Guided Motion Diffusion for Controllable Human Motion Synthesis **Appendix**

A. Analysis on $\mathbf{x}_{0,\theta}$ vs. ϵ_θ DPMs

In this section, we discuss the differences in behavior between the $\mathbf{x}_{0,\theta}$ and ϵ_θ models used to train DPMs. While both models are capable of generating high-quality samples, their denoising processes differ significantly. In Section 4.2, we previously claimed that the $\mathbf{x}_0$ predicting model maximizes its influence on the outcome $\mathbf{x}_{t-1}$ when $t \to 0$, whereas the ϵ predicting model maximizes its influence when $t \to T$. Based on this observation, we argue that the ϵ predicting model is more favorable than the $\mathbf{x}_0$ predicting model in circumstances where the outcome of the diffusion process will be altered by an external factor from the classifier.

To further understand the behavior of the two models, we examine Equation 1, which indicates that $\mathbf{x}_{t-1}$ is sampled from a Normal distribution with mean

$$\mu_t = \underbrace{\frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}}_{a}\mathbf{x}_0 + \underbrace{\frac{\sqrt{1-\beta_t}(1-\alpha_{t-1})}{1-\alpha_t}}_{b}\mathbf{x}_t \tag{1}$$

The coefficients a and b in μ_t modulate the contribution of the $\mathbf{x}_0$ model and the previous output $\mathbf{x}_t$. The larger the coefficient a is relative to b, the larger the contribution of the $\mathbf{x}_0$ model on the outcome of the denoising process.

In the case of an ϵ model, we substitute $\mathbf{x}_0$ based on the relationship $\mathbf{x}_0 = \frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon}{\sqrt{\alpha_t}}$ and get a different expression for μ_t as

$$\mu_t = \underbrace{\left(\frac{a}{\alpha_t} + b\right)}_{c}\mathbf{x}_t - \underbrace{\frac{a\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}}_{d}\epsilon \tag{2}$$

We can see that the contribution of the $\mathbf{x}_0$ model and the ϵ model are starkly different, with the ϵ model having a stronger contribution on μ_t, and hence $\mathbf{x}_{t-1}$, where t is large, while the opposite is true for the $\mathbf{x}_0$ model. In other words, an ϵ model is restricted to make a smaller change over time while an $\mathbf{x}_0$ model can still make a large change even at the very end of the diffusion process.

From the analysis above, we conclude that the choice of modeling ϵ or $\mathbf{x}_0$ is no longer arbitrary. Given the fact that the classifier guidance strength is modulated by Σ_t, which is smaller as $t \to 0$, and the fact that all DPM models are biased toward their training datasets, an $\mathbf{x}_0$ model capable of ever larger change as the guidance signal diminishes is not an ideal choice because it could easily overpower the guidance signal, especially at the end of the diffusion process, undoing all the guidance signal. Therefore, our GMD's trajectory model, which is subject to classifier guidance signals, is carefully chosen to be an ϵ model. We visualize the relative share over time of each model on μ_t in Figure A.1 and show the impact of the choice of model in Figure 3 in the main paper.

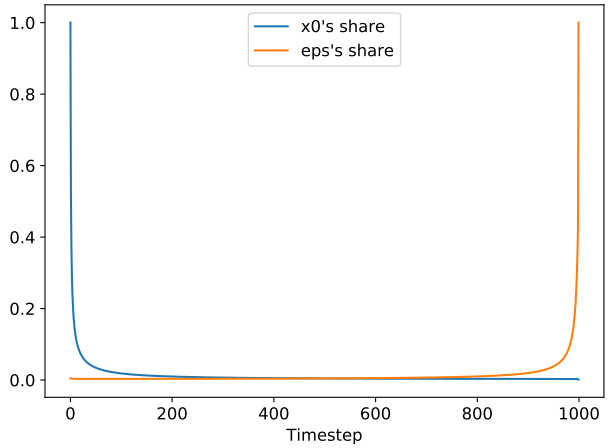

Figure A.1. Comparing $\mathbf{x}_0$ and ϵ contributions in the prediction of $\mathbf{x}_{t-1}$ based on the Cosine β_i scheduler.

A.1. Challenges of modeling ϵ in practice

In Section A, we discussed the benefits of modeling ϵ over $\mathbf{x}_0$ from the perspective of classifier guidance. However, there are fundamental differences and requirements for architectures that excel in predicting $\mathbf{x}_0$ versus ϵ. Specifically, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is independent and full-rank, meaning there is no smaller latent manifold that it resides in. On the other hand, $\mathbf{x}_0$ usually has a smaller latent manifold, which is the case for many real-world data including motions as most of the possible values in $\mathbf{x} \in \mathbb{R}^{263\times M}$ are not valid human motions, only a small subset of that is. Due to these differences, it requires special considerations for architectural design in models that successfully predict ϵ.

Although there is no sufficient reason to believe that modeling ϵ is fundamentally harder than modeling $\mathbf{x}_0$, in practice, modeling ϵ is restricted to cases where its shape is relatively small compared to the latent dimension of the denoising model. For example, when modeling $\epsilon \in \mathbb{R}^{263\times M}$ for a motion DPM, the original MDM architecture for ϵ prediction generated low-quality jagged motions compared to the same architecture for $\mathbf{x}_0$ prediction, which produced high-quality motions flawlessly. Increasing the latent dimension of MDM from 512 to 1,536 did not solve the problem entirely, indicating that predicting $\mathbf{x}_0$ and ϵ requires

different architectural designs that may not be satisfied by a single architecture. We argue that there require further studies on how to effectively design an ϵ predicting model.

However, when the space of ϵ is relatively small, such as in a trajectory DPM, the choice of architecture seems to matter less. The MDM transformer architecture was applied to trajectory modeling with relatively no problem. Ultimately, we used a convolution-based UNET with AdaGN [2] as the final architecture for our proposed method, as it demonstrated superior performance for both modeling trajectories and motions.

B. Relative vs. Absolute root representation

In this section, we discuss different ways of representing the root locations $\mathbf{z}$ of the motion. Generally, the root locations can be represented as absolute rotations and translations (**abs**) or relative rotations and translations compared to the previous frame (**rel**). In MDM [10], the root locations are represented with relative representation following the HumanML3D [3] dataset. In this case, the global locations at an exact frame i can be obtained by a cumulative summation of rotations and translations before i.

However, we observe that representing the root with absolute coordinates (**abs**) is more favorable than the relative one (**rel**) in two aspects: being more straightforward for imputation and easier to optimize. Therefore, we adopt the absolute root representation for our models.

In **rel**, a trajectory is described as velocity $\Delta\mathbf{z}^{(i)}/\Delta i$ in the local coordinate frame of the current pelvis rotation. This representation makes each $\mathbf{z}^{(i)}$ dependent on all previous motion steps in a non-linear relationship. Optimization becomes less stable as a small change in early motion steps may compound and become a larger change later on. Also, imputing specific values becomes ill-posed since there are many possible sets of values that are satisfiable.

On the other hand, for **abs**, the imputation and optimization of $\mathbf{z}$ become straightforward as they only involve replacing or updating $\mathbf{z}^{(i)}$ without dependency on other motion steps. We ablated the root representation by retraining MDM [10] and our model with both relative and absolute root representation, then show the results in Tab B.1. MDM shows a significant drop in performance when converted to the absolute representation, likely because the architecture is highly optimized for the relative representation, while for our models, the representation change results in a trade-off between the *FID* and *R-precision*.

Lastly, we note that the use of absolute root representation is necessary for our final model as the spatial guidance is done via a combination of imputation and optimization.

Table B.1. Text-to-motion evaluation on the HumanML3D [3] dataset. Comparision between relative and absolute root representation. The right arrow $\rightarrow$ means closer to real data is better.

	FID ↓	R-precision ↑ (Top-3)	Diversity →
Real	0.002	0.797	9.503
MDM [10] (**rel**)	0.556	0.608	**9.446**
MDM [10] (**abs**)	0.894	0.638	8.819
Ours (**rel**)	0.305	0.666	9.861
Ours (**abs**)	**0.212**	**0.670**	9.440
Ours $\mathbf{x}^{\text{proj}}$	0.235	0.652	9.726

C. Analysis on Emphasis projection

In this section, we discuss in greater detail our proposed Emphasis projection. Conceptually, we wish to increase the relative importance of the trajectory representation $\mathbf{z}$ within the motion representation $\mathbf{x}$. This could be done most simply by increasing the magnitude of those values of $\mathbf{z}$ by multiplying it with a constant $c > 1$. More precisely, let us assume the shape of x is $263 \times M$. A single motion frame $x = \mathbf{x}^{(i)}$ is a column vector of 263 scalars in which 3 elements (rot, x, z) are a column vector of a trajectory frame $z = \mathbf{z}^{(i)}$ that comprises root rotation and a ground location. The new trajectory elements become $z \times c$.

How to calculate a suitable scalar c?

By introducing a scalar $c > 1$, the trajectory elements z are given a higher relative importance than the remaining 260 elements in x. This relative importance is determined by the cumulative variance of the z elements compared to that of the remaining 260 elements. Assuming that all elements in x are independently and identically distributed according to a standard Normal distribution $\mathcal{N}(0, 1)$, we can represent the cumulative variance of the trajectory elements as

$$\text{Var}[x^{(\text{rot})} + x^{(\text{x})} + x^{(\text{z})}] = \sum_{j \in \text{Traj.}} \text{Var}[x^{(j)}] = 3 \qquad (3)$$

where $j \in$ Traj. refers to the indexes in x that are related to trajectory.

Similarly, we can represent the cumulative variance of the remaining 260 elements as $\text{Var}[\sum_{j \notin \text{Traj.}} x^{(j)}] = 260$, where $j \notin$ Traj. refers to the indexes in x that are not related to trajectory.

When we multiply trajectory by c, the new cumulative variance becomes $\text{Var}[c \times (x^{(\text{rot})} + x^{(\text{x})} + x^{(\text{z})})] = c^2 \sum_{j \in \text{Traj.}} \text{Var}[x^{(j)}] = 3c^2$. Therefore, the relative importance of the scaled trajectory elements compared to the remaining 260 elements in x is given by the expression

$$\frac{3c^2}{260 + 3c^2} \qquad (4)$$

Setting $c = \sqrt{\frac{260}{3}} \approx 9.3$ results in a relative importance of 50%, which strikes a reasonable balance between the trajectory and the rest of human motion. We have selected $c = 10$ as a rounded number of this fact, and it has been found to work well in practice.

Maintaining the uniform unit variance after scaling

After scaling up the trajectory elements by a factor of c, the variance of the new motion representation is no longer uniform. This presents a problem when trying to model it using the original DPM's β_t scheduler. In order to maintain uniform variance, we can redistribute the increased values from the trajectory part $c \times z$ to the rest in x via a random matrix projection.

There are two reasons why a random matrix projection is a good choice. First, it maintains the distance measure of the original space with high probability, meaning that the properties of the motion representation remain relatively unchanged. Second, a random matrix projection is easy to obtain and linear. It has an exact inverse projection, which ensures that there is no loss of information after the projection.

Finally, to maintain unit variance, we scale down the entire vector uniformly by a factor of $\frac{1}{263-3+3c^2}$.

C.1. Trajectory loss scaling

One approach to increase the emphasis on the trajectory part $\mathbf{z}^{(i)}$ of the motion $\mathbf{x}^{(i)}$ is to scale the reconstruction loss of only the trajectory part during the training of the motion DPM. This method does not change the representation but can potentially increase the model's emphasis on the trajectory part of the motion compared to the rest of the motion.

To compare the loss scaling method with the proposed Emphasis projection, we formulate a new loss function for a specific motion frame i, which increases the trajectory importance by a factor of k. This is given by the equation:

$$\mathcal{L}_k^{(i)} = \sum_{j \in \text{Traj.}} \left\| k\hat{x}^{(j)} - kx^{(j)} \right\|^2 + \sum_{j \notin \text{Traj.}} \left\| \hat{x}^{(j)} - x^{(j)} \right\|^2 \tag{5}$$

Here, $\hat{x} = \mathbf{x}_{0,\theta}(\mathbf{x}_t)^{(i)}$ represents the i-th motion frame of the DPM's prediction and $x = \mathbf{x}_0{}^{(i)}$ represents the i-th motion frame of the ground truth motion. The value of k multiplies inside the squared loss, resulting in k^2 times more importance on the trajectory part of the motion. For example, setting $k = 10$ would increase the importance of the trajectory part by 100-fold, which has the same scaling effect as setting $c = 10$ in Emphasis projection. Hence, the reasonable range of k is the same as that of c.

In the main text, we experimented with $k \in 1, 2, 5, 10$ and found that Emphasis projection consistently outperformed loss scaling regarding motion coherence.

D. GMD's Model Architecture

The trajectory and motion architectures of GMDare both based on UNET with Adaptive Group Normalization (AdaGN), which was originally proposed by [2] for class-conditional image generation tasks. However, we have adapted this model for sequential prediction tasks by using 1D convolutions. It should be noted that our architectures share some similarities with [5] with the addition of AdaGN. The architecture overview is depicted in Figure D.1 while the Adaptive Group Normalization is depicted in Figure D.2. The hyperparameter settings of the two DPMs are shown in Table D.1. We currently are in the process of open-sourcing the code base of GMD.

Convolution-based architectures are commonly used in state-of-the-art image-domain DPMs, such as those proposed by [8] and [9]. On the other hand, transformer-based architectures, which were used in the original MDM proposed by [10], are not well-studied architectures for DPMs [1, 7].

Our proposed architecture alone has led to a significant improvement in motion generation tasks, reducing the Fréchet Inception Distance (FID) by more than half compared to the original MDM (0.556 vs 0.212), as shown in Table 1 in the main paper.

Table D.1. Network architecture of our GMD's models based on the proposed 1D UNET with AdaGN.

Parameter	**Trajectory DPM**	**Motion DPM**
Batch size	512	64
Base channels	512	512
Channel multipliers	[0.125, 0.25, 0.5]	[2, 2, 2, 2]
Attention resolution	No attention	No attention
Samples trained	32M	
β scheduler	Consine [6]	
Learning rate	1e-4	
Optimizer	AdamW (wd = 1e-2)	
Training T	1000	
Diffusion loss	ϵ prediction	$\mathbf{x}_0$ prediction
Diffusion var.	Fixed small $\tilde{\beta}_t = \frac{1-\alpha_{t-1}}{1-\alpha_t}\beta_t$	
Model avg. beta	0.9999	

E. Training details

GMD's models. We used a batch size of 64 for motion models and a batch size of 512 for trajectory models. No dropout was used in all of the GMD's models: both trajectory and motion. We used AdamW with a learning rate of 0.0001 and weight decay of 0.01. We clipped the gradient norm to 1 which was found to increase training stability.

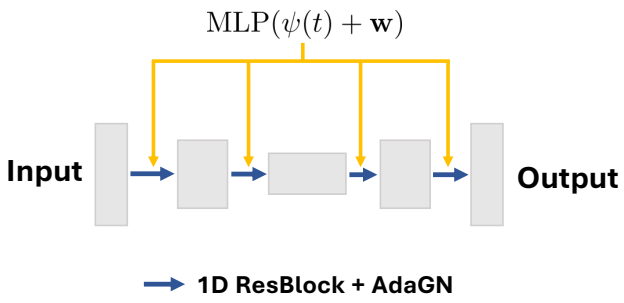

Figure D.1. A simplified overview of our GMD's 1D UNET + AdaGN architecture that is designed to process two input signals: the time step $\psi(t)$ and a text-prompt embedding $\mathbf{w}$. The time step is encoded using sinusoidal functions, while the text-prompt embedding is generated by the CLIP text encoder model, as described in [10]. The ResBlock + AdaGN component of the model is explained in Figure D.2.

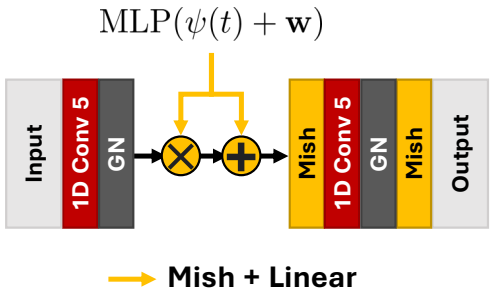

Figure D.2. A single 1D ResBlock with Adaptive Group normalization (AdaGN) [2]. The conditioning signal from the MLP, shared across all ResBlocks, is projected by first applying a Mish activation and then a resizing linear projection specific to each ResBlock. All kernel sizes are 5. We use Mish activation function following [5].

We used mixed precision during training and inference. We trained all motion models for 32,000,000 samples (equivalent to 500,000 iterations at batch size 64, and 62,500 iterations at batch size 512). We also employed the moving average of models during training ($\beta = 0.9999$) [4] and used the averaged model for better generation quality. Do note that our model architecture still improves over the baseline MDM without the moving average.

GMD's trajectory model. While the crucial trajectory elements are only the ground x-z locations, we have found it useful to train the trajectory model with all four components (rot, x, y, z). The additional (rot, y) seem to provide useful information that helps the model learn and reduce overfitting in the trajectory model. Note that the trajectory DPM is sensitive to the overfitting problem. Overtraining the model will result in a strong trajectory bias in the model making the model more resistant to classifier guidance and imputation. Our choice of training the trajectory model for 32,000,000 samples was carefully chosen based on this observation.

Retraining of MDM models. We retrained the original MDM using our absolute root representation and proposed Emphasis projection as the two main baselines. In order to maintain consistency, we kept the original optimization settings for the MDM models. Specifically, we used AdamW optimizer with a learning rate of 0.0001 and without weight decay. We found that gradient clipping of 1 provided more stability, so we also applied it here. We did not utilize mixed precision training for these models. To match the settings of the original MDM, we trained these models for 400,000 iterations at a batch size of 64.

F. Inferencing details

We have chosen the value of s as 100 for the classifier guidance strength. Our experiments have shown that this value of s performs well within the range of 100 to 200. For all our goal functions G_x, we always used the $p = 1$ norm. Whenever feasible, we implemented both imputation and classifier guidance concurrently. However, we ceased the guidance signals, i.e., classifier guidance and imputation, at $t = 20$ as this led to a slight improvement in the motion coherence.

Obstacle avoidance task. In this particular case, it was not feasible to create a point-to-point trajectory because doing so could potentially lead to a collision with an obstacle. As a result, we decided against utilizing any point-to-point trajectory imputation for this task.

References

[1] He Cao, Jianan Wang, Tianhe Ren, Xianbiao Qi, Yihao Chen, Yuan Yao, and Lei Zhang. Exploring vision transformers as diffusion learners. Dec. 2022. 3

[2] Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. May 2021. 2, 3, 4

[3] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5152–5161, 2022. 2

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. June 2020. 4

[5] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. May 2022. 3, 4

[6] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. Feb. 2021. 3

[7] William Peebles and Saining Xie. Scalable diffusion models with transformers. Dec. 2022. 3

[8] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution image synthesis with latent diffusion models. Dec. 2021. 3

[9] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image diffusion models with deep language understanding. May 2022. 3

[10] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. Sept. 2022. 2, 3, 4