026}$

Table 3. Evaluation of two editing tasks, motion in-betweening and upper body editing, both with text and without text conditioning, in comparison to MDM [35], using evaluations from [12].

	Methods	R-Precision Top-1 $\uparrow$	FID $\downarrow$	MM-Dist $\downarrow$	Diversity $\rightarrow$
In-betweening (w/ text)	MDM	0.389	2.371	3.859	8.077
	Ours	**0.5226**	**0.0712**	**2.9097**	**9.5794**
In-betweening (w/o text)	MDM	0.284	3.417	4.941	7.472
	Ours	**0.4239**	**0.114**	**3.6310**	**9.3817**
Upper Body Editing (w/ text)	MDM	0.298	4.827	4.598	7.010
	Ours	**0.500**	**0.1026**	**2.9720**	**9.2540**
Upper Body Editing (w/o text)	MDM	0.258	7.436	5.075	6.647
	Ours	**0.4834**	**0.1338**	**3.1876**	**9.0092**

tion. Meanwhile, the trajectory of T2M-GPT [43] does not align with the "completely around" description in the text prompt. Our approach shows the realistic motion and trajectory compared to the ground truth. More visual results is shown in the **supplementary material**.

5.2. Inference Speed and Editability

Inference Speed. Figure 2 in Section 1 compares the inference speeds (AITS) of MDM [35], MotionDiffuse [44], MLD [4], T2M-GPT [43], AttT2M [47] and MMM (our method). All tests are performed on a single NVIDIA RTX A5000. It can be shown that MMM is at least two times faster than autoregressive motion models (T2M-GPT and AttT2M) and latent-space diffusion model (MLD), while being two orders of magnitude faster than motion-space diffusion models (MDM and MotionDiffuse), which directly operate on raw motion sequences. Since the maximum motion duration of HumanML3D dataset is 9.6 seconds, AITS in Figure 2 actually measures the speed of generating short motion. Such high inference speed offers a significant advantage for long-range generation. As discussed in Section 4, to generate long-range motion, we combine multiple

Table 4. Ablation results on the masking ratio during training.

Masking Ratio	R-Precision Top-1 ↑	FID ↓	MM-Dist ↓	Diversity →	MModality ↑
Uniform 0-1	0.513	0.097	2.923	9.640	1.097
Uniform .3-1	**0.520**	**0.089**	**2.901**	9.639	1.149
Uniform .5-1	0.515	**0.089**	2.926	**9.577**	**1.226**
Uniform .7-1	0.505	0.098	2.975	9.523	1.245

Table 5. Ablation results on speed and quality influenced by the mask scheduling during inference.

# of iterations	AITS (seconds) ↓	R-Precision Top-1 ↑	FID ↓	MM-Dist ↓	Diversity →	MModality ↑
5	**0.043**	0.505	0.169	3.003	9.370	**1.414**
10	0.081	0.515	**0.089**	2.926	9.577	1.226
15	0.118	0.516	0.091	2.919	**9.576**	1.134
20	0.149	0.518	0.096	2.912	9.590	1.058
25	0.176	0.517	0.102	2.911	9.682	1.008
30	0.205	0.515	0.100	2.908	9.698	0.937
35	0.213	**0.519**	0.105	**2.896**	9.701	0.969
49 (Linear)	0.345	0.517	0.109	2.911	9.716	0.952

Table 6. Ablation results on quality influenced by the number of codes and the code dimension.

# of code x code dimension	R-Precision Top-1 ↑	FID ↓	MM-Dist ↓	Diversity →
512 x 512	0.499	0.108	3.077	9.683
1024 x 256	0.496	0.0850	3.001	**9.641**
2048 x 128	0.504	0.093	3.033	9.795
4096 x 64	**0.505**	0.080	3.035	9.697
8192 x 32	0.503	**0.075**	**3.027**	9.697

short motions with transition tokens, where the transition generation can be completed in a single iteration. In particular, our method generates a 10.873-minute sequence in only 1.658 seconds.

Editability. Table 3 demonstrates that our bidirectional mask modeling outperforms MDM [35] in two editing tasks: motion in-betweening and upper body editing. We conducted experiments with and without text conditioning, evaluating on the test set of HumanML3D [12]. Motion in-betweening is evaluated by generating 50% of the sequences conditioned by first and last 25% of the sequences. Qualitative results are shown in Figure 1. Moreover, we also visualize the upper body editing in Figure 6 and motion in-betweening in Figure 7 in compared to MDM [35]. MDM [35] generates unrealistic upper body distinct from the lower part and exhibits discontinuous frame transitions. Our approach, on the other hand, demonstrates realistic motion and smooth transitions for editing tasks. We show more visual results in the **supplementary material**.

6. Ablation Study

The key success of our method lies in mask modeling. To understand how mask scheduling during training and inference impacts performance, we conduct ablation experiments using the same evaluation on HumanML3D [12].

Masking Ratio during Training. During training, we apply a random masking ratio drawn from an uniform distribution bounded between α and 1. The larger α indicates that more aggressive masking is applied during training. As a result, the model has to predict a large number of masked tokens more based on the text prompt due to the reduced context information from unmasked motion tokens. It is shown in Table 4 that too aggressive ($\alpha = 0.7$) or too mild ($\alpha = 0$) masking strategy is not beneficial for generating high-fidelity motions with sufficient diversity.

Iteration Number. During inference, the number of iterations directly affects the speed and the quality of the generation. The number of iterations is the maximum number of iterations for the longest motion sequence in the dataset (196 frames). As shown in Table 5, increasing the number of iterations leads to higher generation latency with slightly improved R-precision and MM-dist. However, using a small number of iterations like 10, the model already achieves the lowest FID score along with high R-precision and low MM-dist. The lowest FID score means the best overall visual quality of the generated motion, ensuring that its realism and naturalness is very close to the genuine and ground-truth human movements. High R-precision and low MM-dist indicate the precise alignment between the generated motion and the text description.

Codebook Size. By adopting codebook factorization, we can learn a large-size codebook for high-resolution motion embedding quantization to preserve fine-grained motion representations, which directly affect motion generation quality. In particular, codebook factorization decoupling code lookup and code embedding, where codebook uses low-dimensional latent space for code lookup and the matched code is projected back to high-dimensional embedding space. Table 6 shows that motion reconstruction quality is improved by increasing the number of code entries in codebook from 512 to 8192, while reducing the dimension of the codebook latent space from 512-d to 32-d.

7. Conclusion

In this work, we propose the generative masked motion model (MMM) to synthesize human motion based on textual descriptions. MMM consists of two key components: (1) a motion tokenizer converting 3D human motion into discrete latent space tokens, and (2) a conditional masked motion transformer predicting masked motion tokens based on text tokens. MMM enables parallel and iteratively-refined decoding for high-fidelity and fast motion generation. MMM has inherent motion editability. Extensive experiments demonstrate that MMM outperforms state-of-the-art methods both qualitatively and quantitatively. MMM is at least two times faster than autoregressive motion models and two orders of magnitude faster than diffusion models on raw motion sequences.

References

[1] Chaitanya Ahuja and Louis-Philippe Morency. Language2pose: Natural language grounded pose forecasting. In *2019 International Conference on 3D Vision (3DV)*, pages 719–728, 2019. 2, 3

[2] 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, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff 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. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. 2, 3

[3] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11305–11315, 2022. 3, 4

[4] Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, Jingyi Yu, and Gang Yu. Executing your commands via motion diffusion in latent space. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18000–18010, 2022. 2, 3, 5, 6, 7, 1, 4

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. 3

[6] Ming Ding, Wendi Zheng, Wenyi Hong, and Jie Tang. Cogview2: Faster and better text-to-image generation via hierarchical transformers. *ArXiv*, abs/2204.14217, 2022. 3

[7] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12868–12878, 2020. 3, 4

[8] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Conference on Empirical Methods in Natural Language Processing*, 2019. 3, 4

[9] Anindita Ghosh, Noshaba Cheema, Cennet Oguz, Christian Theobalt, and P. Slusallek. Synthesis of compositional animations from textual descriptions. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1376–1386, 2021. 6, 7

[10] Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong, and Li Cheng. Action2motion: Conditioned generation of 3d human motions. *Proceedings of the 28th ACM International Conference on Multimedia*, 2020. 5

[11] Chuan Guo, Xinxin Xuo, Sen Wang, and Li Cheng. Tm2t: Stochastic and tokenized modeling for the reciprocal generation of 3d human motions and texts. *ArXiv*, abs/2207.01696, 2022. 6, 7

[12] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5142–5151, 2022. 2, 3, 4, 5, 6, 7, 8, 1

[13] Jonathan Ho, Ajay Jain, and P. Abbeel. Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239, 2020. 3

[14] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey A. Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *ArXiv*, abs/2210.02303, 2022. 3

[15] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. Motiongpt: Human motion as a foreign language. *ArXiv*, abs/2306.14795, 2023. 2, 3

[16] Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. Gmd: Controllable human motion synthesis via guided diffusion models. *ArXiv*, abs/2305.12577, 2023. 3

[17] Jihoon Kim, Jiseob Kim, and Sungjoon Choi. Flame: Freeform language-based motion synthesis & editing. In *AAAI Conference on Artificial Intelligence*, 2022. 2, 3

[18] Hanyang Kong, Kehong Gong, Dongze Lian, Michael Bi Mi, and Xinchao Wang. Priority-centric human motion generation in discrete latent space. *ArXiv*, abs/2308.14480, 2023. 6, 7

[19] Yunhong Lou, Linchao Zhu, Yaxiong Wang, Xiaohan Wang, and Yezhou Yang. Diversemotion: Towards diverse human motion generation via discrete diffusion. *ArXiv*, abs/2309.01372, 2023. 2

[20] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. Amass: Archive of motion capture as surface shapes. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5441–5450, 2019. 5

[21] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, 2021. 3

[22] Mathis Petrovich, Michael J. Black, and Gül Varol. Temos: Generating diverse human motions from textual descriptions. *ArXiv*, abs/2204.14109, 2022. 2, 3, 6, 7

[23] Mathis Petrovich, Michael J. Black, and Gül Varol. Tmr: Text-to-motion retrieval using contrastive 3d human motion synthesis. *ArXiv*, abs/2305.00976, 2023. 2, 3

[24] Matthias Plappert, Christian Mandery, and Tamim Asfour. The KIT motion-language dataset. *Big Data*, 4(4):236–252, 2016. 2, 5, 6, 7

[25] Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. Glancing transformer for non-autoregressive neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, 2020. 3

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 4

[27] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *ArXiv*, abs/2102.12092, 2021. 3

[28] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2021. 3

[29] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, Seyedeh Sara Mahdavi, Raphael Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *ArXiv*, abs/2205.11487, 2022. 3

[30] Yonatan Shafir, Guy Tevet, Roy Kapon, and Amit H. Bermano. Human motion diffusion as a generative prior. *ArXiv*, abs/2303.01418, 2023. 3, 5

[31] Yi Shi, Jingbo Wang, Xuekun Jiang, and Bo Dai. Controllable motion diffusion model. *ArXiv*, abs/2306.00416, 2023. 3

[32] Uriel Singer, Adam Polyak, Thomas Hayes, Xiaoyue Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. *ArXiv*, abs/2209.14792, 2022. 3

[33] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *ArXiv*, abs/2010.02502, 2020. 3

[34] Guy Tevet, Brian Gordon, Amir Hertz, Amit H. Bermano, and Daniel Cohen-Or. Motionclip: Exposing human motion generation to clip space. In *European Conference on Computer Vision*, 2022. 2, 3

[35] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. Human motion diffusion model. *ArXiv*, abs/2209.14916, 2022. 2, 3, 6, 7, 8, 1, 4

[36] Jo-Han Tseng, Rodrigo Castellon, and C. Karen Liu. Edge: Editable dance generation from music. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 448–458, 2022. 3

[37] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *ArXiv*, abs/1711.00937, 2017. 2

[38] Yin Wang, Zhiying Leng, Frederick W. B. Li, Shun-Cheng Wu, and Xiaohui Liang. Fg-t2m: Fine-grained text-driven human motion generation via diffusion model. *ArXiv*, abs/2309.06284, 2023. 2, 3, 6, 7

[39] Will Williams, Sam Ringer, Tom Ash, John Hughes, David Macleod, and Jamie Dougherty. Hierarchical quantized autoencoders. *ArXiv*, abs/2002.08111, 2020. 3, 4

[40] Yiming Xie, Varun Jampani, Lei Zhong, Deqing Sun, and Huaizu Jiang. Omnicontrol: Control any joint at any time for human motion generation. 2023. 3

[41] Sheng Yan, Yang Liu, Haoqiang Wang, Xin Du, Mengyuan Liu, and Hong Liu. Cross-modal retrieval for motion and text via doptriple loss. 2023. 2, 3, 5

[42] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *ArXiv*, abs/2110.04627, 2021. 3, 4

[43] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Shaoli Huang, Yong Zhang, Hongwei Zhao, Hongtao Lu, and Xiaodong Shen. Generating human motion from textual descriptions with discrete representations. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14730–14740, 2023. 2, 3, 6, 7, 1, 4

[44] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *ArXiv*, abs/2208.15001, 2022. 2, 3, 6, 7, 1, 4

[45] Zhu Zhang, Jianxin Ma, Chang Zhou, Rui Men, Zhikang Li, Ming Ding, Jianbo Tang, Jingren Zhou, and Hongxia Yang. M6-ufc: Unifying multi-modal controls for conditional image synthesis via non-autoregressive generative transformers. 2021. 3

[46] Zhu Zhang, Jianxin Ma, Chang Zhou, Rui Men, Zhikang Li, Ming Ding, Jie Tang, Jingren Zhou, and Hongxia Yang. Ufc-bert: Unifying multi-modal controls for conditional image synthesis. In *Neural Information Processing Systems*, 2021. 3

[47] Chongyang Zhong, Lei Hu, Zihao Zhang, and Shihong Xia. Attt2m: Text-driven human motion generation with multi-perspective attention mechanism. *ArXiv*, abs/2309.00796, 2023. 2, 3, 4, 6, 7, 1

MMM: Generative Masked Motion Model

Supplementary Material

A. Overview

The supplementary material is organized into the following sections:
- Section B: Inference speed, quality, and editability
- Section C: Codebook reset
- Section D: Influence of word embedding via cross attention
- Section E: Confidence-based masking
- Section F: Inference speed relative to motion length
- Section G: Impact of mask scheduling functions
- Section H: Impact of token sampling strategies
- Section I: Qualitative results including two new motion editing tasks: motion extrapolation/outpainting, motion completion
- Section J: Limitations

For more visualization, please visit our anonymous website `https://anonymous-ai-agent.github.io/MMMM`

B. Inference speed, quality, and editability

To summarize the advantages of our method in three aspects—speed, quality, and editability—we compare AITS, R-Precision, FID, and Editability for MDM [35], MotionDiffuse [44], MLD [4], T2M-GPT [43], AttT2M [47] in Table 7. Only MDM [35], a diffusion model applied directly to motion space, and our method allows for editability. We denote this feature with a checkmark symbol, '✓'. It's worth noting that MotionDiffuse [44] is also capable of editability, although no specific application is provided. In the table, we represent this with a hyphen symbol, '—'. While applying the diffusion process directly to motion space provides editable capabilities for various applications, the inference time of diffusion models is not practical for real-time applications. The inference time is nearly threefold the duration of the generated motion sequence. Specifically, it takes 28.112 seconds on average to generate 196 frames, which is equivalent to a 9.8-second motion sequence. On the other hand, MLD [4] performs a diffusion process on a single motion latent space to speed up the generation time. However, the compression of its encoder not only results in the loss of fine-grained synthesis detail but also restricts its ability to edit the motion. T2M-GPT [43] and AttT2M [47] compress motion into multiple temporal embeddings, leading to faster generation time. However, due to their autoregressive approach, they lack the ability to see future tokens, which also results in a loss of editability, as indicated by the '✗'. Table 7 shows that our approach exhibits the highest quality and preserves editable capabilities while employing only 0.081 seconds on average to generate motion.

Table 7. Comparison of the inference speed and quality of generation on text-to-motion along with the editable capability of each model. '✓' means editable while '✗' is not and '—' refers to has-capability but no application provided. We calculate the Average Inference Time per Sentence (AITS) on the test set of HumanML3D [12] without model or data loading parts. All tests are performed on a single NVIDIA RTX A5000.

Methods	AITS (seconds) ↓	R-Precision Top-1 ↑	FID ↓	Edit
MDM	28.112	0.320	0.544	✓
MotionDiffuse	10.071	0.491	0.630	—
MLD	0.220	0.481	0.473	✗
T2M-GPT	0.350	0.491	0.116	✗
AttT2M	0.528	0.499	0.112	✗
MMM (our)	**0.081**	**0.515**	**0.089**	✓

C. Codebook Reset

During stage 1, the motion tokenizer learns discrete tokens as discussed in 3.1. Besides codebook factorization, we also adopt codebook reset to prevent codebook collapse, where the majority of tokens are allocated to only a few codes, while the rest of the codebook entries are inactive. As shown in our experiments, the codebook reset frequency directly affects codebook utilization and thus motion generation quality. For example, simply implementing codebook reset every training iteration impedes the codebook from learning motion tokens effectively, aggravating codebook collapse. As visually depicted

in Figure 8, metrics such as FID score, R-precision, and Multi-modal Distance are initially learned during the early stages of training. However, when codebook collapse occurs, all the metrics significantly worsen. In contrast, resetting the unused codebook less frequently means the codebook will not be fully utilized, causing less fine-grain detail that the codebook can capture, which shows worse quality in 60 to 80 codebook reset iterations in Table 8.

Moreover, the results in **Stage 1: Motion Tokenizer** and **Stage 2: Conditional Masked Transformer** may not align. This is because the Stage 1 objective is the reconstruction task without text condition, while the Stage 2 objective is to generate motion conditioned by text. Therefore, even though resetting every 40 iterations leads to the best performance in the first stage, resetting the codebook every 20 iterations works best in the text-to-motion task.

Table 8. Ablation results on codebook reset every different number of training iterations. Codebook reset iteration in stage 2 indicates the pretrained models from stage 1.

| **Stage1: Motion Tokenizer**							
Codebook Reset Every Number of Training Iteration	R-Precision ↑			FID ↑	MM-Dist ↓	Diversity →	Loss ↓
	Top-1	Top-2	Top-3				
1 Iteration	Diverge (as shown in Figure 8)						
20 Iterations	0.503	0.698	0.793	0.075	3.027	9.697	**0.05156**
40 Iterations	**0.507**	**0.698**	**0.793**	**0.059**	**3.013**	**9.629**	0.05196
60 Iterations	**0.507**	**0.698**	**0.793**	0.075	3.019	9.710	0.05185
80 Iterations	0.505	0.697	0.793	0.079	3.022	9.658	0.05163
Stage2: Conditional Masked Transformer							
Codebook Reset Every Number of Training Iterations (From 1st Stage)	R-Precision ↑			FID ↑	MM-Dist ↓	Diversity →	MModality ↑
	Top-1	Top-2	Top-3				
20 Iterations	**0.515**	**0.708**	**0.804**	**0.089**	**2.926**	**9.577**	**1.226**
40 Iterations	0.508	0.702	0.798	0.108	2.954	9.645	1.136
60 Iterations	0.518	0.710	0.805	0.0923	2.897	9.7163	1.157
80 Iterations	0.507	0.701	0.797	0.111	2.963	9.541	1.208
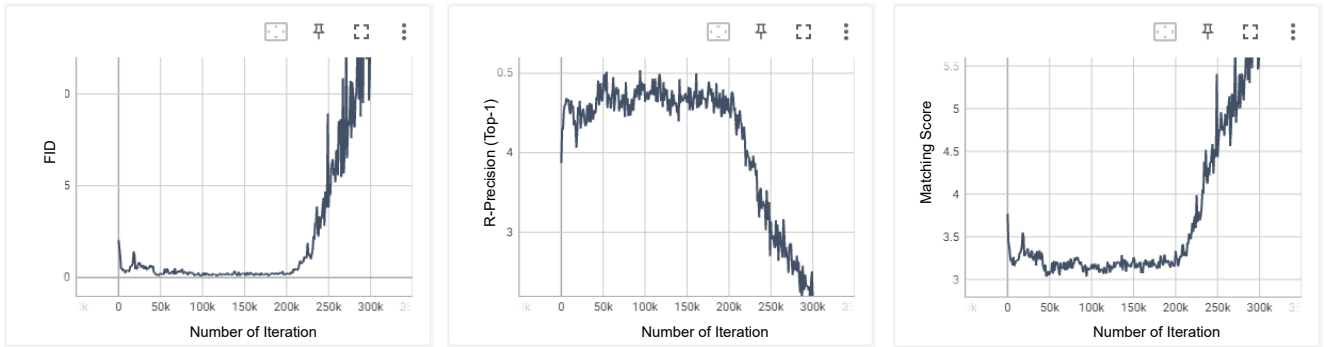
Figure 8. Codebook collapse effects in FID score, R-precision, and Multi-modal Distance metrics on the evaluation set during training using codebook reset in every training iteration.

D. Influence of Word Embedding via Cross Attention

As discussed in Section. 3.2, word embedding and sentence embedding help the model learn global and local text-to-motion relationship. We ablate the number of cross attention between word and motion embeddings to study the influence of word embedding. We replace self-attention with cross-attention at the starting layers while maintaining the same total number of 18 layers. The results presented in Table 9 indicate that a higher number of cross-attention layers leads to improved R-Precision but also worsens the FID score.

Table 9. Ablation results on the number of cross attention layers.

Number of Cross Attention Layer	R-Precision ↑			FID ↓	MM-Dist ↓	Diversity →	MModality ↑
	Top-1	Top-2	Top-3				
0 Layer	0.486	0.674	0.772	**0.082**	3.108	**9.528**	1.224
1 Layer	0.515	0.708	0.804	0.089	2.926	9.577	1.226
2 Layer	0.522	0.714	0.807	0.090	2.902	9.587	1.206
4 Layer	0.524	0.716	0.810	0.094	2.891	9.651	1.234
9 Layer	**0.527**	**0.721**	**0.816**	0.114	**2.872**	9.613	**1.325**

E. Confidence-based Masking

During the inference stage, MMM adopts confidence-based masking as shown in Figure 3. To understand the behavior of this masking strategy, we visualize the confidence levels of all sequence motion tokens in each iteration. As illustrated in Figure 9 and Figure 10, the x-axis represents the indices of temporal tokens ranging from 0 to 48 (where 49 tokens correspond to 196 frames, as 4 frames are compressed into 1 token), and the y-axis indicates the 10 iterations during the generation process. The tokens with the highest confidence are predicted and used as input conditions for the next iteration. In the initial iteration (x-axis = 0), Figure 9 suggests that the model predicts most tokens with very high confidence when it solely focuses on the text condition without being constrained by other prior tokens, as all tokens are masked. However, in the second iteration, conditioned by the highest confidence token from the first iteration, the model's confidence drops before gradually increasing in the following iterations. By the final iteration, all tokens are predicted, and no [MASK] tokens remain, as shown in Figure 10. Notably, we observe that the increasing confidence tends to be around the location of the previous high-confidence token and expands in the later iteration.
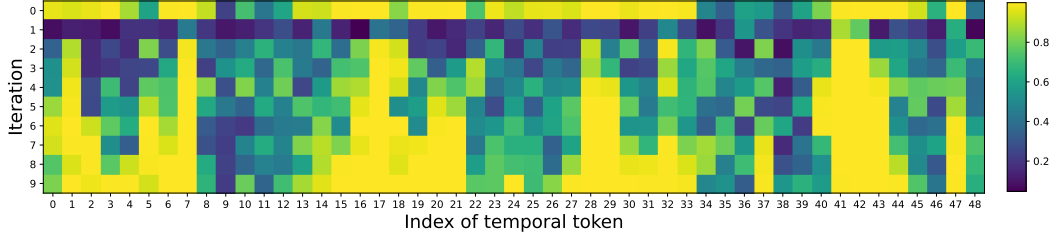

Figure 9. Visualization of the confidence in each iteration from high confidence (🟨) to low confidence(🟦)
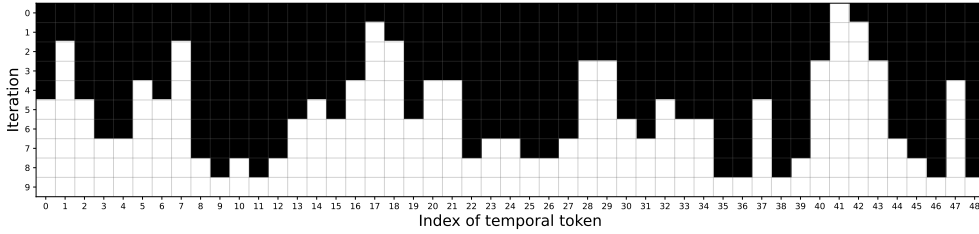

Figure 10. Visualization of mask tokens in each iteration. ■ indicates [MASK] tokens, and □ refers to unmasked tokens.

F. Inference Speed Relative to Motion Length

One notable advantage of our approach resides in its superior inference speed relative to the length of the generated motion. In particular, we use a dynamic masking schedule to calculate the number of [MASK] tokens m_t at iteration t w.r.t. maximum iteration T, the length of the motion tokens L, and the maximum number of motion tokens M that can be taken by the conditional transformer as the inputs, where M is determined by the maximum-length motion in the datasets. We compute

the number of mask n_M at t iteration by a mask scheduling function as follows:

$$n_M(t) = \left\lceil \gamma \left(\frac{t}{T_{dyn}} \right) \cdot L \right\rceil .$$ (4)

where the total iteration of each sample, i.e., $T_{dyn} = T \cdot \frac{L}{M}$ with $L \leq M$, is proportional to its sample length L. $\gamma()$ is a decaying function. The choice of the decaying function is discussed in Section G.

As shown in Figure 11, diffusion-based approaches, MDM [35], MotionDiffuse [44], and MLD [4], exhibit the same inference time regardless of the number of generated frames. In contrast, token-based approaches such as T2M-GPT [43] and AttT2M [47] show slower inference speeds than MLD [4] for long-length sequences but faster speeds for short sequences. Our approach, on the other hand, is not only faster than all state-of-the-art approaches for long sequences but even faster for short sequence generation. Specifically, the inference speed for a 40-frame sequence can be as fast as 0.018 seconds.

This behavior offers a significant advantage for long-range generation. As discussed in Section 4, to generate long-range motion, we combine multiple short motions with transition tokens, which can be generated in parallel. Since the transitions are short and our method's inference speed is relative to the motion length, transition generation can be completed in just a single iteration, as opposed to the thousand steps required by diffusion-based models. In particular, we can generate a 10.873-minute sequence in only 1.65 seconds.
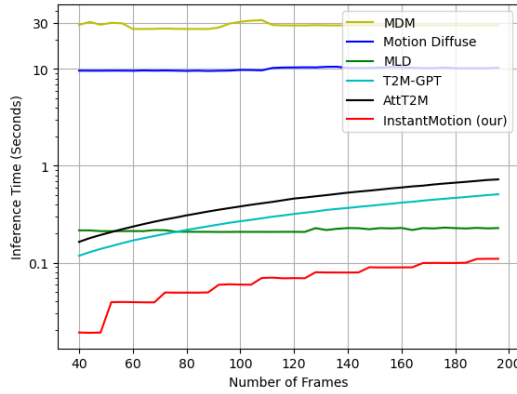

Figure 11. Comparison of Average Inference Time per Sentence (AITS) by the length of motion sequence (number of frames).

It is worth noting that the HumanML3D [12] dataset is heavily skewed towards 196-frame samples (around 1680 samples), while all other shorter samples have less than 200 samples per length. Since our method's speed is relative to the length of the generated samples, the reported average speed in Table 7 could be even significantly lower if the number of samples by length was equal. The distribution of samples by their length is visualized in Figure 12.
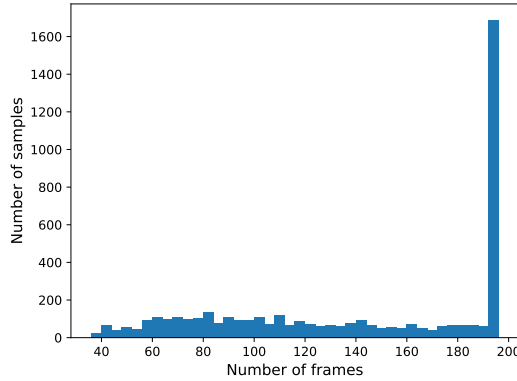

Figure 12. The number of samples in HumanML3D [12] test set by length

G. Mask Scheduling Function

As discussed in the previous section, the mask scheduling function aims to determine the number of tokens to be masked during each iteration during inference. We experiment with three mask scheduling functions to study their influence on motion generation, varying the number of iterations. Let L denote the length of generated motion and The three mask scheduling functions are described as follows:

1. **Cosine** function represents a concave dependency which can be written as $n_M(t) = L \cdot cos(\frac{1}{2}\pi\frac{t}{T_{dyn}})$

2. **Linear** function is simply an inverse relationship function of x and y: $n_M(t) = L \cdot \frac{T_{dyn}-t}{T_{dyn}}$

3. **Square Root** function is a representative convex function, expressed as $n_M(t) = L \cdot (1 - (\frac{t}{T_{dyn}})^2)$

In Table 10, we experiment with each mask scheduling function with 5, 10, 15, 20, 25, and 30 inference iterations. The results show that **Square Root** yields the worst result. **Linear** performs the best FID score at 15 iterations and can reach the best Top-1 R-precision of 0.519. While **Cosine** can achieve its best FID score of 0.089 at only 10 iterations and reach the best Top-1 R-precision at 0.518.

Table 10. Ablation results on different numbers of inference iterations with cosine, linear, and square root mask scheduling functions.

Number of Inference Iterations	R-Precision ↑			FID ↑	MM-Dist ↓	Diversity →	MModality ↑
	Top-1	Top-2	Top-3				
Cosine							
5	0.505	0.695	0.791	0.169	3.003	9.370	**1.414**
10	0.515	0.708	0.804	**0.089**	2.926	9.577	1.226
15	0.516	0.710	0.806	0.091	2.919	**9.576**	1.134
20	**0.518**	0.710	0.807	0.096	2.912	9.590	1.058
25	0.517	0.710	0.805	0.102	2.911	9.682	1.008
30	0.515	**0.711**	**0.807**	0.100	**2.908**	9.698	0.937
Linear							
5	0.505	0.693	0.789	0.195	2.995	9.399	**1.435**
10	0.514	0.707	0.804	0.090	2.920	9.584	1.190
15	0.518	0.710	0.805	**0.086**	2.910	**9.583**	1.133
20	0.519	0.713	0.808	0.091	**2.900**	9.605	1.057
25	0.517	0.713	0.808	0.096	**2.900**	9.715	1.023
30	**0.520**	0.713	**0.808**	0.103	2.907	9.632	0.9774
Square Root							
5	0.497	0.685	0.781	0.379	3.057	9.271	**1.511**
10	0.508	0.701	0.798	0.174	2.960	9.399	1.297
15	0.512	0.705	0.801	0.119	2.933	9.448	1.241
20	0.516	0.708	0.804	0.099	2.916	9.582	1.174
25	0.518	0.709	0.805	**0.092**	**2.909**	**9.540**	1.185
30	**0.520**	**0.712**	**0.807**	0.093	2.910	9.610	0.9815

H. Token Sampling Strategies for Parallel Decoding

After the tokens are masked out during each iteration, the masked tokens are decoded in parallel in the next iteration. The decoding process is based on stochastic sampling, where the tokens are sampled based on their prediction confidences. In particular, we investigate three sampling strategies, including temperature, top-k, and top-p, on the motion generation performance. These sampling strategies are widely adopted by natural language generation tasks, where top-p sampling generally shows superior performance compared with top-k and temperature-based sampling. However, our experiments show that the temperature-based and top-k sampling yields the best performance for the motion generation task.

Temperature Sampling. Temperature sampling generates the tokens according to the softmax distribution function shaped by a temperature parameter β, as shown below:

$$p(y_i|Y_{\bar{M}}, W) = \frac{\exp(e_i/\beta)}{\sum_{i \in E} \exp(e_i/\beta)}$$

where e_i is logits for motion code y_i from the codebook and E is the motion codebook. Low temperature gives more weight to the motion tokens with high prediction confidence. High-temperature trends to sample the tokens with equal probability.

Table 11 shows that with temperature $\beta = 1$, we can achieve the best overall generation performance measured by FID, while maintaining competitive performance for other metrics.

Table 11. Ablation results on different temperatures.

| Temperature (β) | R-Precision ↑ | | | FID ↑ | MM-Dist ↓ | Diversity → | MModality ↑ |
	Top-1	Top-2	Top-3				
.5	**0.517**	**0.714**	**0.810**	0.098	**2.893**	9.725	0.744
1	0.515	0.708	0.804	**0.089**	2.926	9.577	1.226
1.2	0.504	0.695	0.792	0.140	2.998	**9.484**	1.446
1.5	0.478	0.664	0.761	0.495	3.186	9.211	**1.884**

Top-k Sampling. Top-k sampling samples the token from the top k most probable choices. In particular, it first finds the top-k codebook entries to form a new codebook consisting of k entries, $E^k \in E$, which maximizes $\sum_{i \in E^k} p(y_i | Y_{\bar{M}}, W)$. Then, the original distribution is re-scaled to a new distribution $p'(y_i | Y_{\bar{M}}, W)$, from which the motion token is sampled. In particular, $p'(y_i | Y_{\bar{M}}, W) = p(y_i | Y_{\bar{M}}, W)/p_{sum}$, where $p_{sum} = \sum_{i \in E^k} p(y_i | Y_{\bar{M}}, W)$. "10%" refers to the sampling from only the top 10% of most probable motion tokens from the codebook entries."100%" indicates the use of all codebook entries. As shown in Table 12, $k = 100\%$ leads to the best performance, which is mathematically equivalent to temperature sampling with temperature $\beta = 1$.

Table 12. Ablation results on different probability of Top-K sampling.

| Top-k % | R-Precision ↑ | | | FID ↑ | MM-Dist ↓ | Diversity → | MModality ↑ |
	Top-1	Top-2	Top-3				
10%	0.511	0.705	0.803	**0.088**	2.930	9.636	1.180
30%	0.512	0.706	0.803	0.093	2.928	9.624	1.192
50%	0.515	0.707	0.804	0.093	**2.926**	9.637	1.208
70%	0.512	0.705	0.802	0.092	2.934	**9.506**	1.202
90%	0.514	0.706	0.803	0.091	2.927	9.594	1.159
100%	**0.515**	**0.708**	**0.804**	0.089	**2.926**	9.577	**1.226**

Top-p Sampling. The Top-P strategy, also called nucleus sampling, selects the highest probability tokens whose cumulative probability mass exceeds the pre-chosen threshold p. In particular, it first finds the top-p codebook $E^p \in E$, which is the smallest set such that $\sum_{i \in E^k} p(y_i | Y_{\bar{M}}, W) > p$. Then, the original distribution is re-scaled to a new distribution $p'(y_i | Y_{\bar{M}}, W)$, from which the motion token is sampled. In particular, $p'(y_i | Y_{\bar{M}}, W) = p(y_i | Y_{\bar{M}}, W)/p_{sum}$, where $p_{sum} = \sum_{i \in E^p} p(y_i | Y_{\bar{M}}, W)$. $p = 0.1$ refers to the sampling from only a set of probable tokens from the codebook entries whose summation of prediction confidences is larger than 0.1. $p = 1$ indicates the use of all codebook entries. As shown in Table 13, selecting only 0.1 of codebook entries can improve FID score, however, worsens R-precision, MM-Dist, and MModality.

Table 13. Ablation results on different probability of Top-P sampling.

| Probability of Top-P Sampling | R-Precision ↑ | | | FID ↑ | MM-Dist ↓ | Diversity → | MModality ↑ |
	Top-1	Top-2	Top-3				
0.1	0.511	0.704	0.802	**0.084**	2.932	**9.518**	0.060
0.3	0.516	0.711	0.807	0.093	2.909	9.589	0.480
0.5	0.516	0.711	0.807	0.091	2.905	9.622	0.739
0.7	**0.518**	**0.712**	**0.809**	0.090	**2.900**	9.645	0.963
0.9	0.516	0.710	0.806	0.089	2.910	9.636	1.079
1	0.515	0.708	0.804	0.089	2.926	9.577	**1.226**

I. Qualitative Results

I.1. Temporal Motion Editing.

In this section, we show that our method is not only able to edit the motion in-betweening task but also able to modify various temporal motion tasks such as temporal motion outpainting, temporal motion completion with text condition, and temporal motion completion without text condition, as shown in Figure 13.
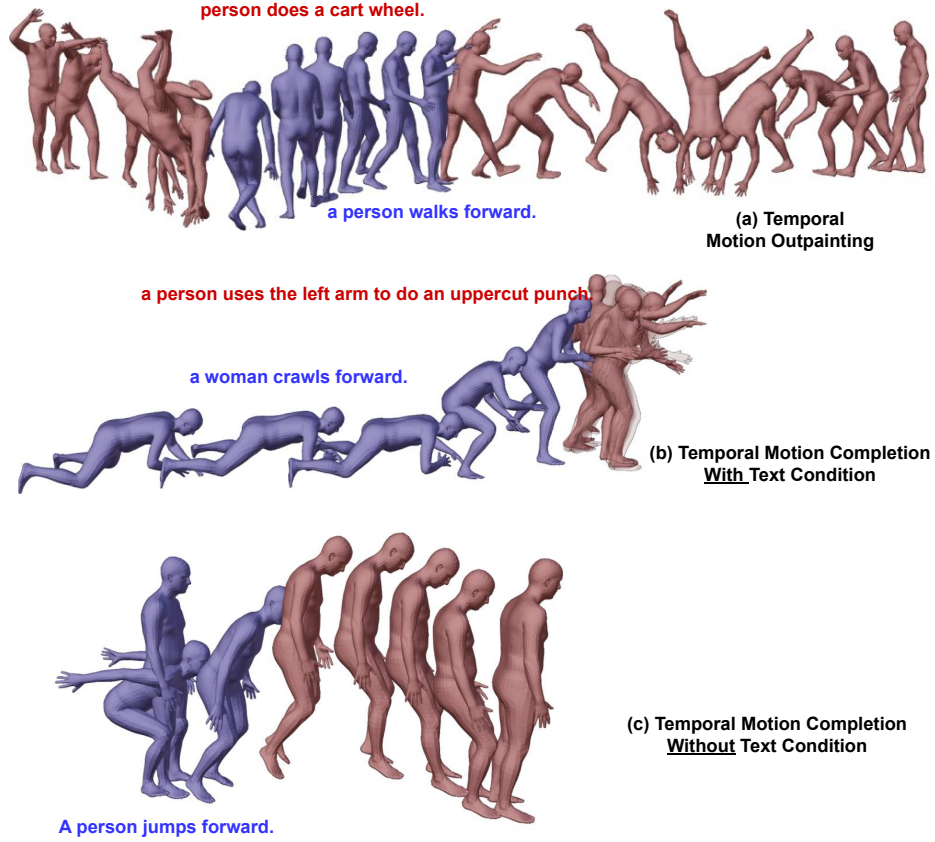

Figure 13. Qualitative results of (a) temporal motion outpainting and (b) temporal motion completion with text condition (c) temporal motion completion without text condition where the red frames indicate generated motion and the blue frames represent conditioned frames.

I.2. Upper Body Editing

The upper body editing task combines upper and lower body parts from different prompts which can lead to the out-of-distribution issue as the joint distribution of upper and lower body parts from the prompts is not present in the dataset. We overcome this challenge by introducing [MASK] tokens into the lower body condition tokens as controllable parameters to adjust the influence of lower body parts of the generation. As illustrated in Figure 14, upper body parts are generated with text "A man punches with both hands." conditioned by lower body parts "a man rises from the ground, walks in a circle and sits back down on the ground.", showing in three cases. **Top figure** shows generated upper body parts with all masked lower body tokens which means no lower body part condition at all. The result shows uncorrelated motion between upper and lower body parts. On the other hand, **middle figure** presents the generated upper body parts with all lower part conditions without any mask. The generated motion is realistic, however, less expressive as the influence from the lower body part is too strong and the joint distribution of upper and lower body parts is unseen in the training set. To solve this problem, **the bottom figure** generates motion by conditioning only some lower body tokens. This can be achieved by applying [MASK] tokens to some of the lower body tokens. The result shows that the motion is realistic while expressing very strong corresponding motion to both upper and lower body parts.
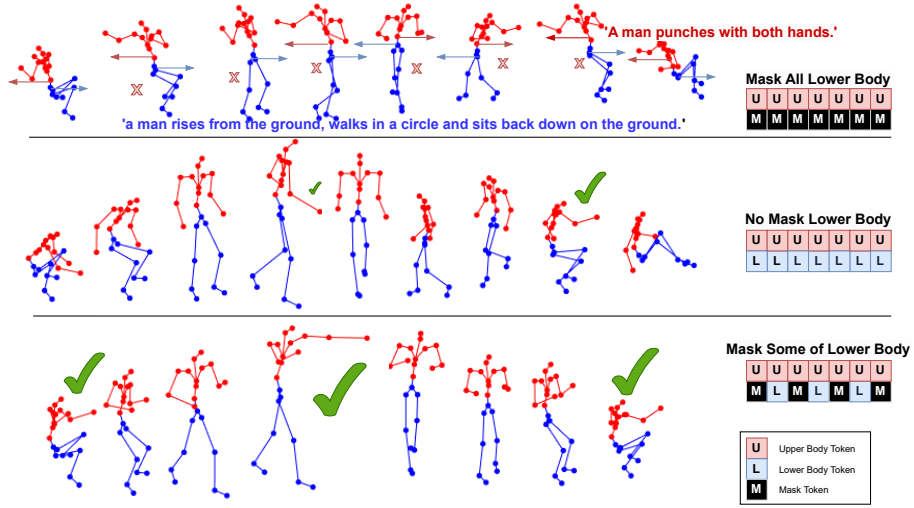
Figure 14. Effect of lower body masked tokens. "✓" shows frames that exhibit strong correlation to the textual description of the upper body part

I.3. Long Sequence Generation

In Figure 15, we generate a long motion sequence by combining multiple text prompts. First, our model generates the motion token sequence for each prompt (red frames). Then, we generate transition motion tokens (blue frames) conditioned on the end of the previous motion sequence and the start of the next motion sequence.
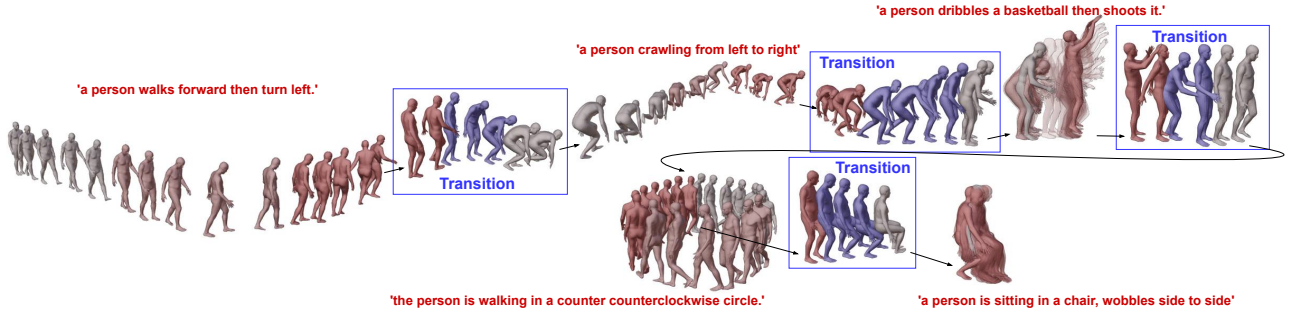

Figure 15. Long Sequence Generation. An arbitrary long motion sequence is generated from multiple prompts (red frames) combined with transitions (blue frames)

J. Limitations

Our model may face challenges in rendering some fine-grain details for exceptionally long single textual descriptions. This is due to limitations in text-to-motion training datasets, which support motions up to a maximum of 196 frames. To address this, we are exploring how our model's long motion generation capabilities can be leveraged. Specifically, we aim to integrate large language models to effectively segment a lengthy texture description into several concise text prompts. Additionally, our current model does not support the generation of interactive motions involving multiple individuals. This limitation is not unique to our model but is also present in other competing methods.