MoLingo: Motion–Language Alignment for Text-to-Human Motion Generation

Supplementary Material

Table 4. **Notation Table**. The main notation used in our paper.

Symbol	Description	Domain
N	Input motion sequence length	$\mathbb{N}$
D	Pose configuration dimension	$\mathbb{N}$
l	Motion latent length	$\mathbb{N}$
d	Latent dimension	$\mathbb{N}$
D_h	Hidden size of Transformer decoder	$\mathbb{N}$
l_{adapter}	Text adapter depth	$\mathbb{N}$
l_{text}	Text token length for cross attention	$\mathbb{N}$
τ	Threshold for filtering repetitive class tokens	$\mathbb{N}$
$\mathbf{m}$	Input motion sequence	$\mathbb{R}^{N\times D}$
m	Motion latent sequence	$\mathbb{R}^{l\times d}$
κ	Class token sequence	$\mathbb{R}^{l\times d}$
$\mathcal{I}$	Index set of chosen positions for $\mathcal{L}_{\text{sem}}$	-
$\mathbf{w}$	Text representation for cross attention	$\mathbb{R}^{l_{\text{text}}\times D_h}$
$\mathcal{E}$	Encoder	-
$\mathcal{D}$	Decoder	-
v_θ	Rectified flow MLP	-

In this supplementary document, we first present additional implementation details, including autoencoder training, the auto-regressive denoising model, integration with the RL tracking controller, and the evaluation metrics. Next, we report further quantitative results, such as effect of text adapter, comparisons with MotionStreamer [69] and a broader set of benchmarks under the TMR-263 [46] evaluation protocol. Finally, we include an ablation study that highlights the impact of our repetitive class-token filtering design during SAE training. The main notation used in our paper is shown in Tab. 4.

6. More Implementation Details

Autoencoder training. We adopt the causal autoencoder architecture from [69] with a hidden size of 1024, and provide a detailed specification in Tab. 10. Unlike [69], we address an issue happening when the 1D convolutional kernel slides from the beginning to the end of the motion sequence: kernels at the start of the sequence encounter padded values (set to 0 in previous works), which leads to jitter in the decoded motion during the first few frames. To mitigate this, for the first convolutional layer of the encoder, we replace zero padding with replicated padding, resulting in more stable training and smoother reconstructions. Tab. 8 shows our SAE achieves lower reconstruction FID (rFID) while maintaining comparable MPJPE compared with MotionStreamer [69]'s VAE. In practice, we set $\lambda_{\text{joint}} = 1.0$ and $\lambda_{\text{vel}} = 10.0$. For both the VAE and SAE, we use $\lambda_{\text{KL}} = 1 \times 10^{-5}$. For the SAE, based on the ablation results in Tab. 3, we set $\lambda_{\text{sem}} = 0.001$. The threshold τ for filtering repetitive class token is set to 0.995.

Table 5. **Effect of text adapter depth.** We compare models without a text adapter and with text adapters of different depths (3, 6, and 9 layers). Using a 6-layer adapter (ours) gives the best overall trade-off, improving FID and R-Precision across all others. We conduct all experiments on MARDM-67 [43] evaluator using a VAE with a downsampling 4× and latent dimension 16.

Configuration	FID ↓	R-Precision ↑		
		Top 1	Top 2	Top 3
w/o text adapter	$0.057^{\pm.001}$	$0.524^{\pm.003}$	$0.715^{\pm.001}$	$0.810^{\pm.002}$
depth=3	$0.060^{\pm.002}$	$0.521^{\pm.002}$	$0.713^{\pm.002}$	$0.807^{\pm.002}$
depth=6 (Ours)	$0.049^{\pm003}$	$0.528^{\pm.002}$	$0.721^{\pm.002}$	$0.815^{\pm.002}$
depth=9	$0.053^{\pm.002}$	$0.527^{\pm.002}$	$0.719^{\pm.002}$	$0.813^{\pm.001}$

Table 6. **Effect of repetitive class token filtering.** We ablate generative performance with and without repetitive class token filtering during SAE training. Filtering consistently improves FID and retrieval scores, indicating that it acts as a soft semantic regularizer, encouraging coherence without forcing adjacent motion latents to collapse to the same text label.

Configuration	FID ↓	R-Precision ↑		
		Top 1	Top 2	Top 3
w/o filtering	$0.096^{\pm.003}$	$0.524^{\pm.002}$	$0.713^{\pm.002}$	$0.811^{\pm.002}$
w/ filtering (Ours)	$0.066^{\pm003}$	$0.544^{\pm.002}$	$0.739^{\pm.002}$	$0.832^{\pm.002}$

Obtaining text labels for a specific motion latent. During SAE training, to obtain text labels that are temporally aligned with a given motion latent m_i, we exploit the causal design: each latent depends only on past frames, not future ones. As shown in Tab. 10, our encoder comprises four causal 1D convolutional layers in total (excluding the ResNet layers). The first and last layers use a kernel size of 3 and stride of 1 to preserve the sequence length, while the two middle layers use a kernel size of 4 and stride of 2, each downsampling the temporal resolution by a factor of 2. This yields a latent sequence whose length is one quarter of the original. As a result, latent index i is influenced by the original frames from $4i - 16$ to $4i + 4$, covering 20 frames in total. We aggregate the frame-level text labels over the same temporal window from $4i - 16$ to $4i + 4$, encode them with T5-Large to obtain embeddings, and then take the average of these embeddings before feeding them into an MLP projection.

Auto-regressive motion latent denoising. Our Transformer for predicting z is a standard Transformer decoder [62] with 16 layers, each using 16 attention heads and a hidden size of $D_h = 1024$. For denoising, we use an MLP with 8 residual blocks and a width of 1280 channels. Each block applies LayerNorm, a linear layer, a SiLU [11] activation, and another linear layer, followed by a residual connection. The

Table 7. **Averaged Inference Time** (AIT, in seconds) and generative performance under different inference and sampling steps.

Inf. Step	Samp. Step	AIT(s) ↓	FID ↓	R-Precision			CLIP-Score ↑
				Top 1↑	Top 2↑	Top 3↑	
49	32	9.02	$0.056^{\pm.005}$	$0.543^{\pm.003}$	$0.737^{\pm.002}$	$0.830^{\pm.002}$	$0.685^{\pm.001}$
24	32	4.43	$0.057^{\pm.004}$	$0.541^{\pm.002}$	$0.737^{\pm.002}$	$0.833^{\pm.002}$	$0.685^{\pm.001}$
16	32	2.83	$0.066^{\pm.003}$	$0.544^{\pm.002}$	$0.739^{\pm.002}$	$0.832^{\pm.002}$	$0.686^{\pm.001}$
5	32	0.83	$0.105^{\pm.005}$	$0.537^{\pm.003}$	$0.734^{\pm.002}$	$0.829^{\pm.002}$	$0.685^{\pm.001}$
16	8	1.58	$0.066^{\pm.003}$	$0.541^{\pm.002}$	$0.737^{\pm.002}$	$0.830^{\pm.002}$	$0.683^{\pm.001}$
16	100	7.00	$0.067^{\pm.002}$	$0.543^{\pm.002}$	$0.739^{\pm.002}$	$0.833^{\pm.002}$	$0.685^{\pm.001}$

MLP is conditioned on the z predicted by the Transformer: we add z to the time embedding of the noise-schedule step t, and use this combined signal to modulate the LayerNorm layers via AdaLN. We set $l_{\text{text}} = 128$. During inference, we use a stochastic sampler that, at each step, applies a small noise refresh, partially replacing the clean component with fresh prior noise. This maintains stochasticity and improves sample diversity compared to a purely deterministic ODE trajectory. To avoid confusion, we distinguish two terms. **Denoising step** is the number of denoising iterations for the reverse ODE process during denoising, whereas **inference step** is the number of auto-regressive iterations used to sample the motion latent sequence. Tab. 7 shows MoLingo can generate per motion **under 1 sec.** (e.g., **0.83s**) while still maintaining strong generative quality, outperforming the majority of baselines. Meanwhile, higher steps consistently result in lower FID and higher R-Precision, so after tuning this trade-off we adopt **16 inf. steps and 32 samp. steps** as our default setting: it delivers SOTA performance while keeping generation faster compared to higher-step settings.

User study interface. The detailed instructions and interface layout of our user study are shown in Fig. 7. In each trial, we randomly select 15 samples and randomly assign them to the left or right side. We prepare three questionnaires with the same layout, each comparing our method against a different baseline. For comparison with MotionStreamer [69], we use the model trained with the 272D representation, and for comparisons with MoMask [18] and DisCoRD [7], we use the model trained with the 263D representation.

Integrate MoLingo with RL tracking controller. To make the generated motions more physically plausible, we integrate each generator with a pre-trained RL tracking controller. Specifically, we first re-train a PHC [40] policy for the Unitree G1 humanoid to track the joint positions of human motions. After generating motions with each candidate generator, we retarget them to the humanoid's body shape and then directly deploy the tracking policy to assess performance. Because the 272D representation contains valid rotational components, retargeting is more convenient; thus, we conduct this evaluation using the 272D model and compare against MotionStreamer. The policy is trained in IsaacGym [42] with 2048 parallel environments. After obtaining the generated motions, we first retarget them to match G1's morphology and extract the corresponding joint position

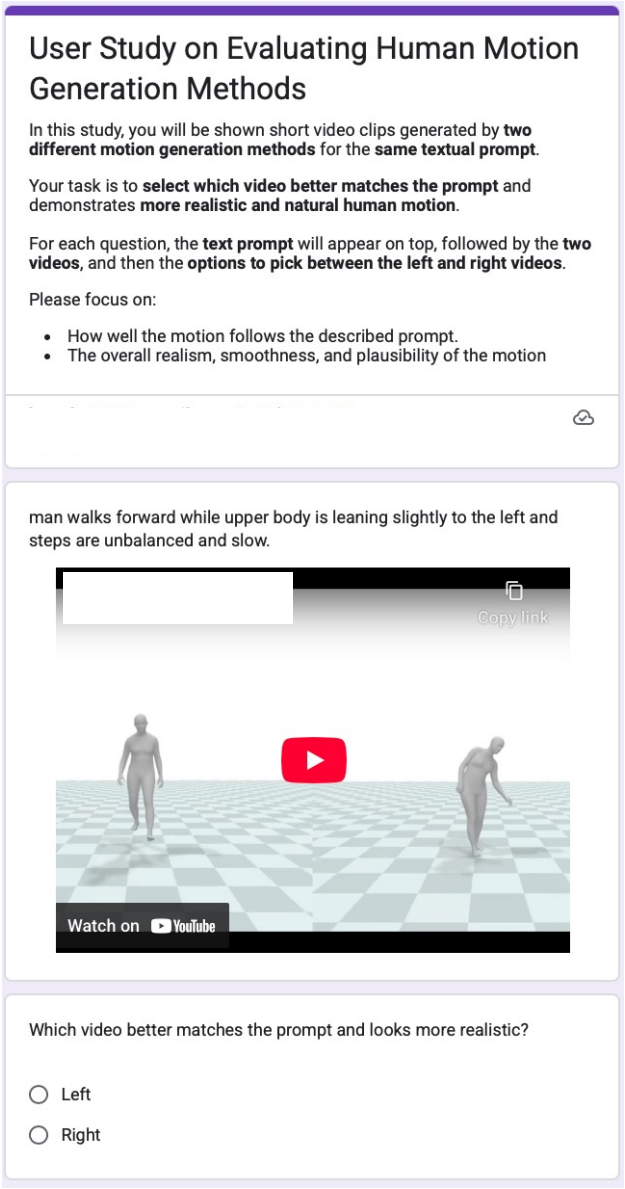

Figure 7. **User study interface.**

sequences. These joint positions are then provided to the policy to roll out action trajectories. Finally, we replay the resulting actions in the MuJoCo [60] for visualization. Video comparisons are available on our local webpage.

Details of evaluation metrics. We use the evaluation metrics following [16, 43, 69]: (1) **Fréchet Inception Distance (FID)** quantifies how close the generated motions are to real ones by comparing the overall distributions of predicted and ground-truth motion sequences, by applying a pre-trained feature extractor for both motion and text. (2) **R-Precision** (Top-1/Top-2/Top-3) together with Matching-score evaluates text–motion consistency by checking whether generated motion embeddings retrieve their paired text prompts among

Table 8. **Quantitative comparison with MotionStreamer.** MotionStreamer proposed a TMR-style [46] feature extractor used as an evaluator in their own 272D representation. To ensure a fair comparison, we train our model on the HumanML3D-272 [69] dataset and evaluate using their evaluator. rFID denotes reconstruction FID, and MPJPE is measured in millimeters. Our replicated padding design improves reconstruction realism in terms of rFID and MPJPE. For generation, our method achieves significant improvements over MotionStreamer across all metrics. We run the experiment 20 runs and $\pm$ indicates 95% confidence interval. We apply $2\times$ downsampling and latent dimension 32 with a well-trained SAE.

Methods	rFID↓	MPJPE↓	FID↓	R-Precision			Matching Score↓
				Top 1↑	Top 2↑	Top 3↑	
Real	0.000	0.000	$0.002^{\pm.000}$	$0.702^{\pm.000}$	$0.864^{\pm.000}$	$0.914^{\pm.000}$	$15.151^{\pm.000}$
MotionStreamer [69]	0.661	22.90	$11.979^{\pm.078}$	$0.629^{\pm.004}$	$0.791^{\pm.003}$	$0.858^{\pm.003}$	$16.019^{\pm.017}$
MoLingo (Ours)	0.026	9.87	$3.415^{\pm.034}$	$0.776^{\pm.003}$	$0.902^{\pm.002}$	$0.941^{\pm.001}$	$14.712^{\pm.017}$

Table 9. **Quantitative results on the TMR-263 evaluator.** We compare our method with a broad set of motion generation approaches, from early models [6, 58] to recent ones [7, 88], covering pose-frame diffusion [58], single-vector latent diffusion [6, 9], VQ-based next-token prediction [7, 18, 19, 48, 49, 72], and continuous-valued auto-regressive models [88]. We report the mean results over 20 independent runs, and the $\pm$ values indicate the 95% confidence interval. We do not compare with MARDM [43] and ACMDM [44] here because of the motion representation inconsistency. Our method achieves state-of-the-art FID, R-Precision, and MultiModality. Green cells highlight the best scores, and yellow cells the second best.

Methods	FID↓	R-Precision			MModality↑
		Top 1↑	Top 2↑	Top 3↑	
Real	$0.000^{\pm.000}$	$0.676^{\pm.002}$	$0.810^{\pm.002}$	$0.861^{\pm.002}$	-
MDM-50Step [58]	$0.093^{\pm.000}$	$0.563^{\pm.007}$	$0.723^{\pm.007}$	$0.794^{\pm.006}$	$34.207^{\pm.598}$
MLD [6]	$0.052^{\pm.000}$	$0.579^{\pm.006}$	$0.729^{\pm.007}$	$0.797^{\pm.004}$	$32.481^{\pm.678}$
MoMask [18]	$0.022^{\pm.000}$	$0.687^{\pm.002}$	$0.825^{\pm.002}$	$0.877^{\pm.002}$	$18.406^{\pm.583}$
MMM [49]	$0.024^{\pm.000}$	$0.710^{\pm.002}$	$0.834^{\pm.002}$	$0.881^{\pm.001}$	$15.286^{\pm.577}$
BAMM [48]	$0.044^{\pm.000}$	$0.652^{\pm.002}$	$0.790^{\pm.002}$	$0.845^{\pm.002}$	$28.703^{\pm.672}$
MogenTS [72]	$0.017^{\pm.000}$	$0.687^{\pm.003}$	$0.819^{\pm.002}$	$0.870^{\pm.002}$	$12.102^{\pm.394}$
MLD++ [9]	$0.019^{\pm.000}$	$0.736^{\pm.003}$	$0.863^{\pm.002}$	$0.907^{\pm.002}$	$26.515^{\pm.621}$
MotionLCM-V2 [9]	$0.026^{\pm.000}$	$0.751^{\pm.003}$	$0.879^{\pm.001}$	$0.921^{\pm.001}$	$28.424^{\pm.753}$
DisCoRD [7]	$0.020^{\pm.000}$	$0.692^{\pm.002}$	$0.828^{\pm.002}$	$0.878^{\pm.001}$	$18.804^{\pm.613}$
MotionGPT3 [88]	$0.021^{\pm.000}$	$0.758^{\pm.002}$	$0.873^{\pm.002}$	$0.915^{\pm.001}$	$20.821^{\pm.426}$
MoMask++ [19]	$0.020^{\pm.000}$	$0.692^{\pm.003}$	$0.824^{\pm.002}$	$0.874^{\pm.002}$	$16.589^{\pm.439}$
MoLingo	$0.015^{\pm.000}$	$0.771^{\pm.002}$	$0.888^{\pm.001}$	$0.925^{\pm.001}$	$20.400^{\pm.519}$

the top candidates. (3) **MultiModality** (MModality) reflects variation under identical prompts, measuring how diverse the produced motion embeddings are when generating multiple motions for the same text. (4) **CLIP-Score** measures textual faithfulness via CLIP [53], computed as the cosine similarity between CLIP embeddings of the generated motion and its text description.

7. More Quantitative Results

Effect of the text adapter. Tab. 5 reports an ablation over different numbers of text adapter layers. Using the adapter improves generative performance in both FID and R-Precision compared to not using it, as it strengthens text–motion communication during training. We select a depth of 6 layers, which yields the best performance.

Comparison with MotionStreamer. To ensure a fair comparison, we use the evaluator from [69], trained on the HumanML3D-272 [69] with the TMR contrastive strategy. We compare FID, R-Precision, and Matching Score as reported under MotionStreamer's original evaluation setting. As shown in Tab. 8, Molingo achieves significant improvements on all metrics. We apply CFG scale 7.0 for 272D format generation. The experiment is conducted with HumanML3D-272 test set.

Quantitative results on the TMR-263 evaluator [46]. We further provide, for the first time, a benchmark evaluating a broad set of methods on the TMR-263 evaluator. TMR is trained with an improved contrastive loss, yielding a

better cross-modal embedding space. We rerun a wide range of baseline methods on this evaluator, as shown in Tab. 9, MoLingo still achieves state-of-the-art performance across all methods. The experiment is conducted with HumanML3D [16] test set.

Effect of repetitive token filtering. During SAE training, to ablate the effect of repetitive class-token filtering, Tab. 6 reports the generative performance with and without filtering. Removing adjacent repetitive class tokens acts as a softer regularizer and yields significantly better performance.

Table 10. **Detail architecture** of our autoencoders. Different from [69], we set the padding mode of the first 1D convolutional layer to replicate the initial frames, which stabilizes training and improves reconstruction.

Components	Architecture
Encoder	(0): CausalConv1D(D, 1024, kernel_size=(3,), stride=(1,), dilation=(1,), padding=((2,), mode=replicate)) (1): SiLU() (2): 2 × Sequential((0): CausalConv1D(1024, 1024, kernel_size=(4,), stride=(2,), dilation=(1,), padding=((2,), mode=zero)) (1): CausalResnet1D((0): CausalResConv1DBlock((activation1): SiLU() (conv1): CausalConv1D(1024, 1024, kernel_size=(3,), stride=(1,), dilation=(9,), padding=((18,), mode=zero)) (activation2): SiLU() (conv2): CausalConv1D(1024, 1024, kernel_size=(1,), stride=(1,), dilation=(1,), padding=((0,), mode=zero))) (1): CausalResConv1DBlock((activation1): SiLU() (conv1): CausalConv1D(1024, 1024, kernel_size=(3,), stride=(1,), dilation=(3,), padding=((6,), mode=zero)) (activation2): SiLU() (conv2): CausalConv1D(1024, 1024, kernel_size=(1,), stride=(1,), dilation=(1,), padding=((0,), mode=zero))) (2): CausalResConv1DBlock((activation1): SiLU() (conv1): CausalConv1D(1024, 1024, kernel_size=(3,), stride=(1,), dilation=(1,), padding=((2,), mode=zero)) (activation2): SiLU() (conv2): CausalConv1D(1024, 1024, kernel_size=(1,), stride=(1,), dilation=(1,), padding=((0,), mode=zero))))) (3): CausalConv1D(1024, 1024, kernel_size=(3,), stride=(1,), dilation=(1,), padding=((2,), mode=zero))
Decoder	(0): CausalConv1D(d, 1024, kernel_size=(3,), stride=(1,), dilation=(1,), padding=((2,), mode=zero)) (1): SiLU() (2): 2 × Sequential((0): CausalResnet1D((0): CausalResConv1DBlock((activation1): SiLU() (conv1): CausalConv1D(1024, 1024, kernel_size=(3,), stride=(1,), dilation=(9,), padding=((18,), mode=zero)) (activation2): SiLU() (conv2): CausalConv1D(1024, 1024, kernel_size=(1,), stride=(1,), dilation=(1,), padding=((0,), mode=zero))) (1): CausalResConv1DBlock((activation1): SiLU() (conv1): CausalConv1D(1024, 1024, kernel_size=(3,), stride=(1,), dilation=(3,), padding=((6,), mode=zero)) (activation2): SiLU() (conv2): CausalConv1D(1024, 1024, kernel_size=(1,), stride=(1,), dilation=(1,), padding=((0,), mode=zero))) (2): CausalResConv1DBlock((activation1): SiLU() (conv1): CausalConv1D(1024, 1024, kernel_size=(3,), stride=(1,), dilation=(1,), padding=((2,), mode=zero)) (activation2): SiLU() (conv2): CausalConv1D(1024, 1024, kernel_size=(1,), stride=(1,), dilation=(1,), padding=((0,), mode=zero))))) (1): Upsample(scale_factor=2.0, mode=nearest) (2): CausalConv1D(1024, 1024, kernel_size=(3,), stride=(1,), dilation=(1,), padding=((2,),mode=zero)) (3): CausalConv1D(1024, 1024, kernel_size=(3,), stride=(1,), dilation=(1,), padding=((2,),mode=zero)) (4): SiLU() (5): CausalConv1D(1024, D, kernel_size=(3,), stride=(1,), dilation=(1,), padding=((2,),mode=zero))
Autoencoder	(0): Encoder(D, 1024) (1): MLP(1024, d) (2): Decoder(d, D)