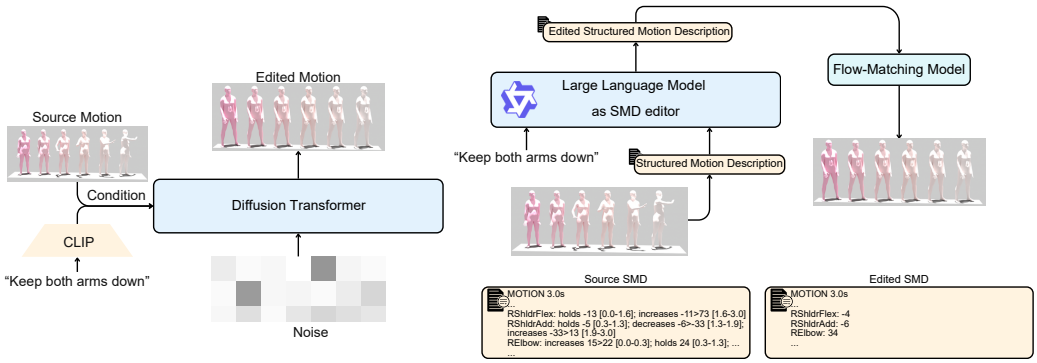
EDITING HUMAN MOTION THROUGH STRUCTURED MOTION DESCRIPTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-driven human motion editing aims to modify a source motion sequence according to natural language instructions, enabling users to iteratively refine generated motions. However, current diffusion-based methods denoise the entire motion sequence from random noise within an opaque latent space. This makes it difficult to preserve regions that should remain unchanged and limits the interpretability of the editing process. In this paper, we leverage Structured Motion Descriptions (SMD) to represent skeletal motions as readable text and perform edits directly via a Large Language Model (LLM) guided by instructions. By converting motion into SMD, the LLM selectively modifies only the instruction-relevant regions to achieve precise, localized edits; meanwhile, the readability of SMD enables direct inspection of which joints are altered, over which temporal intervals, and by what magnitude, offering interpretability. We implement this framework through a three-stage training pipeline: supervised fine-tuning to teach the LLM instruction-guided SMD editing, a flow-matching model to enable bidirectional conversion between motion and SMD, and a Group Relative Policy Optimization (GRPO) stage to improve the LLM's numerical accuracy for higher motion fidelity. Extensive experiments on MotionFix demonstrate that our approach achieves state-of-the-art performance while providing interpretability. Code is available in the supplementary materials.

1 INTRODUCTION



(a). diffusion-based motion editing (b). our approach: SMD-based motion editing

Figure 1: Comparison with diffusion-based approaches during inference. **(a)** Previous diffusion-based methods encode the editing instruction into an opaque conditioning vector and denoise random noise to regenerate the entire motion sequence. **(b)** Our approach first converts the source motion into SMD, a readable per-joint textual representation. An LLM then edits the SMD according to the instruction, and a flow-matching model decodes the edited SMD back into natural motion.

Generating human motion from natural language has progressed rapidly. Given a caption such as "a person walks forward and then sits down", current models can produce motions that are both realistic and well aligned with the text (Tevet et al., 2022; Chen et al., 2023; Guo et al., 2024).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

However, usable motion is rarely obtained in a single generation. Animators, game designers, and robotics engineers typically begin with a relevant motion and then make targeted modifications, such as raising the arms higher, taking smaller steps, or turning left instead of right. Text-based motion editing addresses this setting by taking a source motion and an editing instruction as input and producing the corresponding edited motion.

Recent work on motion editing has driven substantial progress using diffusion models, which generate edited motions by conditioning a denoiser on both the source motion and the edit instruction (Athanasiou et al., 2024; Jiang et al., 2025; Li et al., 2025; Han & Kim, 2026; Shi et al., 2026) (Figure 1a). Precise motion editing requires localized modification; however, this is difficult for diffusion-based approaches because they typically regenerate the entire motion sequence, including segments and body parts that are not intended to be edited. Although recent methods (Jiang et al., 2025; Han & Kim, 2026) introduce auxiliary objectives and architectural constraints to improve fidelity in unedited regions, preservation is not guaranteed because the final motion is still produced through denoising from random noise. A second limitation concerns interpretability. The editing instruction is compressed into an opaque conditioning vector, and the edit takes place inside the diffusion model. As the result, the pipeline does not expose which joints were altered or by what magnitude. This restricts the analysis of the editing effects largely to qualitative visual inspections of rendered motion sequences.

To address these limitations, we build on Structured Motion Descriptions (SMD) (Zhang et al., 2026b), an interpretable representation that abstracts continuous skeletal motion into structured, readable text by recording joint-angle extrema and their timestamps. As illustrated in Figure 1b, our method first converts the source motion into SMD, which an LLM then edits according to the instruction. To decode sparse SMD extrema back into natural continuous motion and ensure that the LLM produces numerically precise edits, we propose a three-stage training pipeline (Section 3): supervised fine-tuning for instruction-grounded SMD editing, a flow-matching model for high-fidelity SMD-to-motion decoding, and Group Relative Policy Optimization (GRPO) (Shao et al., 2024) for numerical refinement.

Extensive experiments on MotionFix show that our framework establishes a new state of the art. Under the standard evaluation protocol, it lowers the average rank over the full test set from 13.06 to 9.61 and the FID from 0.091 to 0.045 (Section 4.3). It also maintains its lead across five independent evaluators (Section 4.4), demonstrating both superior and robust performance.

Our contributions are threefold. First, we show that editing in SMD space enables localized motion modification, allowing the model to alter only the specified joints and temporal regions while preserving unrelated motion (Section 4.5). Second, we introduce an interpretable motion editing framework grounded in a structured per-joint textual representation: by comparing the source and edited SMD, users can inspect which joints were changed, over which temporal intervals, and by what magnitude. Third, we make SMD practical for motion editing by introducing flow-matching decoding and GRPO-based numerical refinement, enabling edited SMD representations to be converted into natural, high-fidelity motion.

2 Related Work

Text-to-motion generation Text-to-motion generation aims to synthesize 3D human motion sequences from natural language descriptions. A prominent line of work uses diffusion models to estimate the motion distribution, either in raw pose space (Tevet et al., 2022) or in a compressed latent space (Chen et al., 2023). Another line of work employs discrete vector quantization (e.g., VQ-VAE) coupled with autoregressive or masked modeling to generate motion tokens from text (Zhang et al., 2023a; Guo et al., 2024; Zhu et al., 2026; Wu et al., 2025; Fan et al., 2025; Jiang et al., 2026; Wu et al., 2026). Although these approaches achieve high-fidelity synthesis, their intermediate representations are difficult for humans to inspect or edit, as they are encoded as implicit latents or learned discrete tokens.

Recent work has therefore begun to introduce more interpretable intermediate representations. CoMo (Huang et al., 2024) decomposes poses into semantically labelled per-body-part codes for LLM-based motion generation, while KinMo (Zhang et al., 2025) generates motion hierarchically by attaching fine-grained descriptions to kinematic joint groups. These methods suggest that expos-

ing a symbolic or semantically structured layer can improve controllability and human interpretability. Motivated by this direction, we build on Structured Motion Descriptions (SMD) (Zhang et al., 2026b), which translate motion into readable text by recording continuous joint-angle extrema and their timestamps. Compared with discretized tokens or latent codes, SMD is particularly suitable for text-driven editing because it preserves numerical motion attributes in a human-readable form, making it possible to inspect which joints change, when they change, and by how much.

Recent motion generation methods have also incorporated reinforcement learning (RL) to improve alignment. Group Relative Policy Optimization (GRPO) (Shao et al., 2024), following its success in eliciting reasoning capabilities in models such as DeepSeek-R1 (Guo et al., 2025), has increasingly been adopted in downstream domains. HY-Motion 1.0 (Wen et al., 2025), for example, uses a Flow-GRPO variant to align a large DiT model with semantic and physical constraints. Other than this, Motion-R1 (Ouyang et al., 2025) combines chain-of-thought reasoning with an RL binding mechanism, while MotionRL (Liu et al., 2024) uses multi-reward optimization to balance text adherence and motion quality. These methods apply RL to implicit latent or token-based motion generators. In contrast, we use GRPO to improve the numerical precision of LLM-generated SMD.

Motion editing. Motion editing aims to modify skeletal motion according to user-specified instructions. Early approaches relied on attention manipulation (Chen et al., 2024), keyframe constraints (Goel et al., 2024), static pose correction (Delmas et al., 2023), or script-level rewriting (Zhang et al., 2023b). However, these methods do not directly support free-form natural language instructions for sequence-level editing. MotionFix (Athanasiou et al., 2024) formalized text-driven motion editing as a conditional triplet task, in which a source motion is transformed into a target motion according to an editing instruction, and introduced the diffusion-based editor TMED.

Subsequent generative approaches have advanced this framework along two main trajectories. Within diffusion-based editing, SimMotionEdit (Li et al., 2025) introduces auxiliary similarity objectives. Cross-Axis (Han & Kim, 2026) uses dual-axis spatial–temporal transformers. OmniME (Shi et al., 2026) and CIME (Lin et al., 2026) adopts positive–negative learning to balance localized editing with global preservation. PartMotionEdit (Yang et al., 2025) decomposes motion into five parts to enable more fine-grained editing in diffusion models. In parallel, flow-matching models (Liu et al., 2022; Lipman et al., 2022) have emerged as an alternative to diffusion models by learning a continuous velocity field from noise to data without stochastic denoising. Motion Flow Matching (Hu et al., 2023) apply it to editing by rewriting the sampling trajectory. Unified Conditional Flow (Li et al., 2026) formulates generation, editing and retargeting as one conditional transport problem. UniMoFlow (Hua et al., 2026) shares a velocity field between generation and editing while anchoring the sampling path to the source motion.

Despite these advances, existing diffusion- and flow-based editors still reconstruct motion through continuous generative processes, making localized edits and preservation of unedited regions difficult to inspect or control directly. In contrast, we map motion into readable SMD and formulate editing as an LLM-based text-editing task. We adopt SMD for its interpretability: by explicitly recording joint-level extrema and timestamps, it allows source and edited motions to be compared in terms of affected joints, temporal locations, and numerical changes.

3 METHOD

Given a source motion sequence $X = [x_1, x_2, \ldots, x_T]$ with T frames and an editing instruction l, text-based motion editing aims to generate an edited motion sequence $M = [m_1, m_2, \ldots, m_{T'}]$ with T' frames. Here, each $m_i \in R^D$ represents a human pose, $D = 135$ following the MotionFix protocol (Athanasiou et al., 2024), including the global translation of the root and the 6D rotation of the 22 body joints.

We use a deterministic function f_{smd} to translate X into a readable string $s = f_{\text{smd}}(X)$, following the definition of SMD (Zhang et al., 2026b), with minor modifications detailed in Appendix A. Specifically, f_{smd} computes angles for $J = 26$ biomechanical degrees of freedom from 3D joint positions. Each joint-angle trajectory is smoothed, segmented at local extrema, and encoded as a sequence of intervals. Each interval records the joint-angle trend (increase, decrease, or hold), its start and end angles, and the corresponding timestamps, as illustrated in Figure 2a. Building on this representation, we design the three-stage training pipeline shown in Figure 2b. The inference process

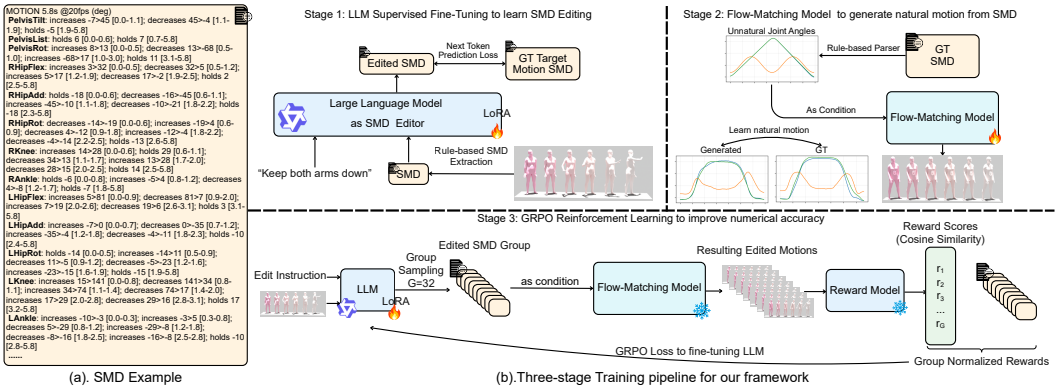


Figure 2: **(a)** Example of SMD representation. **(b)** Three-stage training pipeline: Stage 1 fine-tunes the LLM to generate edited SMD; Stage 2 trains a flow-matching decoder to convert parsed joint angles into smooth motion; Stage 3 applies GRPO, using rewards from decoded motion candidates to improve the LLM's numerical accuracy.

is illustrated in Figure 1b, where the source SMD, generated by f_{smd}, and the editing instruction l are given as input to the LLM.

3.1 Stage 1: Supervised Fine-Tuning

The goal of Stage 1 is to teach an LLM to autoregressively edit a source SMD according to the editing instruction. We adapt a pre-trained instruction-tuned LLM, Qwen2.5-7B-Instruct (Qwen et al., 2025),by supervised fine-tuning (SFT) with LoRA (Hu et al., 2021), which freezes the base model weights and introduces trainable low-rank matrices into the attention and feed-forward layers.

Training is performed on the MotionFix training set $\{(X_i, M_i, l_i)\}_{i=1}^{N}$. For each triplet, we construct the source and target SMD as $s_i^{\text{src}} = f_{\text{smd}}(X_i)$ and $s_i^{\text{tgt}} = f_{\text{smd}}(M_i)$. The training objective is the standard next-token prediction loss, masked to the target SMD tokens:

$$\mathcal{L}_{SFT} = -\sum_{t=1}^{|s^{tgt}|} \log \pi_\theta\big(s_t^{tgt} \mid s^{src}, l, s_{1:t-1}^{tgt}\big) \tag{1}$$

where s_t^{tgt} denotes the t-th token of s^{tgt}. This SFT stage teaches the LLM the syntax and structure of SMD, including valid joint names, well-formed intervals, and plausible angle ranges. After this stage, the LLM can generate syntactically valid SMD, but its numerical angle predictions are not yet sufficiently precise.

3.2 Stage 2: Flow-Matching Model

Given the edited SMD $\hat{s}$ produced by the LLM, the next step is to convert it back into a motion sequence. A rule-based approach to this conversion is to extract the peak and valley angle values and timestamps from the SMD via a deterministic parser f_{parse}, which linearly interpolates between them to recover a dense angle sequence $a = f_{parse}(\hat{s}) \in \mathbb{R}^{T\times J}$. However, this produces stiff, unnatural motions, because the acceleration and deceleration of human joints cannot be captured by linear interpolation. The goal of Stage 2 is therefore to train a flow-matching model that transforms SMD into natural motion.

Training data for the flow-matching model pairs the rule-parsed SMD angles $a = f_{parse}(f_{smd}(M))$ with the corresponding ground-truth motion $m \in \mathbb{R}^{T\times D}$ in the MotionFix training set. As shown in Figure 3a, we take the parsed angles a as the condition and train a velocity field v_ϕ that carries random noise $\epsilon \sim \mathcal{N}(0, I)$ to the ground-truth motion m along a straight path. For a flow time $t \sim \mathcal{U}(0, 1)$, the state on that path and the velocity that the model has to predict are

$$z_t = (1-t)\,\epsilon + t\,m, \qquad v = m - \epsilon, \tag{2}$$

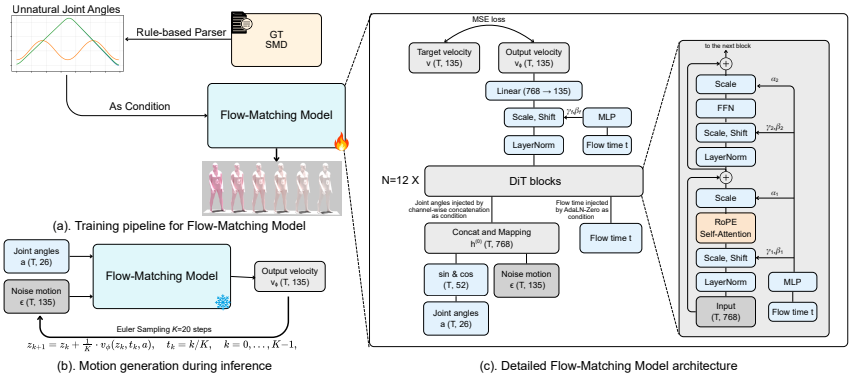


Figure 3: The design of Stage 2. **(a)** During training, the rule-based parser turns a ground-truth SMD into a joint angle sequence, which conditions the flow-matching model. **(b)** During inference, the model is frozen and a K-step Euler solver integrates the velocity field from noise into a motion. **(c)** Detailed architecture of the flow-matching model.

where v is the target velocity in Figure 3c. The model reads z_t, the flow time t and the condition a, and is trained with a mean squared error against v, averaged over the training motions and over the sampled t and ϵ:

$$\mathcal{L}_{FM} = \mathbb{E}_{m,\,a}\, \mathbb{E}_{t\sim\mathcal{U}(0,1),\, \epsilon\sim\mathcal{N}(0,I)} \left\| v_\phi(z_t, t, a) - v \right\|^2. \tag{3}$$

The flow-matching model adds natural dynamics while preserving the peak and valley angles specified in the SMD. We verify this through three tests in Section 4.6. After training, the flow-matching model is frozen for the remaining stages. The detailed architecture design and the training procedure are given in Appendix H.1.

At inference the model is an ordinary differential equation to be integrated, as illustrated in Figure 3b. Starting from pure noise $z_0 = \epsilon \sim \mathcal{N}(0, I)$, a K-step Euler solver advances the state along the learned field,

$$z_{k+1} = z_k + \tfrac{1}{K} \cdot v_\phi(z_k, t_k, a), \quad t_k = k/K, \quad k = 0, \ldots, K-1, \tag{4}$$

and z_K is the generated motion. The condition a is held fixed across all K steps and only the state and the flow time advance. We use $K = 20$ by default.

3.3 Stage 3: GRPO Reinforcement Learning

After SFT, the LLM produces syntactically valid SMD but with imprecise numerical angles. Token-level cross-entropy treats all token prediction errors equally, regardless of the magnitude of the numerical error. More importantly, the SFT loss operates in text space and cannot evaluate the quality of the resulting motion after the flow-matching model decodes it.

To bridge this gap, we apply GRPO (Shao et al., 2024), which optimizes the LLM using a reward computed in motion space. For each training prompt (s^{src}, l), the LLM model π_θ samples a group of G SMD candidates $\{\hat{s}_1, \ldots, \hat{s}_G\}$. Each candidate is parsed and decoded by the frozen flow-matching model into a motion $\hat{m}_i$, which is then scored by the reward model $\mathcal{E}$. Following HY-Motion 1.0 (Wen et al., 2025) and Motion-R1 (Ouyang et al., 2025), we use a pre-trained TMR encoder (Petrovich et al., 2023) as the reward model. The reward for the i-th candidate is:

$$r_i = \cos\big(\mathcal{E}(\hat{m}_i),\ \mathcal{E}(M^{tgt})\big) \tag{5}$$

where M^{tgt} is the ground-truth target motion. Candidates that fail to parse receive a minimum reward (-1). This reward calculates the cosine similarity between the generated motion and the GT target motion, thereby encouraging the LLM to produce SMD that is closer to the GT motion after decoding. By setting the reward to -1 when the LLM outputs non-parsable SMD, the model learns

not to output non-parsable SMD. We also compare this reward against two alternatives, and against a different reward model, in Appendix G. GRPO normalizes the advantage within each group:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^{G})}{\text{std}(\{r_j\}_{j=1}^{G})} \tag{6}$$

and updates the LLM by maximizing the clipped surrogate objective with a KL penalty to the SFT reference policy model π_{ref}:

$$\mathcal{J}_{GRPO} = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\hat{s}_i|}\sum_{t=1}^{|\hat{s}_i|}\min\Big(\rho_{i,t}\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1{-}\epsilon, 1{+}\epsilon)\,\hat{A}_i\Big) - \beta\, D_{KL}(\pi_\theta \| \pi_{ref}) \tag{7}$$

where $\hat{s}_{i,t}$ is the t-th token of the i-th candidate and $\rho_{i,t} = \pi_\theta(\hat{s}_{i,t} \mid s^{src}, l, \hat{s}_{i,<t}) \,/\, \pi_{old}(\hat{s}_{i,t} \mid s^{src}, l, \hat{s}_{i,<t})$ is the per-token importance sampling ratio. The advantage $\hat{A}_i$ is shared by every token of a candidate because the reward is a property of the decoded motion. β controls the KL penalty strength. Only the LoRA weights are updated, the flow-matching model and the reward encoder remain frozen. The reward is computed based on the motion rather than the LLM's text output. This mechanism ensures that the model can only increase its reward by producing numerically accurate SMD to avoid reward hacking.

4 Experiments

4.1 Experimental setup

Dataset. We evaluate on MotionFix (Athanasiou et al., 2024), the standard benchmark for text-based motion editing. MotionFix provides (source motion, target motion, edit instruction) triplets collected from the AMASS motion capture collection (Mahmood et al., 2019). The dataset covers diverse edit types including body-part modifications, temporal changes, and style adjustments. We use the standard train/val/test (5388, 330, 1013) split.

Implementation details. The LLM is Qwen2.5-7B-Instruct (Qwen et al., 2025) adapted with LoRA (Hu et al., 2021) (r=16, α=32). The flow-matching model v_ϕ is a DiT-style transformer velocity field with RoPE positional encoding and AdaLN-Zero conditioning. For GRPO, we use EasyR1/verl (Sheng et al., 2024) with the default group size G=32 and a learning rate of $1\text{e}{-4}$. Complete hyperparameters and implementation details for all three stages are provided in Appendix H.2. Training and inference costs for each stage are reported in Appendix I.

Baselines. We compare our method with previous baselines. TMED (Athanasiou et al., 2024) is the baseline proposed alongside the MotionFix dataset. MotionReFit (Jiang et al., 2025) introduces body-part discriminators to improve local editing fidelity. SimMotionEdit (Li et al., 2025) jointly trains motion editing with a motion similarity prediction task. Cross-Axis (Han & Kim, 2026) applies dual-axis transformers and predicts joint-wise motion differences as an auxiliary task to localize edits. OmniME (Shi et al., 2026) proposes a positive-negative learning framework that balances editing the target regions with preserving unedited parts. While Unified Conditional Flow (Li et al., 2026) and UniMoFlow (Hua et al., 2026) also focus on the motion editing task, they are currently excluded from our comparison as they have not yet open-sourced their code or provided results on the MotionFix benchmark.

4.2 Evaluation design

To comprehensively compare our approach against baselines, we conduct three evaluation experiments: (1) the standard MotionFix evaluation protocol to assess global and semantic alignment between the edited motion and the ground-truth target motion (Section 4.3); (2) a multi-evaluator evaluation to benchmark whether model performance remains robust across diverse evaluators (Section 4.4); and (3) a quantitative comparison against the source motion to determine whether the edited motion preserves unedited regions (Section 4.5).

Following prior work (Athanasiou et al., 2024), the standard MotionFix evaluation protocol is based on motion-to-motion retrieval. The generated motion is used as a query to retrieve the ground-truth

target motion from a set of candidates based on the MotionFix-TMR model (Athanasiou et al., 2024). We report Recall at top-k (R@k, $k \in \{1, 2, 3\}$, higher is better) and average rank (AvgR, lower is better) under two settings: Batch-32, where retrieval is performed within a randomly sampled batch of 32 motions, and Full TestSet, where retrieval is performed against all motions in the test set. We also report FID (Fréchet Inception Distance), computed as the Fréchet distance between the distributions of motion embeddings of the generated and ground-truth target motions following Li et al. (2025). FID measures the distributional realism of the edited motions.

Because every metric in the first evaluation protocol relies on a single evaluator, evaluating performance only on this model risks introducing evaluator bias—meaning that higher scores may simply reflect an artifact of overfitting to the evaluator's specific preference rather than editing fidelity. Following prior practice in image generation (Stein et al., 2023), employing multiple independent evaluators provides a more faithful assessment of model performance. We therefore rescore the same generated motions with four retrieval models trained independently of this benchmark: TMR (Petrovich et al., 2023), TMR++ (Bensabath et al., 2024), Guo et al. (Guo et al., 2022), and MotionColBert (Zhang et al., 2026a), verifying both the generalizability and robustness of our framework. Appendix B provides further details on these models and the evaluation protocol.

While the two evaluations above use retrieval metrics to assess overall motion quality, motion editing also requires executing the instruction while preserving regions of the source motion that should remain unchanged. To quantify this capability, we introduce the **Preservation Error** ($\mathrm{PresErr}$).We compute joint angles for the set of biomechanical degrees of freedom (DoF) used in SMD. These 26 joint-angle are divided into edited and preserved sets according to the difference between the ground-truth source and target motions.

Let $A^{\mathrm{src}}, A^{\mathrm{tgt}} \in \mathbb{R}^{T\times 26}$ denote the corresponding joint-angle arrays, and let $\delta(u, v) = u - v$ be the angular difference. A DoF j is assigned to the edited set $\mathcal{E}$ if the target differs from the source by more than a threshold τ at any frame; all remaining DoFs form the preservation set $\mathcal{P}$. Since τ controls this partition, we vary it from 5° to 20°:

$$\mathcal{E} = \Big\{ j \;:\; \max_{1\le i\le T} \big|\delta(A^{tgt}_{ij}, A^{src}_{ij})\big| \ge \tau \Big\}, \qquad \mathcal{P} = \{1, \dots, 26\} \setminus \mathcal{E}. \tag{8}$$

For a generated motion with joint angles A, $\mathrm{PresErr}$ measures how far the preserved DoFs deviate from the source motion, averaged over all frames and all DoF in $\mathcal{P}$:

$$\mathrm{PresErr} = \frac{1}{T|\mathcal{P}|} \sum_{i=1}^{T} \sum_{j\in\mathcal{P}} \big|\delta(A_{ij}, A^{src}_{ij})\big|. \tag{9}$$

4.3 Comparison on MotionFix

Table 1 compares our method with previous baselines on MotionFix based on the standard evaluation protocol (Athanasiou et al., 2024). Our approach achieves the best performance on every Batch-32 metric, reaching R@1 of 81.25 compared with the previous best, OmniME, at 77.29 ($+3.96$), and reducing AvgR from 1.79 to 1.56. Under the full test-set protocol, our model achieves an AvgR of 9.61, surpassing OmniME (13.06) by 3.45 points across the full ranking distribution. For top-k recall, our method slightly improves R@1 (32.21 vs. 32.02), matches R@3 (59.88), and is marginally lower only on R@2 (49.21 vs. 50.20). Our method also produces the most realistic edited motions, with an FID of 0.045 compared to 0.091 (OmniME), 0.097 (Cross-Axis), 0.110 (SimMotionEdit), and 0.167 (TMED).

4.4 Robustness across retrieval models

In addition to MotionFix-TMR, we evaluate our method using the four other retrieval models described in Section 4.2. As shown in Table 2, our method achieves the highest mean recall and the lowest average rank across all five retrieval models. The ranking of methods is consistent for every model, with SimMotionEdit below OmniME and OmniME below ours. This indicates that the comparison in Table 1 is not tied to a particular retrieval model. These results confirm that our method robustly and consistently outperforms the baselines. Appendix B provides details of the evaluators, the conversion procedure and its validation, and the full set of metrics.

Table 1: Performance on MotionFix. "–" means not reported. ↑ higher is better, ↓ lower is better.

	Batch-32				Full TestSet				
Method	R@1 ↑	R@2 ↑	R@3 ↑	AvgR ↓	R@1 ↑	R@2 ↑	R@3 ↑	AvgR ↓	FID ↓
Ground Truth	100.0	100.0	100.0	1.00	64.36	88.75	95.56	1.74	0.000
$\text{MDM}_{\text{ICLR'23}}$ (Tevet et al., 2022)	4.03	7.56	10.48	15.55	0.10	0.10	0.10	–	–
$\text{MDM-BP}_{\text{SIGA'24}}$ (Athanasiou et al., 2024)	39.10	50.09	54.84	6.46	8.69	14.71	18.36	180.99	–
$\text{TMED}_{\text{SIGA'24}}$ (Athanasiou et al., 2024)	62.90	76.51	83.06	2.71	14.51	21.72	28.73	56.63	0.167
$\text{MotionReFit}_{\text{CVPR'25}}$ (Jiang et al., 2025)	66.33	80.05	84.98	2.64	–	–	–	–	–
$\text{SimMotionEdit}_{\text{CVPR'25}}$ (Li et al., 2025)	70.62	82.92	88.12	2.38	25.49	39.33	49.21	23.49	0.110
$\text{Cross-Axis}_{\text{CVPR'26}}$ (Han & Kim, 2026)	74.38	88.54	92.08	1.92	29.45	45.26	54.55	16.40	0.097
$\text{OmniME}_{\text{CVPR'26}}$ (Shi et al., 2026)	77.29	88.54	91.88	1.79	32.02	**50.20**	**59.88**	13.06	0.091
Ours	**81.25**	**90.42**	**93.96**	**1.56**	**32.21**	49.21	**59.88**	**9.61**	**0.045**

Table 2: Retrieval metrics on five retrieval models. Mean R@k averages the six recalls of the two protocols. Bold marks the best editing method.

	Mean R@k ↑					AvgR, Full TestSet ↓				
Encoder	GT	Source	SimMotionEdit	OmniME	Ours	GT	Source	SimMotionEdit	OmniME	Ours
MotionFix-TMR	91.45	62.58	59.28	66.64	**67.82**	1.74	17.48	23.49	13.06	**9.61**
TMR	91.49	63.83	59.77	63.34	**66.98**	1.73	16.57	20.89	17.09	**12.40**
TMR++	91.34	58.73	53.78	57.22	**62.22**	1.74	19.44	30.26	22.34	**17.37**
Guo et al.	91.38	48.36	45.13	49.75	**50.50**	1.74	34.31	38.39	31.55	**24.87**
Motion-ColBert	91.30	54.47	47.47	51.31	**59.85**	1.74	33.30	38.65	27.76	**24.42**

4.5 Preservation of unedited regions

Table 3 reports the preservation error against the two strongest baselines. Across all thresholds, our method perturbs the DoFs not targeted by the instruction substantially less, with errors between 0.36° and 1.60°, compared with 1.53° to 5.58° for the baselines. The gap widens as τ grows, as the preservation set includes the DoFs that change slightly in the target motion. This reflects the locality of our SMD-based editing: lines that are not rewritten retain their original values, anchoring the corresponding joint-angle trajectories. In contrast, the diffusion-based baselines regenerate the whole motion sequence, so unedited regions are reconstructed rather than retained.

4.6 Ablations

Table 4 isolates the contribution of each training stage. The untrained base LLM (zero-shot) achieves competitive performance (e.g., Batch R@1 of 72.29 and TestSet R@1 of 27.18) primarily because it makes conservative modifications that preserve the original motion's overall kinematic plausibility (detail discussed in Appendix E). As shown in the 'overlap' column, zero-shot keeps 0.878 of the source SMD verbatim, far above the 0.239 of the ground-truth. Applying GRPO directly on the zero-shot model yields marginal improvement and the overlap almost unchanged (0.852). This indicates that without task-specific SFT, the base LLM struggles to execute meaningful structural edits on SMD. SFT is for aligning the model with the edit instructions on MotionFix, enabling it to perform suitable editing (0.255) comparable to the ground-truth edit level. While supervised fine-tuning (SFT) teaches the model to execute substantive edits, its reliance on textual next-token prediction leaves it oblivious to physical constraints. This numerical inaccuracy degrades overall performance. The GRPO stage closes this loop by optimizing the LLM using motion-space reward feedback. This RL stage lifts TestSet R@1 to 32.21 ($+10.23$ over SFT only) and Batch R@1 to 81.25 ($+21.04$). We also provided additional results using different GRPO designs and reward models in Appendix G to show our design are applicable for different reward model.

Why GRPO helps. As discussed in Section 3.3, GRPO raises numerical precision by moving the generation distribution towards the model's own best samples. To quantify this, we decode $G = 16$ rollouts for each of 330 held-out prompts and scoring every rollout against the target motion, the SFT model reaches a mean of 0.498 with an average within-group spread of 0.530: it can already produce good edits, but which one it produces depends on sampling. GRPO lifts the mean to 0.670 and narrows the spread to 0.147, which turns an occasional good rollout into the typical one. The

Table 3: Preservation error on MotionFix under different τ.

Method	$\tau = 5°$	$\tau = 10°$	$\tau = 15°$	$\tau = 20°$
SimMotionEdit (Li et al., 2025)	1.60	3.30	4.63	5.58
OmniME (Shi et al., 2026)	1.53	3.13	4.35	5.29
Ours	**0.36**	**0.82**	**1.24**	**1.60**

Table 4: Ablation on MotionFix. Zero-shot is the base Qwen2.5-7B-Instruct with no task-specific training. The last column is the character-level overlap between the edited and the source SMD.

	Batch-32				Full TestSet					
	R@1 ↑	R@2 ↑	R@3 ↑	AvgR ↓	R@1 ↑	R@2 ↑	R@3 ↑	AvgR ↓	Parse%	Overlap w/ source
Ground Truth	100.0	100.0	100.0	1.00	64.36	88.75	95.56	1.74	-	0.239
Zero-shot (base LLM)	72.29	83.96	89.38	2.14	27.18	45.24	54.17	19.14	99.70	0.878
SFT only	60.21	74.17	79.38	3.39	21.98	34.65	42.38	40.22	99.90	0.255
Zero-shot + GRPO	72.80	84.17	88.79	2.03	28.05	45.18	55.82	17.66	100.0	0.852
SFT + GRPO	**81.25**	**90.42**	**93.96**	**1.56**	**32.21**	**49.21**	**59.88**	**9.61**	100.0	0.348

zero-shot model sits at 0.564 with a spread 0.165, so its rollouts barely differ from each other, hard to generate high qulity editing SMD. Appendix E.1 gives the full analysis and the reward distributions.

The flow-matching model faithfully compiles SMD. The edited SMD is only meaningful if the decoder realizes the angles it specifies rather than re-authoring the motion, which we check in three ways (detailed in Appendix F). Passing the SMD of a ground-truth motion through the parse and the decoder reconstructs that motion to 5.29 cm MPJPE on HumanML3D, against 17.6 cm for the rule-based inversion, and is competitive with VAE-based frameworks that use skeletal motion (Table 11). Comparing the decoded joint angles against the values the SMD declares, over $103{,}456$ key frames, gives a mean error of $3.89°$ with 93.3% of key frames within $10°$. Perturbing a single joint angle in the SMD and decoding gives a mean edit response of 0.71 and a mean leakage of $2.99°$ into the joints that not perturbed, so a change written into SMD stays local to that joint.

5 Conclusion

We introduced a motion-editing framework that reformulates text-driven motion editing as SMD editing. Instead of regenerating the entire motion sequence from random noise, our approach converts skeletal motion into readable structured text and lets an LLM edit motion in its native language modality. This provides a new route to localized and interpretable motion editing: joints and temporal regions not specified by the instruction can be preserved in the SMD representation, while the effects of an edit can be inspected directly by comparing the source and edited SMD joint by joint and over time. To make this representation operational for motion editing, we introduced a flow-matching decoder that converts edited SMD back into natural skeletal motion, and a GRPO refinement stage that optimizes the LLM to improve numerical accuracy. Experiments on MotionFix show that the proposed pipeline outperforms prior methods across five independently trained retrieval models.

Limitations and future work. The current framework also opens several directions for future research. First, the rule-based SMD parser is intentionally compact and interpretable, but this also makes it mildly lossy; future work could explore richer structured descriptions that preserve more dynamic details while remaining human-readable. Second, although the flow-matching decoder learns to produce natural motion from data, it does not yet incorporate explicit physical constraints. Integrating contact, balance, joint-limit, or dynamics-aware constraints could further improve physical plausibility. Third, the present pipeline focuses on single-person motions and standard sequence lengths; extending it to multi-person interaction, object interaction, and longer motion sequences would broaden its applicability. Finally, GRPO training is currently limited by the computational cost of motion rollouts. More efficient reward estimation or selective RL data sampling may further improve performance while reducing training cost.

AI USE STATEMENT

In this work, we have used generative AI tools, specifically LLM-based coding and writing. They were used to literature surveys; to implement methods designed by the authors; to give feedback on the design of experiments and ablations; and to assist with translation and polish. We have not used generative AI tools to generate synthetic data sets. The remaining tasks with required disclosure are not applicable to this work.

Additionally, we also used generative AI tools for tasks whose disclosure is recommended. we used it for polish the paper for readability; searching the literature; suggesting the choice of baselines.

We have reviewed all AI-assisted work. AI-written code was read by the authors before it was used for reported experiment. AI-drafted text was checked by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work edits 3D skeleton sequences from public motion-capture benchmarks, MotionFix (Athanasiou et al., 2024) and HumanML3D (Guo et al., 2022), both of which are from AMASS (Mahmood et al., 2019) and are used here under their original terms. No new motion data was collected and no human subjects were involved.

REPRODUCIBILITY STATEMENT

The architecture and the training procedure of every model in our pipeline are given in the paper, and all hyperparameters we used are listed in Appendix H.2. Code is provided in the supplementary material.

REFERENCES

Nikos Athanasiou, Alpár Cseke, Markos Diomataris, Michael J Black, and Gül Varol. Motionfix: Text-driven 3d human motion editing. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

Léore Bensabath, Mathis Petrovich, and Gul Varol. A cross-dataset study for text-based 3d human motion retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1932–1940, 2024.

Ling-Hao Chen, Wenxun Dai, Xuan Ju, Shunlin Lu, and Lei Zhang. Motionclr: Motion generation and training-free editing via understanding attention mechanisms. 2024.

Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, and Gang Yu. Executing your commands via motion diffusion in latent space. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18000–18010. IEEE, 2023.

Setareh Cohan, Guy Tevet, Daniele Reda, Xue Bin Peng, and Michiel Van de Panne. Flexible motion in-betweening with diffusion models. In *ACM SIGGRAPH 2024 conference papers*, pp. 1–9, 2024.

Ginger Delmas, Philippe Weinzaepfel, Francesc Moreno-Noguer, and Grégory Rogez. Posefix: correcting 3d human poses with natural language. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14972–14982. IEEE, 2023.

Ke Fan, Shunlin Lu, Minyue Dai, Runyi Yu, Lixing Xiao, Zhiyang Dou, Junting Dong, Lizhuang Ma, and Jingbo Wang. Go to zero: Towards zero-shot motion generation with million-scale data. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13336–13348. IEEE, 2025.

Purvi Goel, Kuan-Chieh Wang, C Karen Liu, and Kayvon Fatahalian. Iterative motion editing with natural language. In *ACM SIGGRAPH 2024 Conference Papers*, pp. 1–9, 2024.

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)*, pp. 5142–5151. IEEE, 2022.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Gyojin Han and Junmo Kim. Cross-axis feature fusion with joint-wise motion difference prediction for text-based 3d human motion editing. *arXiv preprint arXiv:2606.01014*, 2026.

Yannan He, Garvita Tiwari, Xiaohan Zhang, Pankaj Bora, Tolga Birdal, Jan Eric Lenssen, and Gerard Pons-Moll. Molingo: Motion-language alignment for text-to-human motion generation, 2026. URL https://arxiv.org/abs/2512.13840.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Vincent Tao Hu, Wenzhe Yin, Pingchuan Ma, Yunlu Chen, Basura Fernando, Yuki M Asano, Efstratios Gavves, Pascal Mettes, Bjorn Ommer, and Cees GM Snoek. Motion flow matching for human motion synthesis and editing. *arXiv preprint arXiv:2312.08895*, 2023.

Yilei Hua, Beibei Jing, Ce Zheng, Hanyu Zhou, Yawei Luo, and Wei Yang. Unimoflow: Grounding instruction-driven 3d human motion editing in generation. *arXiv preprint arXiv:2608.09143*, 2026.

Yiming Huang, Weilin Wan, Yue Yang, Chris Callison-Burch, Mark Yatskar, and Lingjie Liu. Como: Controllable motion generation through language guided pose code editing. In *European Conference on Computer Vision*, pp. 180–196. Springer, 2024.

Nan Jiang, Hongjie Li, Ziye Yuan, Zimo He, Yixin Chen, Tengyu Liu, Yixin Zhu, and Siyuan Huang. Dynamic motion blending for versatile motion editing. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22735–22745. IEEE, 2025.

Nan Jiang, Yunhao Li, Lexi Pang, Zimo He, Siyuan Huang, and Yixin Zhu. Motionmaster: Generalizable text-driven motion generation and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30629–30639, 2026.

Heechang Kim, Gwanghyun Kim, and Se Young Chun. Lamogen: Laban movement-guided diffusion for text-to-motion generation. *arXiv preprint arXiv:2509.24469*, 2025.

Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of imagenet classes in fr\'echet inception distance. *arXiv preprint arXiv:2203.06026*, 2022.

Junlin Li, Xinhao Song, Siqi Wang, Haibin Huang, and Yili Zhao. A unified conditional flow for motion generation, editing, and intra-structural retargeting. *arXiv preprint arXiv:2604.13427*, 2026.

Zhengyuan Li, Kai Cheng, Anindita Ghosh, Uttaran Bhattacharya, Liangyan Gui, and Aniket Bera. Simmotionedit: Text-based human motion editing with motion similarity prediction. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27827–27837. IEEE, 2025.

Shaohui Lin, Zhenwu Shi, Jingyu Gong, Jiao Xie, Yu Zhou, Baochang Zhang, Lizhuang Ma, and Chia-Wen Lin. Spatial temporal synergy: Balancing change and invariance in text driven 3d human motion editing. *arXiv preprint arXiv:2608.16008*, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xiaoyang Liu, Yunyao Mao, Wengang Zhou, and Houqiang Li. Motionrl: Align text-to-motion generation to human preferences with multi-reward reinforcement learning. *arXiv preprint arXiv:2410.06513*, 2024.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

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

Zichong Meng, Yiming Xie, Xiaogang Peng, Zeyu Han, and Huaizu Jiang. Rethinking diffusion for text-driven human motion generation: Redundant representations, evaluation, and masked autoregression. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27859–27871. IEEE, 2025.

Runqi Ouyang, Haoyun Li, Zhenyuan Zhang, Xiaofeng Wang, Zheng Zhu, Guan Huang, and Xingang Wang. Motion-r1: Chain-of-thought reasoning and reinforcement learning for human motion generation. *arXiv e-prints*, pp. arXiv–2506, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Mathis Petrovich, Michael J Black, and Gül Varol. Tmr: Text-to-motion retrieval using contrastive 3d human motion synthesis. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9454–9463. IEEE, 2023.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022a.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):4713–4726, 2022b.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Zhenwu Shi, Jingyu Gong, Peiwei Wang, Xingzan Wang, Tianwen Qian, Wenxi Li, Yuan Fang, Jiao Xie, Lizhuang Ma, and Shaohui Lin. Omni-supervised motion editing: Balancing change and invariance through positive-negative learning. *arXiv preprint arXiv:2605.30969*, 2026.

George Stein, Jesse Cresswell, Rasa Hosseinzadeh, Yi Sui, Brendan Ross, Valentin Villecroze, Zhaoyan Liu, Anthony L Caterini, Eric Taylor, and Gabriel Loaiza-Ganem. Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models. *Advances in Neural Information Processing Systems*, 36:3732–3784, 2023.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Yuxin Wen, Qing Shuai, Di Kang, Jing Li, Cheng Wen, Yue Qian, Ningxin Jiao, Changhai Chen, Weijie Chen, Yiran Wang, et al. Hy-motion 1.0: Scaling flow matching models for text-to-motion generation. *arXiv preprint arXiv:2512.23464*, 2025.

Bizhu Wu, Jinheng Xie, Meidan Ding, Zhe Kong, Jianfeng Ren, Ruibin Bai, Rong Qu, and Linlin Shen. Finemotion: A dataset and benchmark with both spatial and temporal annotation for fine-grained motion generation and editing. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13837–13846. IEEE, 2025.

Bizhu Wu, Jinheng Xie, Wenting Chen, Zhe Kong, Jianfeng Ren, Linlin Shen, Ruibin Bai, and Rong Qu. Motionmerge: A multi-granular framework for human motion editing, reasoning, generation, and explanation. *arXiv preprint arXiv:2605.18956*, 2026.

Yujie Yang, Zhichao Zhang, Jiazhou Chen, and Zichao Wu. Partmotionedit: Fine-grained text-driven 3d human motion editing via part-level modulation. *arXiv preprint arXiv:2512.24200*, 2025.

Qing Yu, Akihisa Watanabe, and Kent Fujiwara. Causal motion diffusion models for autoregressive motion generation. *arXiv preprint arXiv:2602.22594*, 2026.

Jonas Zausinger, Lars Pennig, Anamarija Kozina, Sean Sdahl, Julian Sikora, Adrian Dendorfer, Timofey Kuznetsov, Mohamad Hagog, Nina Wiedemann, Kacper Chlodny, et al. Regress, don't guess–a regression-like loss on number tokens for language models. *arXiv preprint arXiv:2411.02083*, 2024.

Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Yong Zhang, Hongwei Zhao, Hongtao Lu, Xi Shen, and Ying Shan. Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14730–14740, 2023a.

Mingyuan Zhang, Huirong Li, Zhongang Cai, Jiawei Ren, Lei Yang, and Ziwei Liu. Finemogen: Fine-grained spatio-temporal motion generation and editing. *Advances in Neural Information Processing Systems*, 36:13981–13992, 2023b.

Pengfei Zhang, Pinxin Liu, Pablo Garrido, Hyeongwoo Kim, and Bindita Chaudhuri. Kinmo: Kinematic-aware human motion understanding and generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11187–11197. IEEE, 2025.

Yao Zhang, Zhuchenyang Liu, Yanlan He, Thomas Ploetz, and Yu Xiao. Fine-grained motion retrieval via joint-angle motion images and token-patch late interaction. *arXiv preprint arXiv:2603.09930*, 2026a.

Yao Zhang, Zhuchenyang Liu, Thomas Ploetz, and Yu Xiao. Encoder-free human motion understanding via structured motion descriptions. *arXiv preprint arXiv:2604.21668*, 2026b.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Bingfan Zhu, Biao Jiang, Sunyi Wang, Shixiang Tang, Tao Chen, Linjie Luo, Youyi Zheng, and Xin Chen. Motiongpt3: Human motion as a second modality. In *International Conference on Learning Representations*, volume 2026, pp. 12631–12657, 2026.

A SMD REPRESENTATION

We adopt the structured motion descriptions (SMD) introduced by Zhang et al. (2026b) as the text representation. A deterministic rule-based mapping translates continuous motion into a readable text format. It first computes 26 biomechanical degrees of freedom (DOFs) from 3D joint positions, grouped systematically by body part: pelvis tilt, list, and rotation; hip flexion, adduction, and rotation, alongside the knee and ankle for each leg; lumbar extension, bending, and rotation; shoulder flexion, adduction, and rotation, alongside the elbow for each arm; and finally, neck flexion and tilt. Each DOF is recorded as an angle in degrees, sampled at 20 frames per second. Every angular trajectory is subsequently smoothed, partitioned at its local extrema, and encoded as a sequence of discrete intervals. An interval explicitly logs the kinematic trend (increases, decreases, or holds), its starting and ending angles, and its exact time span. A stationary joint is written as a single constant value, whereas an active joint yields a detailed list of intervals. Accompanied by a header line documenting the duration and frame rate, the resulting plain text is both human-readable and editable.

Because the original SMD framework was designed for motion understanding (Zhang et al., 2026b), it abbreviates periodic joint movements using a repeat shorthand that only specifies the number of cycles and their angular range, so the angle during a cyclic pattern is compressed into a single line text. While this compact format is highly effective for comprehension tasks where only the broad angular boundaries matter, it is inherently lossy for motion generation and editing. Specifically, the shorthand discards the exact phase of each cycle and the precise timing of individual strides, rendering a conditioned generative model incapable of recovering a correctly phased, coordinated motion. To resolve this, we modify the representation by unrolling every cycle into standard increases and decreases intervals. This adaptation preserves exact phase alignment and inter-limb coordination, albeit at the cost of increased text length (3823 characters compared to 1646 characters for the motion depicted in Figure 4). This represents the only substantive modification we introduce, the underlying joint schema, angle definitions, and interval grammar remain identical to the prior work.

Figure 4 shows a complete SMD sequence for a walking motion under both version. By unfolding the periodic joint movements, the editing-specific SMD captures fine-grained temporal dynamics, ensuring a better fidelity in reconstructing natural motions.

B EVALUATING WITH MULTIPLE RETRIEVAL MODELS

This appendix provides the details about Section 4.4, describes the retrieval models, the conversion that feeds one motion to all of them, the checks that the conversion is correct, the reimplemented protocol, and the complete table.

B.1 WHY ONE ENCODER MAY NOT BE ENOUGH

The MotionFix protocol scores an edited motion by retrieval, and the metric asks how highly the target ranks among the candidates. Every number in Table 1 therefore may introduce the bias of the used retrieval model. Two motions are close if that encoder says they are close, and a method is better than another if that encoder prefers its output.

Other generative areas have also found this kind of problem. Kynkäänniemi et al. (2022) show that FID is driven to a large extent by how much ImageNet class content the Inception features happen to pick up, so that the score can be moved without changing perceptual quality. Stein et al. (2023) compare feature extractors across a wide range of generative models and report that the ranking of methods changes when Inception is replaced by DINOv2, and they recommend reporting more than one extractor.

MotionFix has one additional property that makes the question concrete. The dataset was built by mining AMASS for pairs of similar motions with the TMR encoder and keeping the top-2 neigh-

SMD for Motion Understanding

```
MOTION 3.8s @20fps (deg)
PelvisTilt: -5
PelvisList: decreases 8>1 [0.0-0.8]; increases 1>10 [0.8-1.4]; decreases 10>1 [1.4-2.0];
    increases 1>11 [2.0-2.6]; decreases 11>1 [2.6-3.1]; increases 1>8 [3.1-3.7]
PelvisRot: repeats 3x(-17..10) [0.0-3.7]
RHipFlex: repeats 3x(-11..30) [0.0-3.7]
RHipAdd: decreases -10>-16 [0.0-0.2]; increases -16>-5 [0.2-0.9]; decreases -5>-16 [0.9-1.4];
    increases -16>-4 [1.4-2.1]; decreases -4>-15 [2.1-2.6]; increases -15>-10 [2.6-2.9];
    increases -10>-5 [2.9-3.2]; decreases -5>-13 [3.2-3.7]
RHipRot: holds -10 [0.8-1.6]; holds -9 [1.8-2.8]; holds -11 [3.1-3.8]
RKnee: repeats 3x(14..59) [0.0-3.2]
RAnkle: repeats 3x(-16..-1) [0.0-3.7]
LHipFlex: repeats 3x(-13..31) [0.0-3.7]
LHipAdd: repeats 3x(-9..5) [0.0-3.2]
LHipRot: holds -11 [0.0-0.9]; decreases -8>-13 [0.9-1.1]; holds -11 [1.2-3.8]
LKnee: decreases 20>12 [0.0-0.6]; increases 12>64 [0.6-1.0]; decreases 64>17 [1.0-1.3];
    decreases 17>12 [1.3-1.7]; increases 12>63 [1.7-2.1]; decreases 63>16 [2.1-2.5]; increases
    16>64 [2.5-3.3]; decreases 64>16 [3.3-3.7]
LAnkle: repeats 3x(-17..3) [0.0-3.7]
LumbarExt: holds 9 [0.5-2.6]; holds 10 [2.8-3.8]
LumbarBend: -5
LumbarRot: repeats 3x(-25..19) [0.0-3.7]
RShldrFlex: repeats 3x(-34..16) [0.0-3.7]
RShldrAdd: repeats 3x(-18..2) [0.0-3.7]
RShldrRot: repeats 3x(-26..-8) [0.0-3.7]
RElbow: repeats 3x(33..53) [0.0-3.7]
LShldrFlex: repeats 3x(-44..20) [0.0-3.7]
LShldrAdd: repeats 3x(-16..5) [0.0-3.7]
LShldrRot: repeats 3x(10..27) [0.0-3.7]
LElbow: repeats 3x(31..58) [0.0-3.7]
NeckFlex: 48
NeckTilt: holds -9 [0.0-0.8]; holds -7 [1.1-1.9]; increases -9>-1 [1.9-2.4]; decreases -1>-12
    [2.4-3.8]; increases -12>-4 [3.0-3.7]
```

SMD for Motion Editing/Generation (ours)

```
MOTION 3.8s @20fps (deg)
PelvisTilt: -5
PelvisList: decreases 8>1 [0.0-0.8]; increases 1>10 [0.8-1.4]; decreases 10>1 [1.4-2.0];
    increases 1>11 [2.0-2.6]; decreases 11>1 [2.6-3.1]; increases 1>8 [3.1-3.7]
PelvisRot: increases -15>8 [0.0-0.7]; decreases 8>-17 [0.7-1.3]; increases -17>10 [1.3-1.9];
    decreases 10>-14 [1.9-2.5]; increases -14>6 [2.5-3.0]; decreases 6>-17 [3.0-3.7]
RHipFlex: increases -6>29 [0.0-0.7]; decreases 29>-8 [0.7-1.3]; increases -8>29 [1.3-1.9];
    decreases 29>-7 [1.9-2.5]; increases -7>30 [2.5-3.0]; decreases 30>-11 [3.0-3.7]
RHipAdd: decreases -10>-16 [0.0-0.2]; increases -16>-5 [0.2-0.9]; decreases -5>-16 [0.9-1.4];
    increases -16>-4 [1.4-2.1]; decreases -4>-15 [2.1-2.6]; increases -15>-10 [2.6-2.9];
    increases -10>-5 [2.9-3.2]; decreases -5>-13 [3.2-3.7]
RHipRot: holds -10 [0.8-1.6]; holds -9 [1.8-2.8]; holds -11 [3.1-3.8]
RKnee: increases 14>59 [0.0-0.4]; decreases 59>14 [0.4-0.8]; increases 14>59 [0.8-1.6];
    decreases 59>14 [1.6-1.9]; increases 14>58 [1.9-2.8]; decreases 58>16 [2.8-3.2]; holds 13
    [3.2-3.8]
RAnkle: decreases -2>-15 [0.0-0.3]; increases -15>-2 [0.3-1.2]; decreases -2>-16 [1.2-1.6];
    increases -16>-1 [1.6-2.4]; decreases -1>-16 [2.4-2.7]; increases -16>-2 [2.7-3.7]
LHipFlex: decreases 30>-13 [0.0-0.7]; increases -13>31 [0.7-1.2]; decreases 31>-12 [1.2-1.9];
    increases -12>31 [1.9-2.4]; decreases 31>-11 [2.4-3.0]; increases -11>30 [3.0-3.7]
LHipAdd: increases -5>5 [0.0-0.4]; decreases 5>-8 [0.4-0.8]; increases -8>4 [0.8-1.6]; decreases
    4>-9 [1.6-2.0]; increases -9>4 [2.0-2.7]; decreases 4>-8 [2.7-3.2]; holds -6 [3.2-3.8]
LHipRot: holds -11 [0.0-0.9]; decreases -8>-13 [0.9-1.1]; holds -11 [1.2-3.8]
LKnee: decreases 20>12 [0.0-0.6]; increases 12>64 [0.6-1.0]; decreases 64>17 [1.0-1.3];
    decreases 17>12 [1.3-1.7]; increases 12>63 [1.7-2.1]; decreases 63>16 [2.1-2.5]; increases
    16>64 [2.5-3.3]; decreases 64>16 [3.3-3.7]
LAnkle: increases -10>2 [0.0-0.7]; decreases 2>-13 [0.7-0.9]; increases -13>2 [0.9-1.8];
    decreases 2>-13 [1.8-2.1]; increases -13>3 [2.1-3.0]; decreases 3>-17 [3.0-3.7]
LumbarExt: holds 9 [0.5-2.6]; holds 10 [2.8-3.8]
LumbarBend: -5
LumbarRot: decreases 16>-25 [0.0-0.8]; increases -25>15 [0.8-1.2]; decreases 15>-25 [1.2-1.9];
    increases -25>18 [1.9-2.5]; decreases 18>-25 [2.5-3.0]; increases -25>19 [3.0-3.7]
RShldrFlex: decreases 13>-32 [0.0-0.8]; increases -32>14 [0.8-1.3]; decreases 14>-34 [1.3-1.9];
    increases -34>16 [1.9-2.5]; decreases 16>-34 [2.5-3.0]; increases -34>16 [3.0-3.7]
RShldrAdd: decreases -8>-18 [0.0-0.7]; increases -18>2 [0.7-1.4]; decreases 2>-9 [1.4-1.9];
    increases -9>2 [1.9-2.5]; decreases 2>-18 [2.5-3.8]; increases -18>2 [3.8-3.7]
RShldrRot: decreases -2>-22 [0.0-0.8]; increases -22>-8 [0.8-1.4]; decreases -8>-22 [1.4-1.9];
    increases -22>-3 [1.9-2.4]; decreases -3>-26 [2.4-3.0]; increases -26>-3 [3.0-3.7]
RElbow: decreases 50>33 [0.0-0.8]; increases 33>52 [0.8-1.3]; decreases 52>33 [1.3-1.8];
    increases 33>53 [1.8-2.5]; decreases 53>35 [2.5-3.2]; increases 35>53 [3.2-3.7]
LShldrFlex: increases -38>17 [0.0-0.7]; decreases 17>-41 [0.7-1.3]; increases -41>20 [1.3-1.9];
    decreases 20>-42 [1.9-2.5]; increases -42>18 [2.5-3.0]; decreases 18>-44 [3.0-3.7]
LShldrAdd: decreases 2>-13 [0.0-0.8]; increases -13>5 [0.8-1.4]; decreases 5>-14 [1.4-1.9];
    increases -14>3 [1.9-2.6]; decreases 3>-16 [2.6-3.8]; increases -16>2 [3.0-3.7]
LShldrRot: decreases 24>10 [0.0-0.9]; increases 10>26 [0.9-1.4]; decreases 26>10 [1.4-1.9];
    increases 10>27 [1.9-2.5]; decreases 27>11 [2.5-3.2]; increases 11>26 [3.2-3.7]
LElbow: increases 33>56 [0.0-0.8]; decreases 56>31 [0.8-1.4]; increases 31>58 [1.4-1.9];
    decreases 58>32 [1.9-2.4]; increases 32>57 [2.4-3.1]; decreases 57>31 [3.1-3.7]
NeckFlex: 48
NeckTilt: holds -9 [0.0-0.8]; holds -7 [1.1-1.9]; increases -9>-1 [1.9-2.4]; decreases -1>-12
    [2.4-3.0]; increases -12>-4 [3.0-3.7]
```

Figure 4: The full 26-DOF SMD representing the instruction "a person walks in place" under both versions. **Left:** The original SMD for understanding, which compresses periodic joint trajectories into a single repeat line (orange). **Right:** Our adapted version for motion editing and generation, which explicitly details every cycle (green) to strictly preserve the phase and temporal alignment of each stride.

bours of each source motion (Athanasiou et al., 2024). The source and the target of a triplet are therefore close in TMR space by construction, and the evaluation then measures distances in a space of the same family. A method that is good at moving a motion the way that particular encoder measures distance is rewarded twice. We do not propose a new metric here. We try to evaluate that if the same generated motions are scored by different retrieval models that were trained independently, does the comparison between methods still work?

B.2 THE FIVE ENCODERS

Table 5 lists the encoders. They differ in the motion representation used, in the training dataset. MotionFix-TMR is the official evaluator. The next two are the original TMR and its cross-dataset version, which share the ACTOR-style architecture of the evaluator but use the 263-dimensional HumanML3D features. The fourth is the text-motion matching model of Guo et al. (2022), a convolutional and recurrent encoder that is the standard feature extractor for text-to-motion generation and is architecturally unrelated to TMR. Motion-ColBert (Zhang et al., 2026a) is a newer text-motion retrieval model that uses joint angles rather than positions. All methods in this comparison including ours and the baselines, write their output in the 135-dimensional SMPL parameter format follow the MotionFix evaluator, we then convert them into the format that each retrieval model uses. For the three model use Guo features, the SMPL parameters are passed through the neutral SMPL-H body model to obtain 22 joint positions, the result is mapped into the HumanML3D coordinate frame, resampled from 30 to 20 frames per second, and turned into the 263-dimensional features with the reference implementation of Guo et al. (2022). For the Motion-ColBert, we calculate the 29 degrees of freedom joint angles based on their released code. We verified that the conversion is correct: the forward kinematics reproduces the same joint positions as MotionFix with a mean error of 0.00 mm over 50 validation motions.

B.3 RESULTS

Table 6 reports every metric of the three editing methods under all five encoders, together with the two reference rows. We had included "Ground Truth and "Source no edit" rows as references. The Ground Truth row direct use ground-truth, which is the upper bound of the protocol. It does not reach 100 on the Full TestSet because a few target motions of MotionFix appear more than once in the test set, and a motion that is identical to another one cannot be separated by any encoder. The

Table 5: The five motion encoders used as retrieval evaluators. Only the first one is the official MotionFix evaluator. The others were trained independently of the MotionFix benchmark.

Encoder	Architecture	Motion input	Training corpus	Dim
MotionFix-TMR (Athanasiou et al., 2024)	ACTOR transformer, 6 layers	SMPL params, 135-d, 30 fps	HumanML3D	256
TMR (Petrovich et al., 2023)	ACTOR transformer, 6 layers	Guo features, 263-d, 20 fps	HumanML3D	256
TMR++ (Bensabath et al., 2024)	ACTOR transformer, 6 layers	Guo features, 263-d, 20 fps	HumanML3D + KIT-ML + BABEL	256
Guo et al. (Guo et al., 2022)	Conv. + BiGRU	Guo features, 263-d, 20 fps	HumanML3D	512
Motion-ColBert (Zhang et al., 2026a)	ViT-B/16	Joint angles, 29-d, 20 fps	HumanML3D	256

second reference row is the source motion remains unchanged as output, which is what the protocol returns for a method that does nothing.

Our method has the highest Batch-32 R@1 of the three editing methods under all five encoders, and the lowest AvgR under both protocols and all five encoders. The ordering of SimMotionEdit below OmniME below ours holds under every one of the five. The conclusion of Section 4.3 does not depend on the evaluator.

One interesting thing is that doing nothing is a strong reference baseline. Keeping the source motion unchanged scores above SimMotionEdit in Batch-32 R@1 under all five encoders, above OmniME under TMR++ and Motion-ColBert, and above our own method in Full TestSet R@1 under the Guo encoder. We think this is a property of the benchmark. The source and the target of a MotionFix triplet were mined as near neighbors, so the source is already a good query for its own target. Evaluating improvements of a motion editing method should against prior baselines and source reference under multiple retrieval models.

Under the Guo encoder our method has the best Batch-32 numbers and the best AvgR under both protocols, and at the same time a lower Full TestSet R@1 and R@3 than OmniME. The two families of metric are two readings of the same vector of ranks, so this is not an inconsistency. Our median rank under that encoder is 7, the same as OmniME's, but the tail is much lighter: the 90th percentile of the rank is 52.5 against 83.0 and the 99th is 317 against 372, which is what moves AvgR from 31.55 to 24.87 while leaving the top of the ranking to OmniME. The Batch-32 protocol follows AvgR here rather than Full TestSet R@1 because with only 32 candidates a query that would land at rank 50 in the full gallery usually still lands in the top few of its batch. The behaviour is specific to this encoder because its space is far more concentrated than the others: the mean cosine between two different MotionFix motions is 0.71 in the Guo space against 0.06, 0.10 and 0.045 in the three TMR-style spaces, and the participation ratio of its embedding covariance is 3.4 of 512 dimensions against 29 to 44 of 256 for the others. Top-1 decisions in such a space are made on very small margins. This also implies that Guo et al. may not be an ideal retrieval model for measuring the similarity between two motions.

Our recommendation, in line with what Stein et al. (2023) recommend for image generation, is that the motion editing literature report the retrieval metrics under more than one encoder. It separates the improvements that survive a change of evaluator from the ones that do not.

C EDIT ERROR ON THE EDITED JOINTS

Section 4.2 splits the degrees of freedom into the edited set $\mathcal{E}$ and the preservation set $\mathcal{P}$ and reports the preservation error in the main paper. The counterpart on $\mathcal{E}$ is the edit error, which measures how close the generated motion is to the ground-truth target on the joints the instruction does concern:

$$\mathrm{EditErr} = \frac{1}{T|\mathcal{E}|} \sum_{i=1}^{T} \sum_{j \in \mathcal{E}} \left| \delta(A_{ij}, A_{ij}^{tgt}) \right|. \tag{10}$$

Table 7 reports it under the same four thresholds. Our method is closer to the target on the edited joints than both baselines throughout, by between 2.9° and 3.1°. We keep this metric out of the main paper because it treats the ground-truth target as the only correct result of the edit instruction. An instruction such as "raise the arms higher" admits many realizations that a human would accept, and the recorded target is one of them, so a large edit error does not by itself mean that the edit was wrong. The preservation error does not have this ambiguity, since the source motion is fixed

Table 6: MotionFix retrieval metrics with five retrieval models on the same generated motions. Rows within a block are directly comparable, columns across blocks are not, because each encoder spreads the candidate motions differently. Bold marks the best of the three editing methods within a block. ↑ higher is better, ↓ lower is better.

		Batch-32				Full TestSet			
Encoder	Method	R@1 ↑	R@2 ↑	R@3 ↑	AvgR ↓	R@1 ↑	R@2 ↑	R@3 ↑	AvgR ↓
MotionFix-TMR	Ground Truth	100.0	100.0	100.0	1.00	64.36	88.75	95.56	1.74
	Source, no edit	74.79	85.21	89.58	2.05	27.67	43.28	54.94	17.48
	SimMotionEdit	70.62	82.92	88.12	2.38	25.49	39.33	49.21	23.49
	OmniME	77.29	88.54	91.88	1.79	32.02	**50.20**	**59.88**	13.06
	Ours	**81.25**	**90.42**	**93.96**	**1.56**	**32.21**	49.21	**59.88**	**9.61**
TMR	Ground Truth	99.70	99.90	100.00	1.02	65.25	88.45	95.66	1.73
	Source, no edit	74.38	85.00	89.79	1.99	29.64	46.84	57.31	16.57
	SimMotionEdit	70.62	83.12	87.71	2.30	28.06	40.12	49.01	20.89
	OmniME	74.79	86.88	91.88	1.95	29.64	44.27	52.57	17.09
	Ours	**77.92**	**88.54**	**92.92**	**1.73**	**32.21**	**50.40**	**59.88**	**12.40**
TMR++	Ground Truth	99.70	100.00	100.00	1.02	64.46	88.65	95.26	1.74
	Source, no edit	70.00	83.33	88.75	2.14	24.51	37.55	48.22	19.44
	SimMotionEdit	62.50	76.67	83.33	2.86	21.94	35.38	42.89	30.26
	OmniME	65.83	81.67	87.92	2.39	25.49	36.76	45.65	22.34
	Ours	**71.88**	**83.75**	**90.83**	**2.01**	**30.04**	**43.87**	**52.96**	**17.37**
Guo et al.	Ground Truth	99.29	100.00	100.00	1.02	64.86	88.65	95.46	1.74
	Source, no edit	58.33	71.46	78.54	3.13	17.59	29.05	35.18	34.31
	SimMotionEdit	53.12	69.17	78.33	3.37	14.82	24.51	30.83	38.39
	OmniME	59.17	73.75	80.42	2.97	**18.77**	**28.46**	**37.94**	31.55
	Ours	**62.08**	**77.08**	**85.00**	**2.50**	16.40	27.47	34.98	**24.87**
Motion-ColBert	Ground Truth	99.70	99.90	100.00	1.02	64.26	88.85	95.06	1.74
	Source, no edit	65.21	75.21	81.04	3.08	22.92	36.36	46.05	33.30
	SimMotionEdit	57.92	70.21	76.67	3.42	16.60	27.67	35.77	38.65
	OmniME	60.83	74.17	83.12	2.68	19.57	31.62	38.54	27.76
	Ours	**70.62**	**81.25**	**86.46**	**2.51**	**26.48**	**42.29**	**51.98**	**24.42**

and a correct edit should keep the preservation joints unchanged. We therefore think that the edit error should serve as an indicative sanity check for edit correctness rather than the primary metric. The overall editing quality still should be evaluated via the retrieval performance across multiple evaluators as shown in Appendix B.

Table 7: Edit error on MotionFix, measured on the edited set $\mathcal{E}$ against the ground-truth target, for four values of the threshold τ. Lower is better.

Method	$\tau = 5°$	$\tau = 10°$	$\tau = 15°$	$\tau = 20°$
SimMotionEdit (Li et al., 2025)	14.14	15.34	16.76	18.37
OmniME (Shi et al., 2026)	13.92	15.10	16.63	18.14
Ours	**10.98**	**12.70**	**14.49**	**16.28**

D ANOTHER USE CASE: TEXT-TO-MOTION GENERATION

Because the LLM's only job is to write SMD, the same pipeline could also performs text-to-motion generation once we change the input from a source SMD plus an instruction to motion caption. The flow-matching model for Human Motion Generation is trained based on the 263-d HumanML3D representation, and the GRPO reward becomes the cosine similarity between the compiled motion and the caption based on TMR evaluator (Petrovich et al., 2023). The LLM, the SMD representation, the LoRA configuration, and the three training stage are unchanged.

Table 8: Text-to-motion generation on HumanML3D. R-precision, FID, and MM-Dist use the standard Guo evaluator. "–" = not reported.

Method	R@1 ↑	R@2 ↑	R@3 ↑	FID ↓	MM-Dist ↓	MModality ↑	CLIP-Score ↑
Real (GT)	0.511	0.703	0.797	0.002	2.974	–	–
$\text{MDM}_{\text{ICLR'23}}$ (Tevet et al., 2022)	0.455	0.645	0.749	0.489	3.330	2.290	0.481
$\text{MLD}_{\text{CVPR'23}}$ (Chen et al., 2023)	0.542	0.735	0.827	0.078	2.808	1.676	0.645
$\text{T2M-GPT}_{\text{CVPR'23}}$ (Zhang et al., 2023a)	0.492	0.679	0.775	0.141	3.121	1.831	0.607
$\text{MoMask}_{\text{CVPR'24}}$ (Guo et al., 2024)	0.521	0.713	0.807	**0.045**	2.958	1.241	0.637
$\text{MARDM}_{\text{CVPR'25}}$ (Meng et al., 2025)	0.517	0.708	0.805	0.116	2.968	1.923	0.642
$\text{CMDM}_{\text{CVPR'26}}$ (Yu et al., 2026)	**0.576**	0.768	0.853	0.079	2.647	1.951	**0.675**
$\text{MotionGPT3}_{\text{ICLR'26}}$ (Zhu et al., 2026)	0.553	0.747	0.837	0.208	2.725	1.018	–
$\text{LaMoGen}_{\text{CVPR'26}}$ (Kim et al., 2025)	0.513	0.704	0.813	0.206	2.993	0.973	–
$\text{MoLingo}_{\text{CVPR'26}}$ (He et al., 2026)	0.528	0.721	0.815	0.049	–	1.414	0.672
Ours (SMD, SFT→GRPO)	0.574	**0.790**	**0.883**	0.134	**2.399**	1.266	0.669

We evaluate on HumanML3D (Guo et al., 2022) with the standard Guo evaluation protocol (Guo et al., 2022).As reinforcement learning training requires a significant amount of resources, we only use 8k caption-motion pairs for RL training, correspond to 2679 unique motions (11% of the HumanML3D training set). As Table 8 shows, the model reaches FID 0.134 and R-Precision@1/2/3 0.574/0.790/0.883 with MM-Dist 2.399, leading on R@2, R@3, and MM-Dist, remaining competitive on FID with generation-specific methods, and offering a readable intermediate representation. This framework is therefore not confined to editing.

E Additional ablation analysis

E.1 Why GRPO helps

As discussed in Section 3.3, GRPO improves the numerical precision of the LLM's SMD outputs by moving its generation distribution toward the high-similarity end. To quantify this, we decode $G = 16$ rollouts for each of 330 held-out prompts through the flow-matching model and measure the cosine similarity of each sample to the target motion. Alongside the mean reward, we also report the mean within-group spread, the gap between the best and the worst of the 16 rollouts from the same prompt, to compare the output distribution across different training stages.

As shown in Table 4, the zero-shot(base LLM) edits the source SMD only slightly, so its rollouts barely differ from each other, and its within-group spread is relatively small (Fig. 5). Its mean of 0.564 is not evidence of good editing: a source motion already close to its target far more than an unrelated motion, so most of that similarity is from source motion. SFT moves the distribution the other way, yielding a broad spread with a mean of 0.498 and an average within-group spread of 0.530. This indicates an inherent capacity for high-quality edits in the SFT-only model, but the output quality is highly dependent on sampling stochasticity. By exploiting this within-group variance, GRPO moves the distribution toward its upper bound: the mean rises to 0.670, while the within-group spread drops to 0.147. GRPO therefore concentrates the output distribution, turning an occasional good rollout into the typical one.



Figure 5: The reward distribution of $G = 16$ rollouts for each training stage

E.2 MOTION GENERATION ABLATION

Complementing the editing ablation in Section 4.6, we also report the ablation results for text-to-motion generation here. The training stages affect generation as they do editing: SFT alone imparts the SMD format but lacks numerical accuracy, whereas GRPO bridges this gap. Table 9 isolates these effects on HumanML3D while holding the LLM, the SMD representation, and the frozen flow-matching model fixed.

The zero-shot (base LLM) baseline is excluded from this generation ablation analysis. For editing tasks, the prompt includes the source SMD, providing a structural template that the base LLM can modify. For generation tasks, the model need generate the entire SMD sequence from scratch based solely on a caption. An untrained base LLM cannot produce the specialized SMD format without any prior examples, making zero-shot generation unfeasible.

Table 9: Generation ablation on HumanML3D (Guo protocol (Guo et al., 2022)). Parse% is the fraction of generated SMD that the rule parser accepts.

	FID ↓	R@1 ↑	R@2 ↑	R@3 ↑	MM-Dist ↓	Parse%
SFT only	0.587	0.424	0.627	0.738	3.214	100.0
SFT + GRPO	**0.134**	**0.574**	**0.790**	**0.883**	**2.399**	99.96

E.3 WHAT EACH TRAINING STAGE CHANGES IN THE SMD

The retrieval numbers say which stage scores better, not what each stage learned to do. Because SMD is text, we can read that off directly: for every test motion we measure the character-level overlap between the edited SMD and the source SMD, and calibrate it against the ground-truth target, which is what a correct edit looks like. A method whose overlap sits far above the target's has barely touched the source; one that matches the target has rewritten as much as the edit demands, whether or not it rewrote the right things.

Table 10: Character-level overlap with the source SMD on the MotionFix test set, measured as the "difflib" sequence-matching ratio. The ground-truth target row is the reference: a value far above it means the source SMD was largely untouched.

	Overlap with source	Median
Ground-truth target	0.239	0.224
Zero-shot (base LLM)	0.878	0.972
SFT only	0.255	0.239
SFT + GRPO	0.348	0.328

The zero-shot model keeps 87.8% of the source verbatim, and its median of 0.972 means that for more than half the test set it returns the source almost unchanged. Its retrieval score therefore reflects the fact that a source motion already close to its target far more than an unrelated motion. SFT moves the overlap to 0.255, similar to the GT target's 0.239: SFT teaches the model how much of the SMD an edit is supposed to rewrite, but the SFT-only model doesn't know whether the rewrite is correct. GRPO keeps that willingness to edit and adds the numerical accuracy. The overlap with the source is 0.348, so the editing is still more conservative than the ground truth.

F FLOW-MATCHING MODEL FIDELITY EXPERIMENTS

As mentioned in Section 4.6, our framework relies on the frozen flow-matching model faithfully compiles the joint angles specified by the LLM, rather than re-authoring the motion. This appendix details the three experiments summarized in Section 4.6, followed by the ablation studies that motivate our architectural design.

F.1 RECONSTRUCTION FIDELITY

A straightforward approach to mapping SMD back to skeletal motion is a purely rule-based reconstruction. This method parses the text into a dense $(T, 26)$ joint angle array and computes the final motion via forward kinematics. While strictly faithful to the declared angles, the resulting motion is kinematically stiff and unnatural because it relies on simple linear interpolation across the temporal intervals. Consequently, it yields a full round-trip MPJPE of 17.60 cm on the HumanML3D test set (Table 11). To overcome this problem, our proposed flow-matching model utilizes this rule-parsed $(T, 26)$ array the condition, learning to recover realistic joint angles and generate natural, continuous motion.

Figure 6 visualizes this difference. For the motion "raise the arms overhead", the rule-based condition only interpolates the raising, holding, and lowering phases. In contrast, the flow-matching model restores the natural plateau where the arms remain elevated. The decoded motion also tracks the ground truth more closely than its input condition. The flow-matching model recovered the fine-grained dynamics omitted by the text while preserving the declared angles in SMD. As discussed in our limitations, this data-driven motion recovery method inherently lacks rigid physical constraints. Therefore, it approximates the target motion by projecting input conditions onto a learned natural motion manifold to recover the motion rather than enforcing strict physical constraints.

Table 11 compares our flow-matching decoder against recent motion autoencoders. A fundamental difference in the task: the baseline methods reconstruct motion sequences by directly encoding the ground-truth motion then decoding for reconstruction (a standard autoencoding objective). Our model is forced to reconstruct the motion from the SMD representation. Despite originating from this compressed representation, our approach achieves an MPJPE of 5.29 cm. This performance is competitive with dedicated discrete tokenizers that directly based on the ground-truth motion (e.g., 4.19 cm for MotionMillion and 6.33 cm for ScaMo), and it outperforms the 17.60 cm error of the rule-based baseline. Reaching an autoencoder-level reconstruction error validates that the motions decoded by our flow-matching model remain natural. Additionally, MPJPE measures the positional error of all joints between generated and GT motion. The fact that our flow-matching model achieves 5.29 cm MPJPE indicates that it recovers plausible global translation from the SMD joint angles. We hypothesize that the model deduces this global translation by jointly analyzing the overall body orientation alongside the angle of the hip and knee joints.

Table 11: Reconstruction MPJPE on HumanML3D.

Method	MPJPE (cm) ↓	Reconstructs from
Rule-based (parse SMD + forward kinematics)	17.60	SMD text
ScaMo FSQ tokenizer (Fan et al., 2025)	6.33	GT motion
MotionMillion FSQ tokenizer (Fan et al., 2025)	4.19	GT motion
Flow-matching model (ours)	**5.29**	SMD text

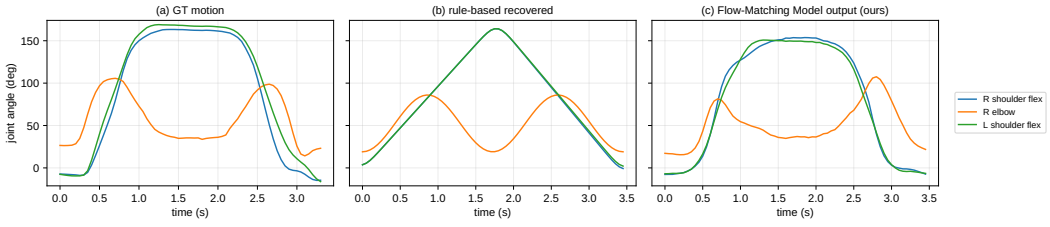


Figure 6: Joint-angle trajectories for "a person raises their arms high above their head." **(a)** ground-truth motion. **(b)** the rule-based SMD reconstruction. **(c)** The output motion from our flow-matching model. The model naturalises the specified angles rather than re-authoring the whole motion.

F.2 ANGLE ROUND-TRIP FIDELITY

To check whether the angle from SMD written by LLM survives after decoding, we re-extract joint angles from the decoded motion and compare them, at the key frames SMD specifies, to the values it

declares. Over 300 motions the mean realised-versus-declared error is $3.89°$ with a median of $2.47°$, and 77.2% of key frames fall within $5°$ while 93.3% fall within $10°$. The rule-based condition fed to the model already differs from the declared angles by $2.15°$, since the parse discretises to whole degrees, so the model itself contributes about $1.7°$ of execution error while naturalising the motion. It realises the motion the LLM wrote rather than substituting one of its own, which is what makes an SMD edit produce the intended change.

F.3 Perturbation experiments

In this section, we test whether modifying a single numerical value in SMD alters the corresponding joint in the decoded motion while leaving the rest of the body untouched. We validate this localized control through perturbation experiments. Specifically, we take a source motion, perturb a single joint angle within its SMD by a known magnitude d, decode the sequence by frozen flow-matching model, and compare the angles before and after the edit.

Let A and A' be the decoded joint angles before and after the edit, over the T frames of the motion. For an edit of size d applied to joint j,

$$\text{Response} = \frac{1}{T}\sum_{k=1}^{T}\frac{A'_{kj} - A_{kj}}{d}, \qquad \text{Leakage} = \frac{1}{T}\sum_{j'\neq j}\sum_{k=1}^{T}\left|A'_{kj'} - A_{kj'}\right|, \tag{11}$$

where the response is ideally 1 and the leakage ideally $0°$. Response indicates the realized magnitude of the applied perturbation on the targeted joint, while leakage represents the change of other joint angles where in the SMD remains unchanged. Across our evaluation on 100 motions, the mean response reaches 0.71, and the mean leakage is $2.99°$. Figure 7 illustrates a qualitative example: lower the right shoulder flexion in the SMD shifts the corresponding right shoulder in the decoded motion, while other joints adhere to their original position.

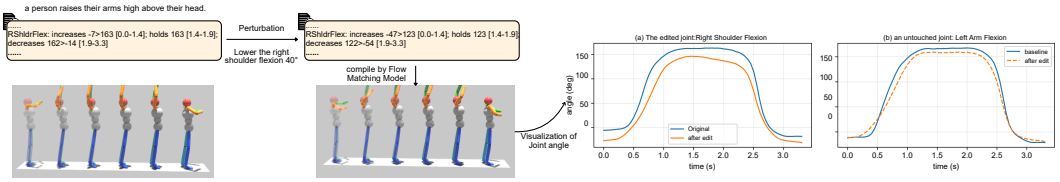


Figure 7: Perturbation experiments. Starting from one motion, we change a single joint angle (Right Shoulder Flexion) in its SMD and decode. Only the targeted joint's trajectory shifts, and other joints stays on its original path with a leakage of about $3°$.

Because the flow-matching model is learned from real motion rather than directly applying the requested angle, we also investigate its behavior beyond reasonable edit magnitudes. we change the edit magnitude from $5°$ to $180°$ across 10 motions and 4 different joints (Table 12). Alongside these single-joint perturbations, we also test contradictory edits by editing two connected joints in opposite directions to observe how the model handles unnatural SMD conditions.

Single-joint edits behave well over a wide band. The response is around 0.70 from $5°$ to $140°$, which confirms that normal perturbations are executed reliably. Only at $180°$, where the request inverts the joint, the response drops to 0.36. Because our flow-matching model is trained on real human motion, such extreme perturbation make the resulting motion are out of distribution, so the model is difficult to execute the implausible perturbation. Editing two joints of one arm in opposing directions exhibits a similar problem. With the shoulder raised by d and the elbow lowered by d, the model's responsiveness drops overall. When two conflicting demands cannot coexist on the natural motion manifold, the model struggles to produce the resulting unnatural motion.

In conclusion, The model is accurate on edits supported by real-world data but degrades on out-of-distribution or implausible motions. Therefore, the SMD should be viewed as a guiding request applied to a natural motion manifold, rather than a strict geometric constraint.

Table 12: Response and leakage against the requested magnitude. Left: a single joint, averaged over 10 motions and 4 joints. Right: contradictory edits by editing two connected joints in opposite directions. A negative response means the joint moved to the wrong direction.

	single joint		opposite-direction pair		
Requested (deg)	response	leakage (deg)	shoulder	elbow	leakage (deg)
5	0.65	0.15	0.60	0.85	0.31
20	0.65	0.60	0.61	0.91	1.49
40	0.65	1.27	0.58	0.73	2.47
80	0.73	3.32	0.58	0.35	4.11
100	0.74	4.49	0.67	0.22	5.31
140	0.67	5.40	0.43	-0.04	6.05
180	0.36	4.83	0.47	-0.20	5.74

F.4 Ablation experiments

Condition ablation. The flow-matching model is conditioned on the parsed numeric joint angles rather than the SMD string. To compare these two conditioning schemes, we train an identical model conditioned on the text embedding of SMD (we use CLIP (Radford et al., 2021) as text encoder), under the same backbone and the same data. As shown in Table 13, the round-trip error of SMD text-based model roughly doubles and a perturbation of a single joint value no longer produces the corresponding change. Its lower leakage follows from the same weakness, since a model that barely responds to the condition has little to leak.

We attribute this degradation of SMD text-based model to the fact that standard text encoders cannot accurately represent dense numerical values within their embedding spaces. Previous text-to-motion diffusion and flow models (Tevet et al., 2022; Chen et al., 2023) successfully condition directly on natural language captions because those inputs (e.g., "a person raises their arms high above their head") are dominated by high-level semantic information, which models like CLIP (Radford et al., 2021) are designed to encode.

Table 13: Conditioning ablation on HumanML3D.

Conditioning	Round-trip err (deg) $\downarrow$	Perturb response $\rightarrow 1$	Leakage (deg) $\downarrow$
SMD text	7.93	0.42	1.38
Parsed numeric angles (ours)	**3.89**	**0.71**	2.99

Sampling-step ablation. The flow-matching model is a DiT, and it is integrated at inference with an Euler solver. Table 14 varies the number of Euler steps, feeding GT-derived SMD through the frozen model so that the numbers isolate it from the LLM. FID drops from 0.045 at 2 steps to 0.020 by 20 steps and does not improve further, R-precision is flat throughout and matches the ground-truth. We use 20 as the default Euler solver steps.

Table 14: Euler-step ablation on HumanML3D.

Euler steps	FID $\downarrow$	R@1 $\uparrow$	R@3 $\uparrow$
2	0.045	0.522	0.821
5	0.029	0.514	0.814
10	0.022	0.516	0.816
20 (default)	0.020	0.524	0.817
50	0.020	0.515	0.820
100	0.020	0.519	0.814
Ground truth	–	0.511	0.797

Model structure ablation. We compare the effectiveness of different model architectures for our motion decoder. We train another motion decoder as a DDPM diffusion model (Ho et al., 2020) with the same DiT backbone, the same dataset and the same $(T, 26)$ joint angle conditioning. Only the training objective, which predicts noise instead of a rectified-flow velocity, and the sampler, which is DDIM instead of Euler. Table 15 compares the two model. The diffusion model could learn to turn SMD into motion, so the representation is not tied to a single generative framework, but it is less accurate (9.12 against 5.29 cm MPJPE) and slightly less smooth (acceleration 0.0078 against 0.0059. We think this discrepancy stems from how each model approaches the generation process. While diffusion models learn to reverse a stochastic noise process, flow-matching constructs deterministic, optimal-transport velocity fields. This formulation directly maps noise to the data manifold via straight paths, which is inherently better suited for capturing the continuous, fluid dynamics of natural human motion. Consequently, we adopt rectified flow for our final framework.

Table 15: Decoder architecture ablation (HumanML3D, same DiT backbone, dataset and SMD conditioning).

Model	MPJPE (cm) ↓	Acceleration ↓
DDPM diffusion	9.12	0.0078
Flow-matching (ours)	**5.29**	**0.0059**

G GRPO DESIGN

As discussed in Section 3.3, the RL stage compensates for the numerical imprecision of the SFT-only model. Before turning to RL we tried to fix the numbers inside SFT directly, by adding a number-token regression loss (Zausinger et al., 2024) that penalises numeric tokens in proportion to their value error rather than with plain cross-entropy. On editing this did not improve over plain SFT (TestSet R@1 21.96 against 21.98). The reason is the gap SFT already has. A token-level numeric loss still measures error in text space, while the quantity we care about is the motion those angles produce after a non-differentiable decoding step, and no token-level loss can propagate error through it. GRPO could solve this problem, because its reward is calculated in motion space. The rest of this section reports the choices and experiments we made in this stage.

G.1 REWARD FUNCTION

The reward we used is a single cosine similarity in embedding space. For generation, it is $\cos(\mathcal{E}(\mathcal{C}(\text{SMD}_{\text{gen}})), \mathcal{E}_{\text{text}}(c))$, and for editing it is $\cos(e_{\text{gen}}, e_{\text{tgt}})$. We compared two alternatives on editing.

Write e_{gen}, e_{tgt} and e_{src} for the encoder embeddings of the generated, target and source motions, and let $\mathcal{B}$ be the other targets in the same batch. The contrastive variant keeps the cosine to the target and subtracts the similarity to those in-batch negatives, so a motion is rewarded for being nearer to its own target than to anything else in the batch,

$$r_{\text{contrast}} = \cos(e_{\text{gen}}, e_{\text{tgt}}) - \lambda \max_{e^- \in \mathcal{B}} \cos(e_{\text{gen}}, e^-). \tag{12}$$

The in-gallery variant replaces the raw similarity with where the true target lands when the generated motion is used as a retrieval query against a gallery $\mathcal{G}$ that contains it, which rewards the ordering the evaluation actually measures rather than the absolute cosine,

$$r_{\text{gallery}} = 1 - \frac{\text{rank}\big(e_{\text{tgt}} \mid e_{\text{gen}}, \mathcal{G}\big) - 1}{|\mathcal{G}| - 1}, \qquad \text{rank} \in \{1, \ldots, |\mathcal{G}|\}, \tag{13}$$

so the reward is 1 when the target is retrieved first and 0 when it is retrieved last.

Under a same training setup all three reward functions get almost same results (Table 16), so the added complexity does not earn better performance and we keep the single cosine similarity as reward function.

Table 16: Reward-function variants on MotionFix editing.

Reward function	TestSet R@1 ↑	Batch R@1 ↑
Plain cosine (ours)	32.21	81.25
cosine similarity + contrastive (in-batch negatives)	32.17	80.76
cosine similarity + In-gallery (retrieval-rank reward)	32.31	81.02

G.2 Reward model

In our framework, the reward function for motion editing is defined as the cosine similarity between the encoded embeddings of the generated motion and the target motion. Since TMR shares the same evaluation protocol as MotionFix, to rule out concerns about directly optimizing the evaluation embeddings, we reproduce the reward using an independent Guo encoder (Guo et al., 2022). We report the results in Table 17.

As shown in Table 17, the framework achieves strong performance using either reward model, proving that our RL paradigm is robust to the specific choice of evaluator. However, the TMR encoder yields better results across the metrics. We attribute this advantage to the underlying capacity of the models. Because the TMR model is a more advanced and robust retrieval model than the Guo encoder, it extracts richer and more representative motion embeddings. This also translates into a more accurate cosine similarity signal, which provides the LLM with a more precise reward direction to adjust its output distribution, then guiding the generation of more natural and instruction-aligned motions.

Table 17: Reward-model comparison.

Task	Reward model	Batch-32 R@3 ↑	TestSet R@3 ↑	FID ↓
Editing	Guo encoder	91.40	58.61	0.050
Editing	TMR encoder	93.96	59.88	0.045
Task	**Reward model**	**R@3 ↑**	**FID ↓**	**MM-Dist ↓**
Generation	Guo encoder	0.875	0.135	2.457
Generation	TMR encoder	0.883	0.134	2.399

G.3 Scaling with RL data.

Since the group sampling in reinforcement learning requires substantial resources, this section investigates the impact of varying RL data size on performance. Figure 8 shows the validation reward during GRPO training for three RL data budgets: 1024, 2048, and the full 5388 MotionFix training set. All three subsets surpass the SFT baseline, confirming that GRPO is effective even with limited data. The full-data run does not saturate within our training budget, indicating that scaling up the RL data or developing more effective subset selection strategies could push editing quality further. This is also relevant from a computational perspective: the GRPO rollout cost is a common bottleneck of GRPO-based RL methods (Appendix I), so sample-efficient data selection is a promising direction for reducing RL training cost. The same cost is why we do not ablate the group size G: every additional candidate has to be decoded through the flow-matching model and scored, so ablation on G would multiply an already expensive stage. This is also why most of current research about GRPO (Liu et al., 2024) generally does not involve ablation on G. Following the analysis of GRPO in Shao et al. (2024), we expect a larger G to be preferable, since the advantage is normalized within the group and a wider group samples the model's output distribution more thoroughly, which is what the update needs in order to tell a good rollout from a poor one.

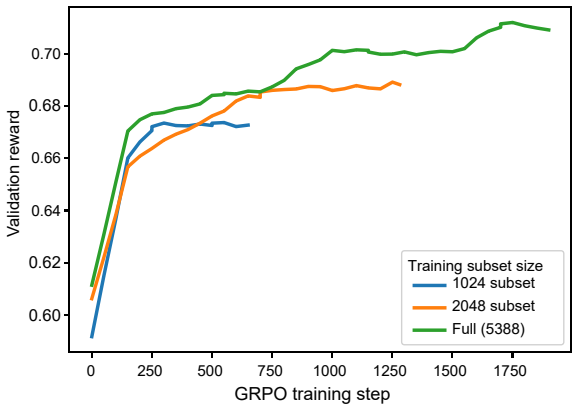


Figure 8: Validation set reward during GRPO training on three data subsets of MotionFix.

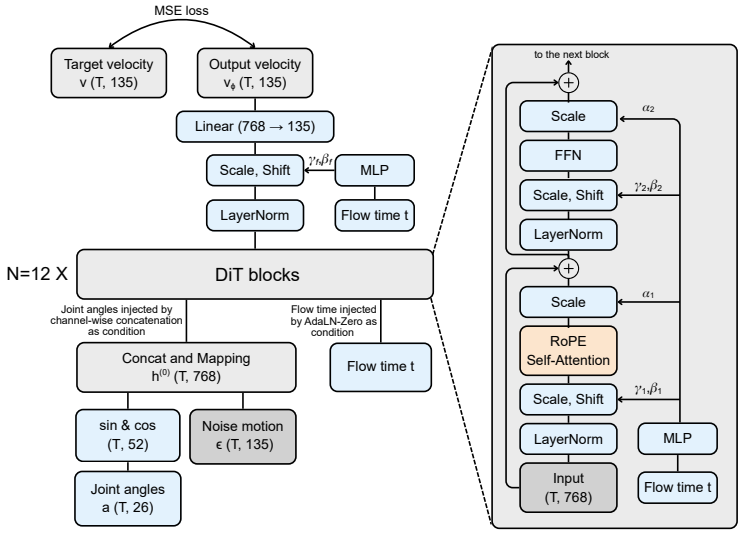


Figure 9: The flow-matching model and how it is trained. **Left:** the parsed joint angles a are sine&cosine encoded and concatenated channel-wise onto the noised motion, and the flow time t enters every block through AdaLN-Zero. The stack of $N = 12$ DiT blocks is followed by a LayerNorm and a linear map to a velocity, which is matched against the target velocity $v = m - \epsilon$ by a mean-squared error. **Right:** one DiT block.

H Implementation Details

H.1 Flow-matching model architecture

This subsection details the architecture of the flow-matching model and the procedures used to train it and to sample from it. The notation follows Figure 9.

H.1.1 Model Design

The first step is to parse the joint angles from the SMD produced by the LLM with the rule-based parser f_{parse}, which turns an SMD string into a dense angle array $a = f_{parse}(\hat{s}) \in \mathbb{R}^{T\times 26}$, one row per frame at the frame rate of the motion. Each angle is mapped to its sine and cosine,

$$c = \left[\sin(a) \,\|\, \cos(a)\right] \in \mathbb{R}^{T\times 52}, \tag{14}$$

which removes the discontinuity a joint would otherwise cross at $\pm 180°$. The condition is then joined to the noised motion $z_t \in \mathbb{R}^{T\times 135}$ by channel-wise concatenation, frame by frame, and

projected to 768 dimensions:

$$h^{(0)} = \big[z_t \,\|\, c\big] W_{in} + b_{in}, \qquad W_{in} \in \mathbb{R}^{187\times d}, \quad d = 768. \tag{15}$$

Condition and state share the same T, so no resampling or attention-based alignment is needed. We inject the joint angles as a condition by concatenation because this is the standard route for a condition that is registered to the output sample by sample, established for image-to-image diffusion by SR3 and Palette (Saharia et al., 2022b;a) and used in motion by CondMDI (Cohan et al., 2024). Concatenation writes the known frame-to-frame correspondence into the architecture, whereas cross-attention would discard it and ask the model to rediscover it from data.

The flow-matching model contains a stack of $N = 12$ DiT blocks (Peebles & Xie, 2023). Writing $\tau = \mathrm{MLP}(t)$ for the embedded flow time, block ℓ computes

$$(\gamma_1, \beta_1, \alpha_1, \gamma_2, \beta_2, \alpha_2) = \mathrm{MLP}(\tau), \tag{16}$$

$$u = h^{(\ell)} + \alpha_1 \odot \mathrm{Attn}_{\mathrm{RoPE}}\big(\gamma_1 \odot \mathrm{LN}(h^{(\ell)}) + \beta_1\big), \tag{17}$$

$$h^{(\ell+1)} = u + \alpha_2 \odot \mathrm{FFN}\big(\gamma_2 \odot \mathrm{LN}(u) + \beta_2\big), \tag{18}$$

where both LayerNorms are parameter-free, so every scale and shift the block applies comes from the flow time. The velocity is then read out from the final hidden state by a LayerNorm, followed by a linear map to the motion dimension:

$$v_\phi(z_t, t, a) = \big(\gamma_f \odot \mathrm{LN}(h^{(N)}) + \beta_f\big) W_{out}, \qquad (\gamma_f, \beta_f) = \mathrm{MLP}_f(\tau), \quad W_{out} \in \mathbb{R}^{d\times 135}. \tag{19}$$

Setting α_1, α_2 and W_{out} to zero at initialisation is AdaLN-Zero (Peebles & Xie, 2023): every block starts as the identity and the read-out starts at zero, so early training does not destroy the condition it has just been given. Attention uses rotary position embeddings (Su et al., 2024) over the time axis, which encode relative frame offsets rather than absolute indices, so a segment such as "increases -7>163" is realised the same way wherever in the sequence it falls and the model transfers across motions of different length.

Note that the network outputs a velocity, not a motion. Each frame of the motion is written as a 135-dimensional SMPL-6D vector containing the root translation, the global orientation and the 21 body joint rotations, following MotionFix (Athanasiou et al., 2024).

H.1.2 Training

Training follows the rectified-flow construction (Lipman et al., 2022; Liu et al., 2022). Given a parsed angle condition a and the corresponding ground-truth motion $m \in \mathbb{R}^{T\times 135}$, we draw a noise sample $\epsilon \sim \mathcal{N}(0, I)$ and a flow time $t \sim \mathcal{U}(0, 1)$, and form the interpolated state

$$z_t = (1 - t)\,\epsilon + t\,m, \qquad v = m - \epsilon, \tag{20}$$

where v is the target velocity of Figure 9. Because the path between noise and data is a straight line, v does not depend on t: the same target applies everywhere along the path, which is what keeps the learned field close to straight. The network is asked to predict this velocity, and the objective is a mean-squared error over the valid frames

$$\mathcal{L}_{FM} = \mathbb{E}_{t,\,\epsilon}\left[\frac{1}{|\Omega| \cdot 135} \sum_{i\in\Omega} \big\| v_\phi(z_t, t, a)_i - v_i \big\|^2\right], \tag{21}$$

with Ω the set of frames that are not padding. Normalising by $|\Omega|$ rather than by the padded length keeps motions of different duration equally weighted within a batch. Each training example therefore contributes a single forward pass at one random t; the sampler of Section H.1.3 is not run during training.

Training pairs the rule-parsed SMD angles $a = f_{parse}(f_{smd}(M))$ with the corresponding ground-truth motion m from the MotionFix training split, so the condition the model sees at training time is produced by exactly the parser that will feed it at inference time and no train/test gap is introduced by the parsing step. The model uses $d = 768$, $N = 12$ blocks, 8 attention heads and a feed-forward width of 2048, and is trained on windows of 64 frames with batch size 64, learning rate 2×10^{-4}, for 800 epochs. After training it is frozen for the remaining stages.

H.1.3 INFERENCE

At inference the model is an ordinary differential equation to be integrated. Starting from pure noise $z_0 = \epsilon \sim \mathcal{N}(0, I)$, we integrate the learned velocity field with a K-step Euler solver,

$$z_{k+1} = z_k + \tfrac{1}{K} \cdot v_\phi(z_k, t_k, a), \qquad t_k = k/K, \qquad k = 0, \ldots, K-1, \tag{22}$$

and take $z_K \in \mathbb{R}^{T \times 135}$ as the generated motion. The angle condition a is held fixed across all K steps; only the state and the flow time advance. We use $K = 20$ by default. Because the rectified-flow path is straight the field is nearly constant along it, so few steps suffice.

H.2 HYPERPARAMETERS AND TRAINING DETAILS

Stage 1, supervised fine-tuning. We adapt Qwen2.5-7B-Instruct with LoRA of rank 16 and $\alpha = 32$, dropout 0.05, on the query, key, value and output projections and on the gate, up and down projections of the feed-forward blocks; the base weights stay frozen. Training runs for 8 epochs over the 5388 MotionFix triplets at learning rate 10^{-4} with a cosine schedule, a warmup ratio of 0.03 and no weight decay. A single sequence is processed at a time with gradient accumulation over 16 steps, giving an effective batch of 16, in bfloat16 with a context window of 12288 tokens, which is long enough to hold a source SMD, an instruction and a target SMD without truncation. The stage takes about six hours on one H200, which is 0.8 h per epoch (Appendix I).

Stage 2, the flow-matching model. The velocity field is trained separately and stays frozen afterwards, so it is never updated by either LLM stage. It is trained for 800 epochs on 64-frame windows at batch size 64 and learning rate 2×10^{-4}, with the architecture of Appendix H.1, and reconstruction on the validation split is checked every 20 epochs.

Stage 3, GRPO. We use EasyR1/verl (Sheng et al., 2024) as the GRPO environment. Each step draws 16 prompts and samples $G = 32$ candidates per prompt at temperature 1.0 and top-p 1.0. Prompts are capped at 4096 tokens and responses at 4608. Only the LoRA weights are updated, with AdamW at learning rate 10^{-4}, weight decay 10^{-2}, no warmup and gradient clipping at 1.0; the flow-matching model and the reward encoder are frozen throughout. The KL penalty to the SFT reference uses the low-variance estimator with $\beta = 10^{-2}$. Memory is kept in range with FSDP full sharding, gradient checkpointing, and parameter and optimizer offload, since the policy, the reference policy, the frozen decoder and the frozen encoder are resident at once.

Validation and checkpoint selection. Validation runs before training and then every 50 steps on the validation split, and selecting checkpoint based on validation average reward.

I EFFICIENCY ANALYSIS

Table 18 summarizes the training and inference cost. Training is the expensive stage. Supervised fine-tuning (SFT) of the LLM (Qwen2.5-7B with LoRA) takes about 0.8 h/epoch on a single H200, roughly six hours for the full eight epochs. During reinforcement learning, each GRPO step samples a group of G=32 SMD candidates per prompt, decodes all of them through the frozen flow-matching model, and scores them with the reward model. This rollout dominates the training cost, taking about half of the wall time, resulting in GRPO taking roughly 9.5 h/epoch on $2\times$H200. The rollout cost is the main efficiency bottleneck, so more sample-efficient RL (for example, DAPO-style dynamic sampling and rollout reuse) is the natural next optimization, we leave this to future work.

At inference, the pipeline is cheap and close to real time: the LLM writes an SMD in about 0.16–0.18 s on a single H200 with vLLM, so the entire HumanML3D test set (2656 motions) is generated in roughly 8 minutes at 12–15k tokens/s. The frozen flow-matching model adds only a small fixed cost of 0.02 s/motion for a 20-step Euler sampling, so the full pipeline inference is approximately 0.18 s/motion on HumanML3D.

Table 18: Compute cost. Inference is measured with vLLM on a single H200; GRPO is measured on 2×H200 with group size G=32.

Stage	Setting	Cost
SFT	1×H200, Qwen2.5-7B + LoRA	≈ 0.8 h/epoch
GRPO	2×H200, G=32	≈ 9.5 h/epoch
Inference (LLM writes SMD)	Qwen2.5-7B + LoRA, vLLM, 1 H200	$0.16 - 0.18$ s/motion
Inference (flow-matching model)	20-step Euler, DiT	0.02 s/motion
Inference (full pipeline)	tested on HumanML3D	≈ 0.18 s/motion

J General-ability retention after fine-tuning

Unlike works such as MotionGPT3 (Zhu et al., 2026) that require full parameter LLM training to achieve high-quality motion generation, transforming motion editing and generation into purely text-modality tasks enables our LLM to retain its general capabilities while integrating motion editing. We verify this by evaluating the base Qwen2.5-7B-Instruct and our two task models (SFT+GRPO merged) on common LLM evaluation benchmarks: MMLU (Wang et al., 2024) (knowledge and reasoning, 5-shot) and IFEval (Zhou et al., 2023) (instruction following), using the same chat template for all three. As shown in Table 19, enabling the LLM to generate or edit SMD successfully adds a new skill without overwriting the model's general ability.

Table 19: General-ability retention. The base LLM and our two task models on MMLU and IFEval.

Model	MMLU (5-shot) ↑	IFEval prompt-strict ↑	IFEval inst-strict ↑
Base Qwen2.5-7B-Instruct	0.688	0.725	0.803
Ours (editing)	0.694	0.693	0.776
Ours (generation)	0.699	0.702	0.772

K More Qualitative Results

K.1 Editing examples and the SMD they came from

Figure 10 presents three editing examples. For each, the source motion is shown alongside its SMD, the edited SMD, and the resulting edited motion. Comparing the two SMD columns shows which joints are modified and by how much.

In the first example ("move the opposite arm"), the source SMD shows active right-shoulder joints (RShldrFlex, RShldrAdd, RShldrRot) with multiple increases and decreases, while the left-shoulder joints remain nearly static. After editing, the right-shoulder joints change to static, and the left-shoulder joints show dynamic patterns. In the second example ("wave the right hand instead of just raising it"), the source shows a simple raise-hold-lower trajectory for RShldrFlex and a single-arc elbow flexion. The edited SMD shows rapid alternating increases and decreases, producing the wave motion. In the third example ("sit in a chair, not on the floor"), the edited SMD shows static chair-height values (RHipFlex $59°$, RKnee $79°$) while adding a dynamic RShldrFlex trajectory for the arm spread. This transparency is an advantage of our text-based editing framework and is not available in diffusion-based approaches where edits happen inside opaque latent spaces.

K.2 Qualitative comparison between our method and baselines

Figure 11 shows a qualitative comparison with SimMotionEdit (Li et al., 2025) on Motionfix dataset. The failures the baseline makes are of two kinds. On "keep on sitting on the floor instead of getting up" it reproduces the source motion. On "do it harder 4 times and sidestep left" it departs from both the source and the target, so the edit is attempted but lands somewhere else. Our outputs follow the instruction while leaving the rest of the body on its original path, which is what the SMD rewrite encodes directly: only the joints the instruction concerns are changed, and the remaining lines of the SMD are retained.

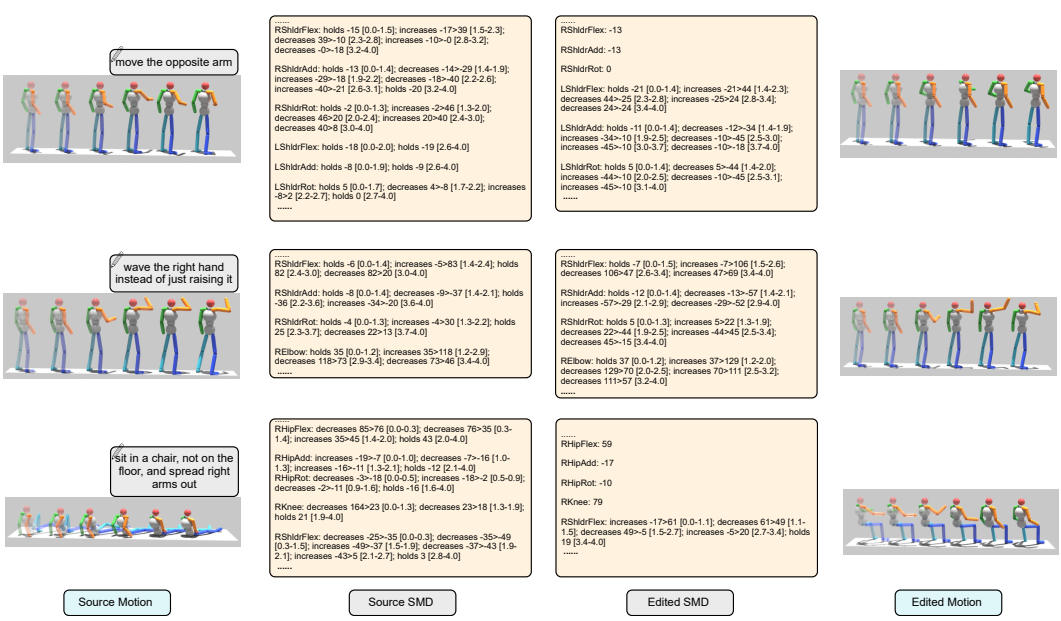


Figure 10: Qualitative editing examples. Each row shows the source motion, the source and edited SMD, and the edited motion. Comparing the two SMD columns shows which parts are modified.

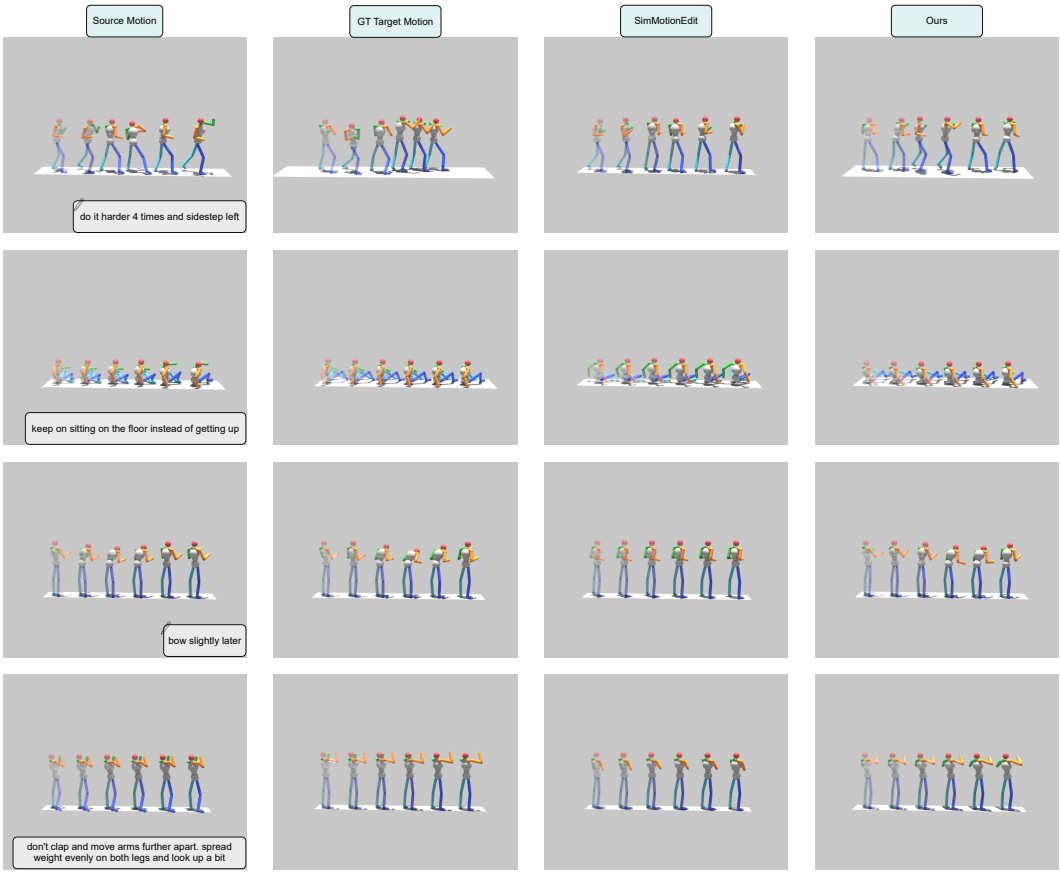


Figure 11: Qualitative comparison against SimMotionEdit on MotionFix. Each row shows the source motion with its edit instruction, the ground-truth target, the baseline output, and ours. Poses run from transparent to opaque with time.