Mimic-X: a large-scale motion dataset via fast physics-based controller adaptation Supplementary Material

Anonymous submission

1 Background of Option Framework

In reinforcement learning, the option framework (Sutton, Precup, and Singh 1999; Konidaris and Barto 2009) learns option policies that specialize in different sub-tasks, along with a meta-policy that arranges these options to solve more complex tasks. Similarly, humans possess the ability to perform a variety of movements and can organize these movements into sequences to accomplish complex tasks. Several works (Liu, Panne, and Yin 2016; Lee et al. 2019; Tao et al. 2023; Wang, Hodgins, and Won 2024) have manually defined option movements and designed transitions between these movements to successfully complete complex tasks. However, this approach requires expert knowledge to predefine the option movements. In contrast, Wang et al. (2020) cluster the motion space into multiple sub-spaces and employ a mixture-of-experts structure to model motions within these sub-spaces. However, their approach necessitates manual selection of the number of clusters, and it is challenging to scale to large motion datasets. In this paper, we evaluate a hierarchical clustering method and apply a modified option framework to construct a large-scale, physically accurate human motion dataset.

2 Kinematic Motion Extraction

Before applying our adaptive option framework, we need to extract 3D human motions from in-the-wild videos as reference motions. To achieve this, we utilize GVHMR (Shen et al. 2024) to recover human motions from videos, and then optimize the recovered motions by minimizing the projection error of keypoints to further improve the quality of motions. Our optimization includes three steps: local pose optimization, camera focal length optimization, and global translation optimization. In this section, we introduce the optimization pipeline for producing the reference motions.

Local Pose Optimization. The poses recovered by GVHMR are represented in SMPLX (Pavlakos et al. 2019) format, including non-root joint rotations θ, root rotations R^a, root translations R^p, and body shape parameters β. We optimize the joint rotations θ by minimizing the following objective function:

$$L_{local} = w_{2d}L_{2d} + w_{reg}L_{reg} + w_{smooth}L_{smooth} + w_{pen}L_{pen} \quad (1)$$

where L_{2d} is the projection error of the keypoints:

$$L_{2d} = \|\Pi(SMPLX(\hat{\theta}, R_c^a, R_c^p, \beta), f_g) - x_{2d}\|_1 \quad (2)$$

where R_c^a and R_c^p refer to the root rotations and root translations in the camera space, f_g is the predicted camera focal length by GVHMR, and $\Pi(.,.)$ refers to the projection function that project the 3D joints into image space. L_{reg} is the regularization loss. L_{reg} ensures the optimized joint rotations $\hat{\theta}$ remain close to the initialized values θ:

$$L_{reg} = \|\hat{\theta} - \theta\|_1 \quad (3)$$

L_{smooth} is the smooth term, preventing sudden changes during optimization. It can be formulated as:

$$L_{smooth} = \sum_t \|\hat{\theta}_t - \hat{\theta}_{t+1}\|_1 \quad (4)$$

where θ_t refers to the joint rotations at frame t.

As for L_{pen}, the penetration loss that penalizes mesh interpenetration of the SMPLX model, it is calculated using a collision penalizer (Pavlakos et al. 2019). However, the calculation is time-consuming due to the need to detect a large number of triangle collisions in each optimization iteration. To speed up the process, we detect triangle collisions every 100 steps, and utilize the detected collisions to calculate L_{pen} within those 100 steps.

In our experiments, the weights of different terms are: $w_{2d} = 6.0$, $w_{reg} = 0.1$, $w_{smooth} = 1.0$, and $w_{pen} = 0.2$, and the joint rotations $\hat{\theta}$ are optimized for 1,000 iterations.

Camera Focal Length Optimization. Since the global root translations recovered by GVHMR (Shen et al. 2024) are prone to error accumulation, it is necessary to optimize them in order to refine the quality of the motions. During this optimization, the 2D keypoints projected from the global motions must align with the input videos. However, reprojecting the global motions into the image space is infeasible due to the lack of accurate camera intrinsic parameters. To address this, we use a perspective camera and approximate the camera focal length f by minimizing the following objective function:

$$L_{focal} = \|(\hat{x}_{2d} - \hat{x}_{2d}^r) - (x_{2d} - x_{2d}^r)\|_1 \quad (5)$$

where $\hat{x}_{2d} = \Pi(SMPLX(\theta, R^a, R^p + \tilde{p}, \beta); \hat{f})$, x_{2d}^r refers to the coordinates of the root in 2D image space, and $\tilde{p} \in \mathbb{R}^3$

represents the global position offsets. These offsets are precomputed before each iteration by transforming the difference between the projected 2D positions of the global positions and the ground-truth 2D keypoints. The transformation is performed using the current optimized $\hat{f}$ to align the root projection with the ground truth.

Global Translation Optimization. To reduce the accumulated translation error of the recovered motions, we optimize the global translations by the following loss function:

$$L_{global} = w_{2d,r} L_{2d,r} + w_{smooth,r} L_{smooth,r} \quad (6)$$

where $L_{2d,r}$ is similar to L_{2d}, but instead of using root rotations and translations represented in the camera space, it uses those represented in the world space:

$$L_{2d,r} = \|\Pi(SMPLX(\theta, R^a, \hat{R}^p, \beta), \hat{f}) - x_{2d}\|_1 \quad (7)$$

where $\hat{f}$ is the camera focal length optimized by camera focal length optimization. $L_{smooth,r}$ is the smooth loss to penalize sudden changes of root translations between two adjacent frames $\hat{R}_t$ and $\hat{R}_{t+1}$:

$$L_{smooth,r} = \sum_t \|\hat{R}^p_t - \hat{R}^p_{t+1}\|_1 \quad (8)$$

In our experiments, we set $w_{2d,r} = 6$ and $w_{smooth,r} = 1.5$.

When extracting kinematic motions, we utilize all available videos from the AIST, IDEA400, and fitness datasets. The AIST dataset includes videos captured from 9 different view angles, while the IDEA400 and fitness dataset provide only monocular videos.

3 Details of Network Structure and Parameters

In this section, we report the detailed structure and parameters of the network in our experiments.

Auto-encoder Module. In our option policies, we first utilize an auto-encoder to map the input $\mathcal{X}_t$ into latent space for each cluster. The input $\mathcal{X}_t \in \mathbb{R}^{350}$ is comprised of the pose difference between current pose and pose $\mathcal{P}_{t+1} - \hat{\mathcal{P}}_t$, the reference pose $\mathcal{P}_{t+1}$ and body shape parameters β. The detailed structure and parameters of the auto-encoder are reported in Table 3.

Policy and Value Network. For each option policy in our framework, we employ an MLP-based policy network to generate the action a_t, and an MLP-based value network to produce the value scalar. Their detailed structure and parameters are reported in Table 4 and Table 5.

***w/o enc* and *Whole4*×**. As described in Section 4.1.3 of the main manuscript, we conduct ablation studies on the auto-encoder module *w/o enc* and the impact of larger model size *Whole4*×. The structures and parameters of policy network and value network for *w/o enc* are reported in Table 6 and Table 7, respectively; while structures and parameters of auto-encoder, policy network and value network for *Whole4*× are reported in Table 8, Table 9 and Table 10, respectively.

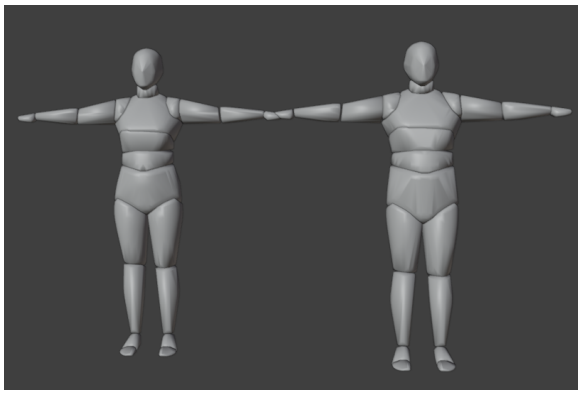

Figure 1: Character models used in the physics engine. The shape of our character models varies with the SMPLX shape parameters. Thus, more than two character models are used in our training.

4 Implementation Details

4.1 Character Models

In our experiments, we use the SMPLX model to represent the subjects in the datasets. For our Mimic-X dataset, we employ GVHMR to recover the SMPLX model parameters for each subject, while for the Human3.6M dataset, we obtain the parameters using the method in (Moon, Choi, and Lee 2022). After that, we can get a mesh with V vertices, J joints, and a skinning weight matrix $W \in \mathbb{R}^{V \times J}$ from the SMPLX model for each subject. Subsequently, we exploit the same method as Yuan et al. (2021) to generate character models used in the physics engine for each subject. Specifically, we first assign each vertex on the mesh to one joint with the maximum skinning weight. Next, for each joint, we compute the 3D convex hull of all the vertices assigned to it, creating the geometry actuated by that joint. This approach provides a more precise representation of the human model compared to the traditional method of using boxes and capsules. In our experiments, we remove the fingers in the SMPLX model since our dataset focuses on physical plausibility of human body. The character models used in our experiments are shown in Fig. 1

4.2 Dataset Implementation Detials

In this section, we present the training details of our adaptive option framework trained on Human3.6M, AIST, fitness, and IDEA400. In our paper, Human3.6M and subset of AIST are used to evaluate the quality of motion recovery achieved by our method, while the whole AIST, fitness, and IDEA400 are utilized to construct the Mimic-X dataset.

AIST. We cluster the AIST dataset into six clusters and pretrain six option policies, one for each cluster, to mimic the motions sampled from these clusters. Instead of using the entire cluster for pre-training, we sample 10% of motion sequences from each cluster, ensuring at least 100 sequences are included. During our initial experiments, we observed that our method struggled to mimic certain Jazz Bullet motions involving jumping. To address this issue, we created a

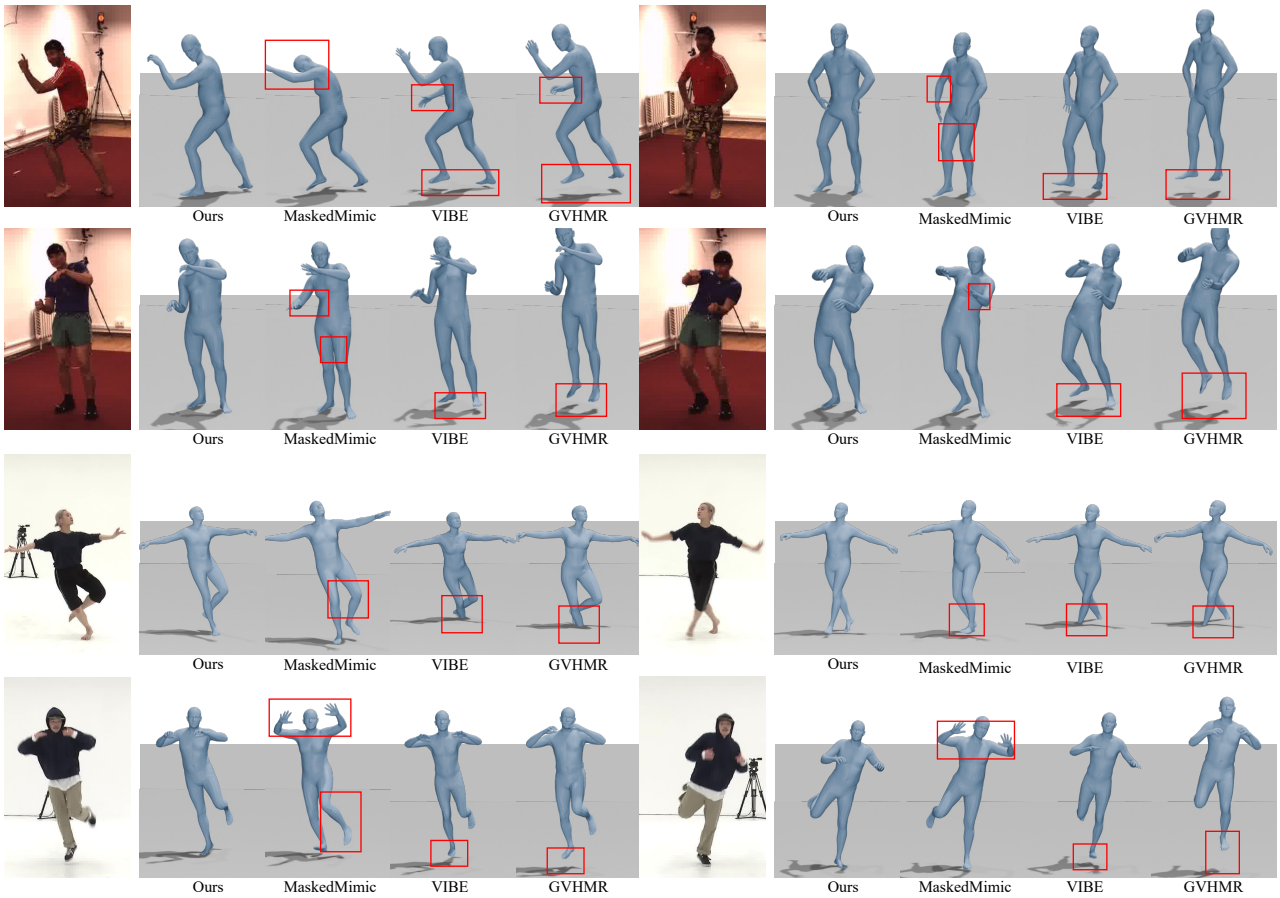

Figure 2: More comparison results on the Human3.6M and AIST dataset. We use red boxes to indicate the artifacts. The comparison results demonstrate that the motions recovered by our method show less floating, penetration, and wrong poses (from 2m15s to 2m44s in the accompanying video).

new cluster specifically for these motions and incorporated it into the original five clusters. Additionally, we increased the PD controller's parameters fivefold compared to the original settings to enhance its control capacity. This experiment demonstrates the flexibility of our method in controlling diverse motions under varying configurations. As a result, we construct a high-quality, physically plausible 3D human motion dataset from AIST, consisting of 3.1 hours of motion data at 30 frames per second (fps).

IDEA400. We cluster the IDEA400 dataset into three motion clusters. For sitting motions in IDEA400, we calculate a box for the character to sit on. The scale and position of this box are determined based on the lowest height of the vertices associated with the pelvis joint. To ensure realism, we constrain the box to avoid penetration into other body parts. After discarding 1.1 hours of low-quality data, our subset of IDEA400 comprises 22.9 hours of motion data at 30 fps. Most of the discarded data resulted from failures in calculating the correct sitting box.

Fitness. We cluster the fitness dataset into seven motion clusters. The cameras in these videos were level (no tilt), but the estimated camera poses had slight errors. Thus, we optimized the predicted camera poses to ensure they remained tilt-free. The total size of fitness dataset is 26.3 hours.

Human3.6M. In the comparison and evaluation experiments, we pre-train four option policies using the entire training set of Human3.6M and fine-tune them on the test set. Unlike the setting used for AIST and IDEA400, where we sampled 10% of the data for pre-training, we utilize the full training set of Human3.6M due to its relatively smaller scale.

5 Experiments and Evaluations

In this section, we show the details and the extra information about our experiments. More comparison results between our method and SOTA motion recovery methods are included in Fig. 2. More comparison between Motion-X dataset and our Mimic-X dataset can be found in Fig. 3 and Fig. 4.

5.1 Metrics

In this section, we report the formulations of the metrics used in our main manuscript.

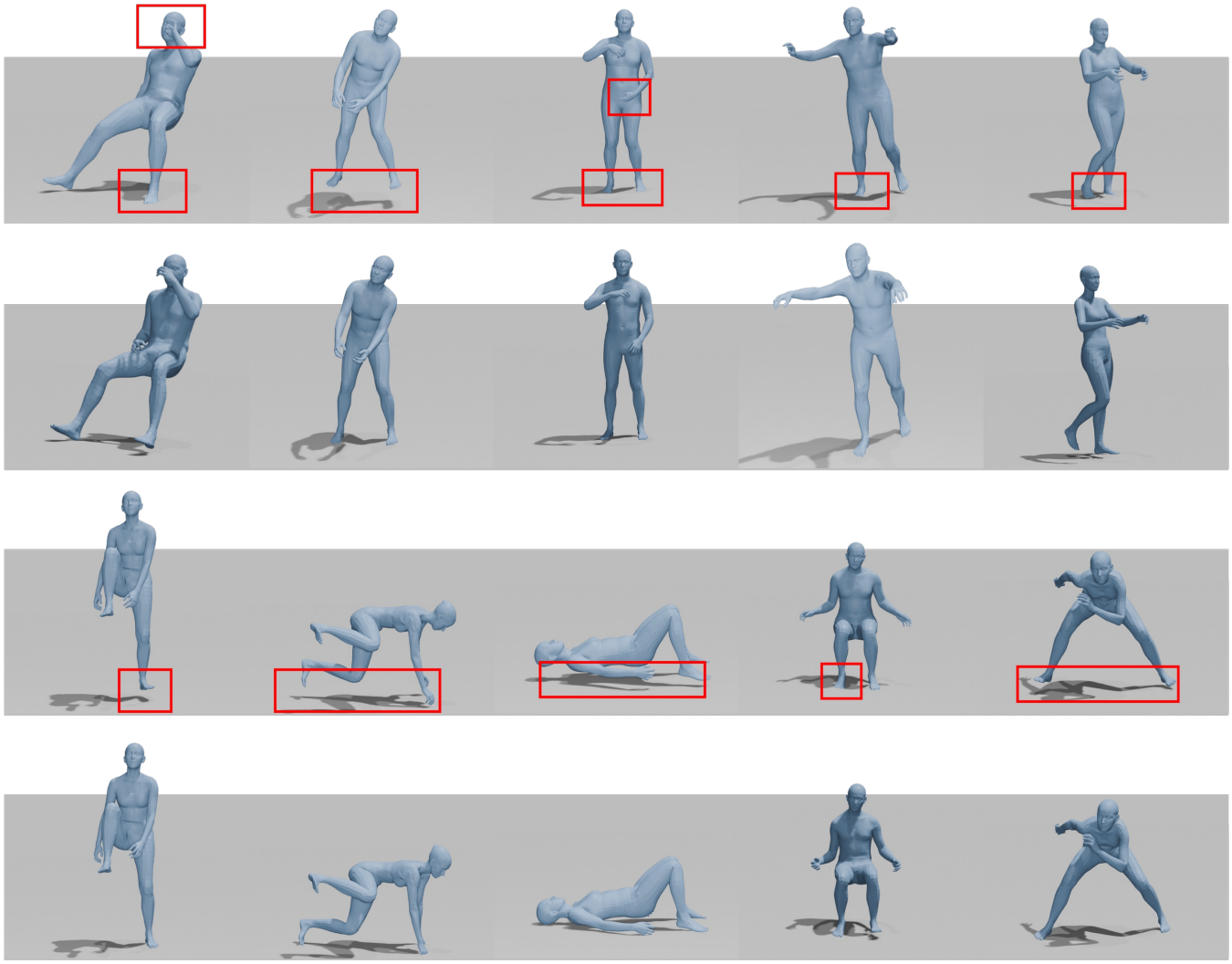

Figure 3: Additional comparisons between motions in the Motion-X dataset (rows 1 and 3) and our Mimic-X dataset (rows 2 and 4). Artifacts are highlighted with red boxes. These results demonstrate that the motion quality in our Mimic-X dataset surpasses that of the Motion-X dataset (from 0m38s to 1m22s in the accompanying video).

Mean root position error. The mean root position error (MRPE) is adopted from (Shimada et al. 2020). It can be formulated as follows:

$$MRPE = \frac{1}{T}\sum_{t=0}^{T-1} \|\hat{r}_t^p - r_t^p\|_2 \tag{9}$$

where $\hat{r}_t^p$, r_t^p are the root's generated and ground truth positions at the t_{th} frame, respectively. T is the sequence length of the test motion.

Mean per joint position error. The mean per joint position error (MPJPE) is adopted from (Ionescu et al. 2013). It can be formulated as follows:

$$MPJPE = \frac{1}{22T}\sum_{t=0}^{T-1}\sum_{j=0}^{21} \parallel \hat{\mathbf{p}}_t^j - \mathbf{p}_t^j \parallel_2 \tag{10}$$

where $\hat{\mathbf{p}}_t^j, \mathbf{p}_t^j$ are the t_{th} frame's j_{th} joint's generated and ground-truth positions relative to the root joint position, respectively. The calculation of Procrustes-aligned mean per-joint position error (PA-MPJPE) is identical to that of MPJPE, except that the joint positions undergo Procrustes alignment beforehand.

Velocity error. The velocity error (Vel) is calculated as follows:

$$Vel = \frac{1}{22T}\sum_{t=0}^{T-1}\sum_{j=0}^{21} \parallel \hat{v}_t^j - v_t^j \parallel_2 \tag{11}$$

where $\hat{v}_t^j, v_t^j$ are the generated and ground-truth joint velocities, represented by the difference of joint positions in the world coordinate between successive frames.

Acceleration error. The acceleration error (Accel) can be

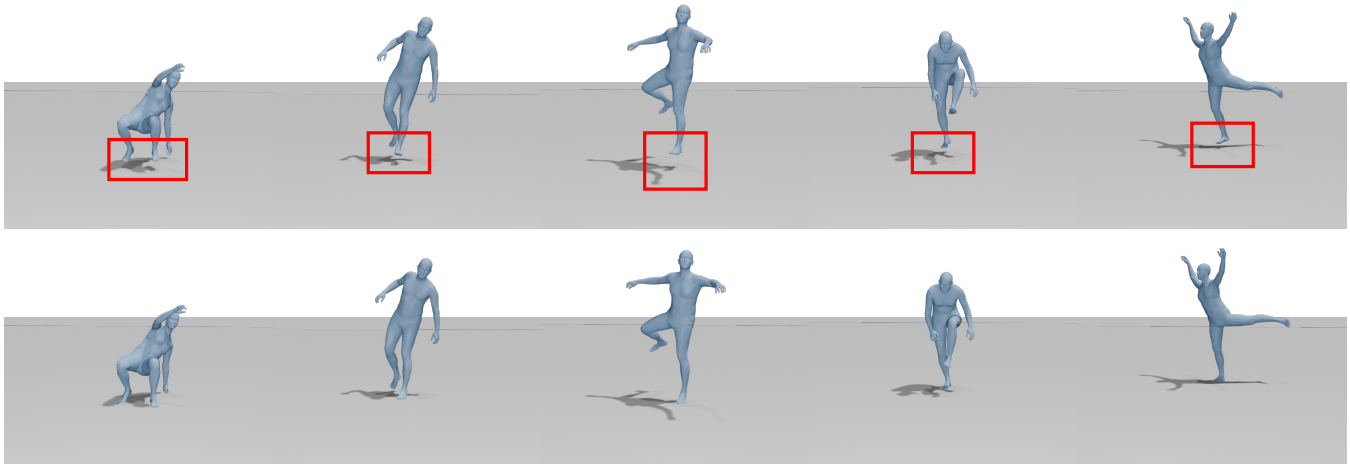

Figure 4: Comparisons between reference dancing motions optimized from GVHMR (row 1) and the corresponding imitated results in our Mimic-X dataset (row 2). Artifacts are highlighted with red boxes. These results demonstrate that our adaptive option framework is capable of mimicking complex motions, such as dancing, with producing high-quality and physically plausible results (from 1m28s to 1m40s in the accompanying video).

calculated as follows:

$$Vel = \frac{1}{22T}\sum_{t=0}^{T-1}\sum_{j=0}^{21} \parallel \hat{\dot{v}}_t^j - \dot{v}_t^j \parallel_2 \tag{12}$$

where $\hat{\dot{v}}_t^j, \dot{v}_t^j$ are the generated and ground-truth joint accelerations in world coordinate, represented by the difference of joint velocities v_t^j between successive frames.

Foot sliding. The foot sliding (FS) is calculated on the SMPLX vertices, adopted from (Yuan et al. 2021). We first detect the vertices that are under the ground as the contacting vertices, and calculate the average offset of the contacting vertices:

$$FS = \frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{|V_t^{con}|}\sum_{i\in V_t^{con}} \parallel vp_t^i - vp_{t+1}^i \parallel_2 \tag{13}$$

where V_t^{con} is the index set of contacting vertices, vp_t^i represents the position of the i_{th} vertex in t_{th} frame.

Ground Penetration. The ground penetration (GP) is also calculated on the SMPLX vertices. Following Yuan et al. (2021), we compute the average distance to the ground for vertices under the ground:

$$GP = \frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{|V_t^{con}|}\sum_{i\in V_t^{con}} \parallel vp_{t,y}^i \parallel_2 \tag{14}$$

where V_t^{con} is the index set of under ground vertices, $vp_{t,y}^i$ represents the height of the i_{th} vertex at the t_{th} frame.

Floating. The floating (Float) proposed in this paper is computed on the SMPLX vertices. To achieve this, the entire motion sequence is segmented into 2-second sub-sequences. For each sub-sequence, only the lowest vertex height above the ground is considered. This segmentation strategy helps distinguish between sustained floating and motions such as jumping. Additionally, to mitigate the impact of ground penetration on floating evaluation, we use the frame count where all vertices are above the ground as the weight for each segment. The complete formulation is as follows:

$$Float = (\sum_{S\in\mathcal{S}}|S| \cdot \min_{t\in S, i\in V} vp_{t,y}^i) / \sum_{S\in\mathcal{S}}|S| \tag{15}$$

where $\mathcal{S}$ is the set of sub-sequences, S represents the set of frame indices of ground-penetration-free frames in a 2-second sub-sequence, V is the vertex set, and $p_{t,y}^i$ is the height of vertex i at frame t.

Metrics modification for generation. In our motion generation evaluation, the generated motions are represented in joint positions rather than SMPLX parameters. Consequently, we compute FS, GP, and Floating directly on the joints instead of SMPLX vertices. Furthermore, instead of inferring ground contact from vertex heights, we use the contact labels provided in the data representation (Zhang et al. 2023a).

Dataset	Model	MRPE↓	MPJPE↓	PA-MPJPE↓
Human3.6M	MaskedMimic	95	60	49
	MaskedMimic-ft	265	121	60
	Ours	**59**	**33**	**26**
Human3.6M-tiny	MaskedMimic	86	57	48
	MaskedMimic-ft	74	53	44
	Ours	**59**	**32**	**25**

Table 1: Evaluations of motion recovery quality. The best results are highlighted as **1st**.

5.2 Comparison Details

In the comparison, the results for VIBE and GVHMR are obtained using their publicly available codes, while for PhysCap, SimPoE, PhysAware, TrajOp, and DiffPhy, whose

Sequence Name		
S09_WalkDog 1	S09_WalkDog	S09_Waiting
S09_Discussion 1	S09_Discussion 2	S09_Purchasing
S11_WalkDog 1	S11_WalkDog	S11_Waiting
S11_Waiting 1	S11_Directions	

Table 2: Sequence name that is not included in the Human3.6M-tiny.

codes have not been released, we directly use the metrics reported in their respective papers. For MaskedMimic, we utilize the released checkpoint to mimic motions in the test set. It is important to note that a fair comparison is not always possible, as our pipeline requires per-motion fine-tuning; methods like PhysCap, TrajOp, and DiffPhy also need per-motion optimization, while other methods do not process the motions in the test dataset. Therefore, our primary objective is to demonstrate the quality of motions produced by our approach, rather than a direct, controlled comparison.

Since the text descriptions for the fitness subset of Motion-X (Lin et al. 2023) are not publicly available, we use alternative subsets for training in our text-driven motion generation evaluation: for T2M-GPT (Zhang et al. 2023a), we train T2M-GPT-Mimic-X on the IDEA400 subset of our Mimic-X dataset and T2M-GPT-Motion-X on the IDEA400 subset of Motion-X, while for ReMoDiffuse (Zhang et al. 2023b), we train ReMoDiffuse-Mimic-X's motion VQVAE on both IDEA400 and fitness subsets of Mimic-X (without text descriptions) and its generator on only the IDEA400 subset, following the same approach for ReMoDiffuse-Motion-X. For the floating metric computation, we exclude stair-climbing motions and randomly select 100 sequences from HumanML3D to prevent interference. All other metrics are evaluated using the full test set. To approximate comparable sampling numbers, we increase the number of repeated generations for the floating metric to 300 (versus the standard 20 used for other metrics).

5.3 Finetuing MaskedMimic

For fair comparison, we fine-tune MaskedMimic (Tessler et al. 2024) on individual motion sequences, following the same protocol as our method. We retain the original training configuration and fine-tune each sequence until the reward reaches 0.8, termed as MaskedMimic-ft. The results are presented in Table 1. After fine-tuning, MaskedMimic fails to mimic certain motions and, in some cases, performs worse than before fine-tuning. After excluding these unsuccessful cases (listed in Table 2), denoted as Human3.6M-tiny, the fine-tuned MaskedMimic (MaskedMimic-ft) outperforms its original version but remains inferior to our approach. This demonstrates that directly fine-tuning a universal motion controller does not consistently improve the quality of all mimicked motions.

5.4 Ablation Details

Auto-encoder Module. Instead of directly remove the encoder module, to maintain a comparable model size, we add extra layers to both the policy and value networks, resulting in a size of 24.2MB for the new control networks (compared to 23.6MB for the control networks with the auto-encoder). We choose to deepen the network rather than widen it, as the encoder itself adds depth to the original architecture.

Option Arrangement. For *w/o dp*, we compute motion features as described in the main paper, and then calculate the Euclidean distance between these features and C cluster centers. Each sequence is assigned to the single nearest cluster.

Clustering. For *Whole4*$\times$, we primarily increase the network width (95.1MB), as arranging C control policies is structurally similar to using a single control policy with a wider network. The variety of cluster radius thresholds of 20, 50, and 70 result in 9, 2, and 1 clusters, respectively.

References

Ionescu, C.; Papava, D.; Olaru, V.; and Sminchisescu, C. 2013. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7): 1325–1339.

Konidaris, G.; and Barto, A. 2009. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22.

Lee, Y.; Sun, S.-H.; Somasundaram, S.; Hu, E. S.; and Lim, J. J. 2019. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*.

Lin, J.; Zeng, A.; Lu, S.; Cai, Y.; Zhang, R.; Wang, H.; and Zhang, L. 2023. Motion-x: A large-scale 3d expressive whole-body human motion dataset. *Advances in Neural Information Processing Systems*, 36: 25268–25280.

Liu, L.; Panne, M. V. D.; and Yin, K. 2016. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)*, 35(3): 1–14.

Moon, G.; Choi, H.; and Lee, K. M. 2022. Neuralannot: Neural annotator for 3d human mesh training sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2299–2307.

Pavlakos, G.; Choutas, V.; Ghorbani, N.; Bolkart, T.; Osman, A. A.; Tzionas, D.; and Black, M. J. 2019. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10975–10985.

Shen, Z.; Pi, H.; Xia, Y.; Cen, Z.; Peng, S.; Hu, Z.; Bao, H.; Hu, R.; and Zhou, X. 2024. World-Grounded Human Motion Recovery via Gravity-View Coordinates. In *SIGGRAPH Asia 2024 Conference Papers*, 1–11.

Shimada, S.; Golyanik, V.; Xu, W.; and Theobalt, C. 2020. Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Transactions on Graphics (ToG)*, 39(6): 1–16.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.

Tao, H.; Hou, S.; Zou, C.; Bao, H.; and Xu, W. 2023. Neural motion graph. In *SIGGRAPH Asia 2023 Conference Papers*, 1–11.

Tessler, C.; Guo, Y.; Nabati, O.; Chechik, G.; and Peng, X. B. 2024. Maskedmimic: Unified physics-based character control through masked motion inpainting. *ACM Transactions on Graphics (TOG)*, 43(6): 1–21.

Wang, J.; Hodgins, J.; and Won, J. 2024. Strategy and skill learning for physics-based table tennis animation. In *ACM SIGGRAPH 2024 Conference Papers*, 1–11.

Won, J.; Gopinath, D.; and Hodgins, J. 2020. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 39(4): 33–1.

Yuan, Y.; Wei, S.-E.; Simon, T.; Kitani, K.; and Saragih, J. 2021. Simpoe: Simulated character control for 3d human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7159–7169.

Zhang, J.; Zhang, Y.; Cun, X.; Zhang, Y.; Zhao, H.; Lu, H.; Shen, X.; and Shan, Y. 2023a. Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 14730–14740.

Zhang, M.; Guo, X.; Pan, L.; Cai, Z.; Hong, F.; Li, H.; Yang, L.; and Liu, Z. 2023b. Remodiffuse: Retrieval-augmented motion diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 364–373.

Block name	Output size	Inputs	Outputs
		Encoder	
linear1+ReLU	1024	$\mathcal{X}_t$	the linear1 features
linear2+ReLU	512	the linear1 features	the linear2 features
linear3+ReLU	512	the linear2 features	the linear3 features
linear4	128	the linear3 features	the latent vectors
		Decoder	
linear1+ReLU	1024	the latent vectors	the linear1 features
linear2+ReLU	512	the linear1 features	the linear2 features
linear3+ReLU	512	the linear2 features	the linear3 features
linear4	350	the linear3 features	$\tilde{\mathcal{X}}_t$

Table 3: Network parameters of the auto-encoder.

Block name	Output size	Inputs	Outputs
linear1+ReLU	512	the latent vectors	the linear1 features
linear2+ReLU	256	the linear1 features	the linear2 features
linear3+ReLU	1024	concatenation of $\hat{\mathcal{P}}_t$ and the linear2 features	the linear3 features
linear4+ReLU	1024	the linear3 features	the linear4 features
linear5+ReLU	512	the linear4 features	the linear5 features
linear6	49	the linear5 features	the action a_t

Table 4: Network parameters of the policy network.

Block name	Output size	Inputs	Outputs
linear1+ReLU	1024	concatenation of $\hat{\mathcal{P}}_t$ and the latent vectors	the linear1 features
linear2+ReLU	1024	the linear1 features	the linear2 features
linear3+ReLU	512	the linear2 features	the linear3 features
linear4	1	the linear3 features	the value vector

Table 5: Network parameters of the value network.

Block name	Output size	Inputs	Outputs
linear1+ReLU	512	$\mathcal{X}_t$	the linear1 features
linear2+ReLU	256	the linear1 features	the linear2 features
linear3+ReLU	1024	concatenation of $\hat{\mathcal{P}}_t$ and the linear2 features	the linear3 features
linear4+ReLU	1024	the linear3 features	the linear4 features
linear5+ReLU	512	the linear4 features	the linear5 features
linear6+ReLU	512	the linear5 features	the linear6 features
linear7+ReLU	512	the linear6 features	the linear7 features
linear8+ReLU	512	the linear7 features	the linear8 features
linear9	49	the linear8 features	the action a_t

Table 6: Network parameters of the policy network without encoder.

Block name	Output size	Inputs	Outputs
linear1+ReLU	1024	concatenation of $\hat{\mathcal{P}}_t$ and $\mathcal{X}_t$	the linear1 features
linear2+ReLU	1024	the linear1 features	the linear2 features
linear3+ReLU	512	the linear2 features	the linear3 features
linear4+ReLU	512	the linear3 features	the linear4 features
linear5+ReLU	512	the linear4 features	the linear5 features
linear6+ReLU	512	the linear5 features	the linear6 features
linear7	1	the linear6 features	the value vector

Table 7: Network parameters of the value network without encoder.

Block name	Output size	Inputs	Outputs
		Encoder	
linear1+ReLU	2048	$\mathcal{X}_t$	the linear1 features
linear2+ReLU	1024	the linear1 features	the linear2 features
linear3+ReLU	1024	the linear2 features	the linear3 features
linear4+ReLU	128	the linear3 features	the latent vectors
		Decoder	
linear1+ReLU	2048	the latent vectors	the linear1 features
linear2+ReLU	1024	the linear1 features	the linear2 features
linear3+ReLU	1024	the linear2 features	the linear3 features
linear4+ReLU	350	the linear3 features	$\tilde{\mathcal{X}}_t$

Table 8: Network parameters of the auto-encoder for *Whole4*×.

Block name	Output size	Inputs	Outputs
linear1+ReLU	1024	the latent vectors	the linear1 features
linear2+ReLU	512	the linear1 features	the linear2 features
linear3+ReLU	2048	concatenation of $\hat{\mathcal{P}}_t$ and the linear2 features	the linear3 features
linear4+ReLU	2048	the linear3 features	the linear4 features
linear5+ReLU	1024	the linear4 features	the linear5 features
linear6+ReLU	1024	the linear5 features	the linear6 features
linear7+ReLU	512	the linear6 features	the linear7 features
linear8	49	the linear7 features	the action a_t

Table 9: Network parameters of the policy network for *Whole4*×.

Block name	Output size	Inputs	Outputs
linear1+ReLU	2048	concatenation of $\hat{\mathcal{P}}_t$ and the latent vectors	the linear1 features
linear2+ReLU	2048	the linear1 features	the linear2 features
linear3+ReLU	1024	the linear2 features	the linear3 features
linear4+ReLU	1024	the linear3 features	the linear4 features
linear5+ReLU	512	the linear4 features	the linear5 features
linear6	1	the linear5 features	the value vector

Table 10: Network parameters of the value network for *Whole4*×.