ATM: Enhanced Alignment for Text-to-Motion Generation

Supplementary Material

Number	Inter-motion		Intra-motion	
	Top 1 ↑	FID ↓	Top 1 ↑	FID ↓
1	**0.434**	**0.354**	**0.434**	**0.354**
3	0.421	0.366	0.402	0.525
5	0.416	0.362	0.367	0.779

Table 6. Evaluations with varying numbers of hard negative examples for inter-motion alignment, and pairs of positive and negative examples for intra-motion alignment on KIT dataset.

Formulation		Top 1 ↑	FID ↓
$\hat{\boldsymbol{m}}_i^h$	$\arg\min \frac{\mathcal{D}(\hat{\boldsymbol{m}}_i, \hat{\boldsymbol{m}}_j)}{\mathcal{D}(\boldsymbol{m}_i, \boldsymbol{m}_j)}$	**0.519**	**0.320**
	$\arg\min \mathcal{D}(\hat{\boldsymbol{m}}_i, \hat{\boldsymbol{m}}_j)$	0.463	0.879
	$\arg\max \mathcal{D}(\boldsymbol{m}_i, \boldsymbol{m}_j)$	0.435	1.263
$\hat{\boldsymbol{c}}_{pos}$	$\arg\max \frac{\mathcal{D}(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}}_k)}{\mathcal{D}(\boldsymbol{c}, \boldsymbol{c}_k)}$	**0.519**	**0.320**
	$\arg\max \mathcal{D}(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}}_k)$	0.383	0.936
	$\arg\min \mathcal{D}(\boldsymbol{c}, \boldsymbol{c}_k)$	0.492	1.195
$\hat{\boldsymbol{c}}_{neg}$	$\arg\min \frac{\mathcal{D}(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}}_k)}{\mathcal{D}(\boldsymbol{c}, \boldsymbol{c}_k)}$	**0.519**	**0.320**
	$\arg\min \mathcal{D}(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}}_k)$	0.406	0.793
	$\arg\max \mathcal{D}(\boldsymbol{c}, \boldsymbol{c}_k)$	0.495	1.476

Table 7. Evaluation on the rules for identifying examples on HumanML3D dataset, by applying the corresponding formulation to compute $\hat{\boldsymbol{m}}_i^h$, $\hat{\boldsymbol{c}}_{pos}$, and $\hat{\boldsymbol{c}}_{neg}$ as defined in Eq (2), (5), and (6), respectively.

7. Ablation Study

Number of examples. In our method, we select one single misaligned example for each instance during inter-motion alignment using Eq. (2), and one pair of positive and negative proxies during intra-motion alignment using Eq. (5) and (6). Table 6 presents the results of varying the number of selected examples. Increasing the number of examples for training does not lead to performance improvement, underscoring the effectiveness and efficiency of our approach in selecting the most informative examples to enhance model training.

Rules for identifying examples. To validate the effectiveness of identifying semantically misaligned examples using Eq. (2), positive and negative proxies using Eq. (5) and (6), we conducted experiments with different variants of these formulations. As shown in Table 7, we compare each rule, formulated as a ratio, with two variants where only the numerator or denominator of the formulation is used. The results demonstrate that our ratio-based formulations significantly outperform the variants. This improvement is attributed to the ratio-based approach effectively identifying examples where the model fails to capture semantic similarity. By comparing the semantic similarity between pairs of generated and real motions, these formulations enhance the model's ability to address these discrepancies, thereby optimizing training performance.

Duration	Pre-taining	BL	BL+$\mathcal{L}_{inter}$	BL+$\mathcal{L}_{intra}$	ATM
One step	-	97.9ms	111.2ms	113.8ms	114.1ms
All steps	5.4h	10.9h	12.7h	12.8h	12.9h

Table 8. Training time cost on KIT. Training includes 400,000 steps. BL indicates the baseline trained solely with $\mathcal{L}_{mse}$.

Training time cost. The training time cost of our method is presented in Table 8. Pretraining the text-motion alignment embedding module takes 5.4 hours on the KIT dataset. Once pre-trained, this module can be integrated with various methods without requiring retraining. Compared to the baseline, additionally employing inter-motion or intra-motion alignment, or both, increases training time by over 10 ms per step due to the projection of generated motions into the joint embedding space. But all these approaches incur a similar additional training cost over the baseline by projecting motions in parallel, even though inter-motion alignment approach needs to process more motion clips than intra-motion alignment approach.

Hyper-parameters. *1) Weight factor.* Fig. 6 (1) illustrates how top-1 accuracy varies with the weight factor λ, which balances the inter-motion alignment loss $\mathcal{L}_{inter}$ and intra-motion alignment loss $\mathcal{L}_{intra}$ against the MSE loss $\mathcal{L}_{mse}$. The accuracy is sensitive to the value of λ, with the best performance achieved at $\lambda = 0.01$.

2) Batch size. Fig. 6 (2) shows that batch size impacts top-1 accuracy, partly by influencing the selection of examples for computing $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$. No further improvement is observed when the batch size exceeds 64.

3) Clip duration. Fig. 6 (3) examines the impact of clip duration on intra-motion alignment. Motion clips are adjusted by setting durations to 1, 1.5, 2, 2.5, and 3 seconds, corresponding to 20, 30, 40, 50, and 60 frames respectively, given the 20 FPS frame rate of the HumanML3D dataset. The motion clips are sequential and non-overlapping, with clips shorter than 1 second excluded. Accuracy varies with clip duration, reaching its peak at 2 seconds. Shorter clips may lack sufficient context, while longer clips may contain excessive actions, both of which can negatively impact clip-level semantic alignment.

4) Threshold. The threshold parameters ϵ and γ control example selection for inter-motion and intra-motion alignment, respectively. The curves in Fig. 7 show that the top-1 accuracies of the ATM model vary with ϵ and γ, but consistently peak at ϵ=0.85 and γ=0.8 on both HumanML3D and KIT datasets, indicating their robustness across datasets.

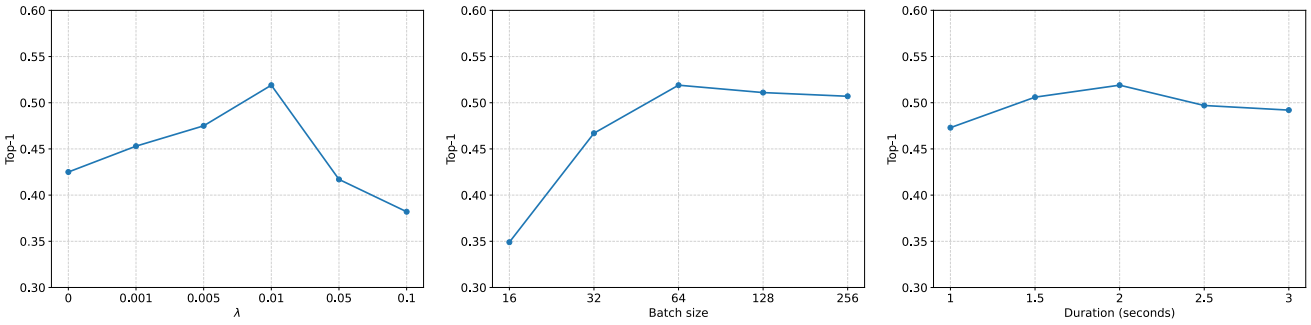

Figure 6. The changing curves of top-1 accuracy on the HumanML3D dataset with respect to the weight factor (λ), batch size, and clip duration (in seconds).

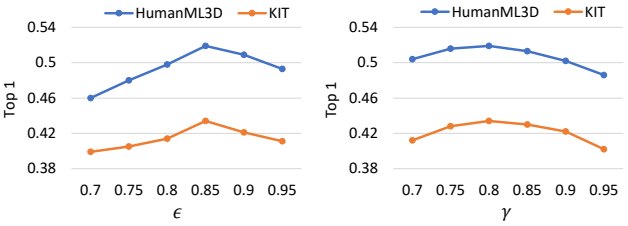

Figure 7. Evaluation of thresholds.

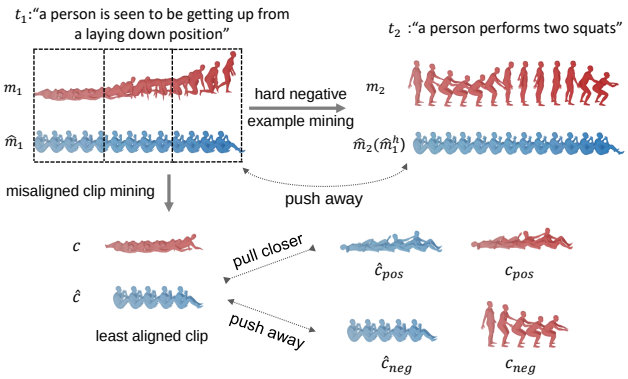

Figure 8. Examples of the identified hard negative example $\hat{\boldsymbol{m}}_1^h$ for the generated motion $\hat{\boldsymbol{m}}_1$, and pseudo-positive clip $\hat{\boldsymbol{c}}_{pos}$ and pseudo-negative clip $\hat{\boldsymbol{c}}_{neg}$ for the misaligned clip $\hat{\boldsymbol{c}}$. Blue and red indicate generated and ground truth motions, respectively.

8. Visual Explanation and Examples

How ATM addresses inter-motion and intra-motion misalignment? To intuitively illustrate the inter-motion and intra-motion semantic misalignment identified by our method, we present visualized examples in Fig. 8. For the generated motion sequence $\hat{\boldsymbol{m}}_1$, its inter-motion misaligned example $\hat{\boldsymbol{m}}_1^h$ is visually similar to it. However, the text descriptions $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$, along with the corresponding ground truth motions $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, exhibit very different semantics. These examples represent challenging cases where the model struggles to capture the semantic differences conveyed by the texts. Through inter-motion alignment, $\hat{\boldsymbol{m}}_1$ and $\hat{\boldsymbol{m}}_1^h$ are adaptively separated, prompting the model to better distinguish semantics and generate motions more closely aligned with the textual descriptions. Within $\hat{\boldsymbol{m}}_1$, the least aligned clip $\hat{\boldsymbol{c}}$ fails to reflect the semantics of "lying down", instead resembling "sitting down". Its negative proxy clip $\hat{\boldsymbol{c}}_{neg}$ shares similar semantics to $\hat{\boldsymbol{c}}$, yet the corresponding ground truth clips $\boldsymbol{c}_{neg}$ and $\boldsymbol{c}$ are semantically distinct. Conversely, the positive proxy clip $\hat{\boldsymbol{c}}_{pos}$ captures the semantics of "lying down". Intra-motion alignment optimizes $\hat{\boldsymbol{c}}$ by pushing it further from $\hat{\boldsymbol{c}}_{neg}$ and closer to $\hat{\boldsymbol{c}}_{pos}$, effectively transforming its semantics from "sitting down" to "lying down". Refining these local motions contributes to optimizing the entire motion sequence, ultimately enhancing its overall alignment.

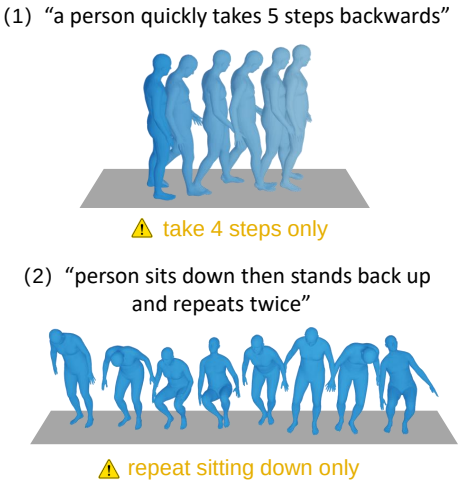

Figure 9. Failure cases generated by ATM. In (1), the darkening color highlights position change, while in (2), the unchanged color indicates remaining in the same position. Misaligned semantics are explained by yellow words.

Failure cases. Fig. 9 illustrates motion sequences generated by ATM that exhibit misalignment with the given texts. In (1), the generated motion includes "take 4 steps" instead of "take 5 steps", while in (2), the motion repeats "sit down" but fails to repeat "stand back up". These examples suggest that while our method effectively captures high-level textual

semantics related to verbs, such as "take steps backward" and "sit down and stand back up", it struggles with fine-grained details like numerical information, *e.g.*, "5 steps" and "twice". We infer that this limitation arises because our approach excludes highly similar text pairs from negative examples to prevent enforcing separation between their generated motions, such as "a person quickly takes 5 steps backwards" and "a person quickly takes steps backwards". This may lead the model to overlook subtle textual distinctions, resulting in misalignment at a finer granularity. The failure cases provided in the supplementary material attachment demonstrate that SOTA methods MoMask [9] and BAMM [31] also produce misaligned motions for these text descriptions. This indicates that these methods face challenges in addressing fine-grained alignment issues as well. To mitigate this issue, a potential solution is to emphasize these overlooked semantic concepts to improve the model's text comprehension ability.

9. Implementation Details

Text-motion alignment embedding module. Following previous works [23, 29], the text-motion alignment embedding module is a VAE-based architecture composed of a motion encoder, a text encoder and a motion decoder. It is pre-trained using text-motion pairs from the HumanML3D [8] or KIT [33] dataset by minimizing the training loss $\mathcal{L}_{align}$, defined as:

$$\mathcal{L}_{align} = \mathcal{L}_{REC} + \lambda_{KT} \cdot \mathcal{L}_{KT} + \lambda_{E} \cdot \mathcal{L}_{E} + \lambda_{NCE} \cdot \mathcal{L}_{NCE}, \tag{9}$$

where $\mathcal{L}_{REC}$ is a motion reconstruction loss designed to ensure accurate reconstruction of motion sequences, thereby enhancing the quality of motion representations. $\mathcal{L}_{KT}$ is a Kullback-Leibler (KL) divergence loss that aligns the latent distributions of text and motion. $\mathcal{L}_{E}$ is a cross-modal embedding similarity loss that brings text-motion pair embeddings closer in the feature space. $\mathcal{L}_{NCE}$ is an InfoNCE loss [27], which encourages positive pairs to be closer while pushing negative pairs apart. The weight factors $\lambda_{KT} = 1 \times 10^{-5}$, $\lambda_{E} = 1 \times 10^{-5}$ and $\lambda_{NCE} = 1 \times 10^{-1}$.

We exclude those negative examples with high textural similarities when computing $\mathcal{L}_{NCE}$ to prevent enforcing separation on motions with closely related semantics. Similar to the inter-motion alignment loss $\mathcal{L}_{inter}$, negative examples are filtered out if the cosine similarity between their corresponding text descriptions exceeds a threshold ϵ, where $\epsilon = 0.85$. The similarity is also computed by encoding the texts into embeddings using the pre-trained language model [35].

Text-to-motion generation model. We adopt the diffusion-based MDM [43] model as the default text-to-motion generation model, which has a transformer encoder-only architecture and a CLIP-based text encoder. Each human motion sequence is converted into a joint rotation representation, denoted as $\mathbb{R}^{V}$, where V represents the dimension of the rotation representation (263 for HumanML3D and 251 for KIT), and then input to the model. The model is trained with 1,000 noising steps, by the AdamW optimizer using a fixed learning rate of 1×10^{-4}.

Integration with other models. When integrating ATM into MLD [1], MotionDiffuse [57], ReMoDiffuse [58], and MoMask [9], we follow the implementation configurations provided in their official codebases. For continuous-representation-based T2M models such as MLD, MotionDiffuse, and ReMoDiffuse, we incorporate both $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$ into their original training objectives, as described in Eq. (8). In contrast, for VQ-VAE-based T2M models that operate on discrete motion tokens, such as MoMask, our alignment losses are incorporated into the motion reconstruction objective to refine the codebook, rather than being applied to the motion token prediction loss, as the mapping from discrete motion tokens to the text-motion embedding space is non-differentiable. Specifically, in MoMask, the VQ-VAE is first trained using the motion reconstruction loss:

$$\mathcal{L}_{rec} = \|\boldsymbol{m} - \hat{\boldsymbol{m}}\|_1, \tag{10}$$

where $\hat{\boldsymbol{m}}$ denotes the reconstructed motion sequence. When integrating ATM, $\mathcal{L}_{rec}$ is combined with $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$ following a similar composition to Eq. (8):

$$\mathcal{L}_{all} = \mathcal{L}_{rec} + \lambda \cdot (\mathcal{L}_{inter} + \mathcal{L}_{intra}). \tag{11}$$

The remaining training procedures after the VQ-VAE stage, such as training the masked transformer and residual transformer, remain unchanged.