Supplemental Material for Hierarchical Generation of Human-Object Interactions with Diffusion Probabilistic Models

Huaijin Pi[1], Sida Peng[1], Minghui Yang[2], Xiaowei Zhou[1], Hujun Bao[1]*
[1] State Key Lab of CAD&CG, Zhejiang University [2] Ant Group

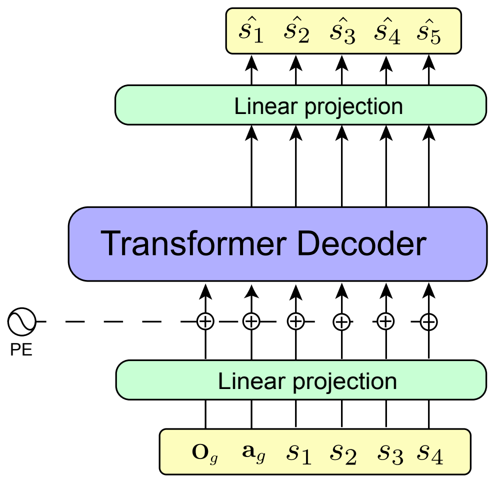

$\hat{s}_1$ $\hat{s}_2$ $\hat{s}_3$ $\hat{s}_4$ $\hat{s}_5$
Linear projection
Transformer Decoder
PE
Linear projection
$\mathbf{O}_g$ $\mathbf{a}_g$ s_1 s_2 s_3 s_4

Figure 1. **Overview of the transformer for part VQ-VAE inference.** The code is predicted in an auto-regressive manner using the GPT-like transformer.

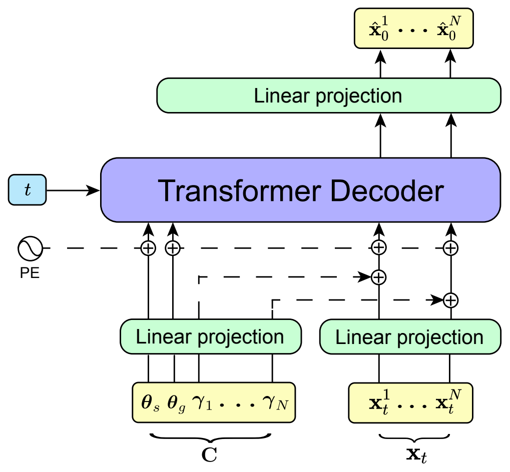

$\hat{\mathbf{x}}_0^1 \dots \hat{\mathbf{x}}_0^N$
Linear projection
t
Transformer Decoder
PE
Linear projection
Linear projection
$\boldsymbol{\theta}_s$ $\boldsymbol{\theta}_g$ $\gamma_1 \dots \gamma_N$
$\mathbf{x}_t^1 \dots \mathbf{x}_t^N$
$\mathbf{C}$
$\mathbf{x}_t$

Figure 2. **Overview of transformer DDPM for milestone pose generation.** Unlike the model for milestone point generation, the inputs are summed with frame-wise conditions.

1. Social Impact

Our proposed framework can generate realistic human motions, which might be misused for generating fake videos with the techniques in neural rendering.

2. Implementation Details

2.1. Part VQ-VAE

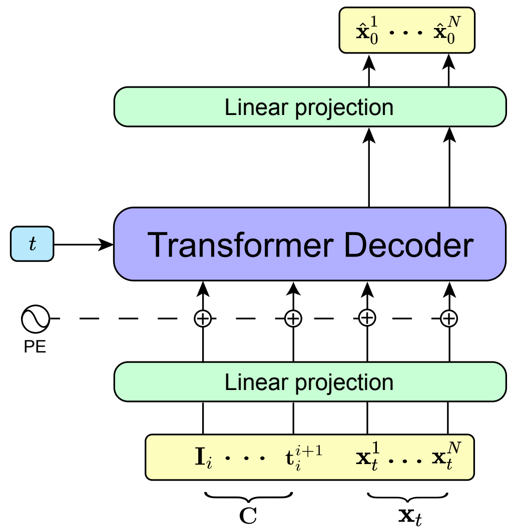
We train the part VQ-VAE for 50 epochs. The human skeleton is split into 5 parts as *body*, *right hand*, *left hand*, *right leg*, and *left leg* like [12], each of which contains 6, 4, 4, 4, and 4 joints. The sizes of all codebooks are set as 8 and the dimension of the code is 4. The encoders of all splits are 3 layers MLPs with 256 channels. The decoder is a 4 layers MLP with 256 channels. The β for commitment loss [22, 16] is 0.1.

We use a GPT-like [2] transformer with 4 transformer decoder layers [23] to build the auto-regressive transformer like [4]. The model is illustrated in Fig. 1 and trained for 1 epoch. We set the number of heads in multi-head attention to 4, the dropout rate to 0.1, and the dimension of the intermediate feedforward network to 1024. For inference, the codes are predicted with a pre-defined order: *body*, *left leg*, *right leg*, *right hand*, and *left hand*. The full training of part VQ-VAE costs 5 minutes on a TITAN Xp GPU.

2.2. Transformer DDPM

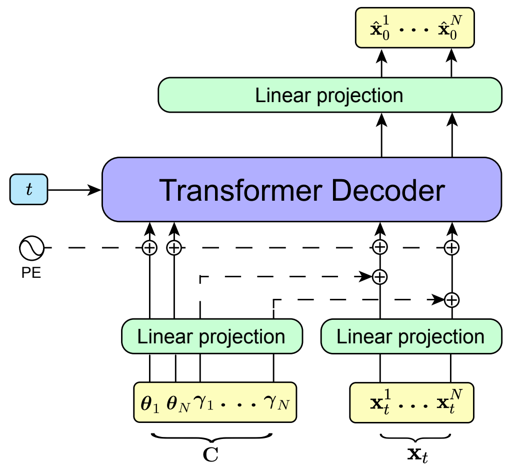
The transformer DDPM models for milestone point generation, milestone pose generation (Fig. 2), trajectory completion (Fig. 3), and motion infilling (Fig. 4) are trained for 250, 100, 250, and 10 epochs. The training times of these models are 12, 12, 10, and 1 hours on a TITAN Xp GPU. During training, the number of sampling steps for all DDPM models is set to 1000 [10]. We set the forward process variances to constants increasing with the cosine schedule [14] from $\beta_1 = 0.0001$ to $\beta_T = 0.02$. For inference, we accelerate DDPM models with DDIM [18] and set $T = 250$. All models are set with 4 transformer decoder layers [23]. We set the number of heads in multi-head at-

*Corresponding author.



$\hat{\mathbf{x}}_0^1 \cdots \hat{\mathbf{x}}_0^N$
Linear projection
t
Transformer Decoder
PE
Linear projection
$\mathbf{I}_i \cdots \mathbf{t}_i^{i+1}$ $\mathbf{x}_t^1 \cdots \mathbf{x}_t^N$
$\mathbf{C}$
$\mathbf{x}_t$

Figure 3. **Overview of transformer DDPM used for trajectory completion.** Different from the model used for milestone generation, we remove the length head.



$\hat{\mathbf{x}}_0^1 \cdots \hat{\mathbf{x}}_0^N$
Linear projection
t
Transformer Decoder
PE
Linear projection
Linear projection
$\boldsymbol{\theta}_1\ \boldsymbol{\theta}_N \gamma_1 \cdots \gamma_N$
$\mathbf{x}_t^1 \cdots \mathbf{x}_t^N$
$\mathbf{C}$
$\mathbf{x}_t$

Figure 4. **Overview of transformer DDPM for motion infilling.** The inputs are summed with frame-wise conditions.

tention to 4, the dropout rate to 0.1, and the dimension of the intermediate feedforward network to 1024.

2.3. Representation

Pose representation. In our paper, the pose representation follows previous works [19, 6, 30]. We use $\boldsymbol{\theta}_i$ to represent the local pose at the i-th frame, which is composed of $\{\mathbf{j}_i^p, \mathbf{j}_i^r, \mathbf{j}_i^v\}$, where $\mathbf{j}_i^p \in \mathbb{R}^{3k}$, $\mathbf{j}_i^r \in \mathbb{R}^{6k}$, $\mathbf{j}_i^v \in \mathbb{R}^{3k}$ are the positions, rotations, and velocities of k joints related to the root. $\mathbf{r}_i = \{\mathbf{r}_i^p, \mathbf{r}_i^d\}$ are the root position and forward direction.

Milestone point representation. We employ a bi-directional representation [19] for milestone points:

$$\mathbf{m}_i = \{\mathbf{r}_i^b, \mathbf{c}_i, \mathbf{w}_i\}. \tag{1}$$

The representation of milestone points is the root of these points, which is similar to previous works [19, 6, 30]. It is projected on the ground so the position and forward directions are 2 dimensions instead of 3. The $\mathbf{r}_i^b \in \mathbb{R}^8$ is bi-directional [19, 6, 30] and contains the root position and forward directions relative to the starting point and the goal point $(2*(2+2) = 8)$. $\mathbf{c}_i \in \mathbb{R}^5$ are contact labels indicating the contact between the environment and the body including the pelvis, feet, and hands, also same as previous works [19, 6, 30].

To capture the state of the milestone, we also predict a high-dimensional feature at milestone i which encodes the information of a 2-second (61 frames) window like previous works [19, 6, 30]:

$$\mathbf{w}_i = \{\mathbf{w}_i^p, \mathbf{w}_i^d, \mathbf{w}_i^a\}. \tag{2}$$

Specifically, $\mathbf{w}_i^p \in \mathbb{R}^{2\mathcal{T}}$ and $\mathbf{w}_i^d \in \mathbb{R}^{2\mathcal{T}}$ are the root positions and forward directions in the nearby window relative to the root of milestone i. $\mathcal{T} = 13$ denotes the number of frames selected to form this feature. $\mathbf{w}_i^a \in \mathbb{R}^{n_a \times \mathcal{T}}$ denotes action labels. The number of actions is denoted by n_a. Following [19, 6, 30], the action variable $\mathbf{w}_i^a$ is in the one-hot label shape but the value is continuous between $[0, 1]$, as there are transitions between different actions.

2.4. Root representation for trajectory completion.

The root representation for trajectory completion is similar to the one used in milestone point generation. The only difference is that the bi-directional scheme here indicates the root positions and directions relative to the two consecutive milestone points.

2.5. Implementations of the results in a cluttered scene.

Due to the lack of cluttered objects in the training data, we find directly applying our method may generate paths with collision. Similar to other diffusion model methods [28, 21], DDPM has the ability to edit [28] and interpolate [21]. Thus we run our method with the guide of A*. Because the diffusion process is done in an interactive manner, we can inject the path generated by A* into the iterative sampling. Specifically, we obtain the interpolating predictions $\hat{\mathbf{m}}_0$ in one step of the diffusion process with smoothing factor s as

$$\hat{\mathbf{m}}_0 = s * f_{\mathbf{m}}(\mathbf{m}_t, t, C) + (1 - s) \cdot \mathbf{m}_a, \tag{3}$$

where $\mathbf{m}_a$ is the path generated by A*. Then, we noise it back to $\mathbf{m}_{t-1}$. This is repeated from $t = T$ until the final

s	FD↓	APD$_M$ ↑	APD$_P$ ↑	APD$_T$ ↑	Percentage ↓	Sliding↓
0.25	23.80	4.00	4.38	75.45	**3.6**	0.55
0.50	23.06	4.14	4.46	76.47	3.8	0.53
0.75	22.64	**4.19**	**4.62**	80.18	3.9	0.56
1.00	**22.34**	4.06	4.52	**91.38**	7.5	**0.50**

Table 1. **The effect of the interpolation factor.** The percentage of frames with penetration is calculated in a cluttered scene and other metrics are calculated in the test setting described in Sec. 3.1. Ratio: The percentage of frames with penetration.

$\mathbf{m}_0$ is achieved. Consequently, we obtain the final results which combine the generation and the planned path by A*.

The effect of the magnitude of s is shown in Tab. 1. We calculate the percentage of penetration frames in the cluttered scene and calculate other metrics in the setting described in Sec. 3.1. Empirically, we find that different magnitudes of s have similar results. Because our trajectory completion module learns to generate trajectories that avoid obstacles from the motions of characters walking around objects, the percentage of penetration frames of our method is lower than SAMP (5.2%).

3. Datasets and Evaluation

3.1. Test setting

All experiments are trained and tested on the same dataset. For each dataset, we select some test sequences for providing the starting point, starting pose, and endpoint as they are the input of our task. We do not use any other information from the test sequences. As SAMP proposes GoalNet [6] to generate the goal point on the object, we compare our approach with previous methods on the SAMP dataset using the generated goal from the GoalNet. For the COUCH and NSM datasets, we use the goal point on the object provided by the datasets. For each sequence, we run experiments on unseen objects multiple times.

3.2. Datasets

SAMP. Since SAMP [6] does not provide the details about their testing, we compare our method with other baselines in the following setting. On the SAMP [6] dataset, 6 sequences are selected as the testing sequences from the original paper [6]. For each sequence, we test with 3 unseen objects and generate 10 sequences for each object. Therefore, there are 180 generated sequences in total.

COUCH. The COUCH dataset [30] provides 100 testing sequences and 30 objects for testing from their released code, we do not have enough computation resources for running on all combinations of test sequences and objects. We select 30 sequences for the starting point, starting pose, and endpoint. For each sequence, we choose 1 object and generate 5 sequences. Consequently, we have 150 sequences in total and test with all objects.

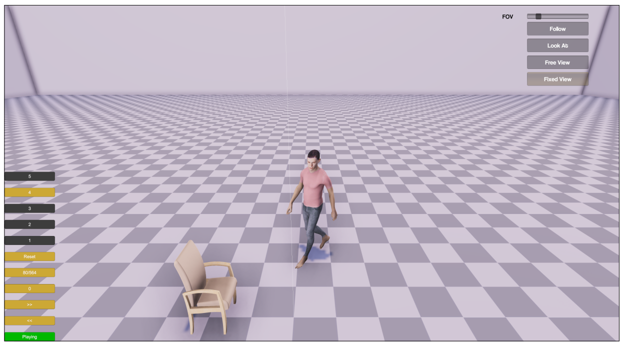


Figure 5. **User study GUI.** The volunteer is asked to rate sequences using the provided GUI.

NSM. On the NSM [19] dataset, we remove the sequences that contain the interactions with dynamic objects and select 4 sequences as the testing sequences for our method. For NSM [19], we use the provided pre-trained model and we retrain SAMP [6] on all data. For each sequence, we test with 1 object and generate 10 sequences for each object.

PROX. Given that [24] did not release their annotations of actions on the PROX [7], we do not conduct experiments on the PROX [7] dataset.

3.3. Metrics

FD. Like previous work [6, 24], we employ FD [9] to measure the motion quality. We calculate FD using the following features $\mathbf{X}_i = \{\mathbf{w}_i^\theta, \mathbf{w}_i^r\}$ at frame i in the sequence, where $\mathbf{w}_i^\theta$ and $\mathbf{w}_i^r$ are high-dimensional features about local poses and trajectories at frame i. To be more specific, $\mathbf{w}_i^\theta \in \mathbb{R}^{135\mathcal{T}}$ and $\mathbf{w}_i^r \in \mathbb{R}^{4\mathcal{T}}$ are the human poses and the root trajectories relative to the root of frame i with length $\mathcal{T} = 13$ sampled uniformly in a 2-second window between $[-1, 1]$ seconds [19, 6, 30]. The human pose is represented by the pelvis position and the 6D rotations [33] of 22 joints which is 135 dimensions in total. The root trajectory is represented by the position and the direction which are 4 dimensions. Following the form of FID [9], we calculate the distribution of $\mathbf{X}_i$ and compare it with the ground-truth data [6].

User study. We invite some volunteers from different backgrounds to rate generated results. Each sequence is scored by at least 3 participants. The highest score is 5 and the lowest score is 1. The volunteer is asked to consider both physical plausibility and motion naturalness to rate sequences. The user study GUI is shown in Fig. 5.

APD. We employ APD like previous work [6, 24] to measure the diversity. We calculate motion APD, pose APD, and trajectory APD.

Motion APD. For motion diversity, we employ the formal definition of Frechét distance [1] to measure the distance between the predictions of various lengths. For sequence A and sequence B, the distance is defined as the infimum over all reparameterizations α and β of $[0, 1]$ of the maximum over all $t \in [0, 1]$ of the distance between $A(\alpha(t))$ and $B(\beta(t)))$ [1], which is shown as

$$F(A, B) = \inf_{\alpha,\beta} \max_{t\in[0,1]} \{d(A(\alpha(t)), B(\beta(t)))\}. \tag{4}$$

It should be noticed that this form is different from the FD used for measuring motion quality [9]. Since the total length of the generated sequences can be hundreds of frames, we select the frames with an interval of 30 for fast calculation. We use the human pose in these frames, which contains the pelvis position and the 6D rotations of 22 joints relative to its root, to calculate the distance.

Pose APD. We use the pose that human interacts with the object for pose APD. We measure the pose APD using the 6D rotations of human poses which are 132 dimensions.

Trajectory APD. For trajectory diversity, we also employ the formal definition of Frechét distance [1] to measure the distance between the predictions of various lengths. This form is the same as the one for measuring motion diversity [1]. Like motion APD, we select the frames with an interval of 30 for fast calculation. We use the root positions at these frames relative to the starting point to calculate the distance.

PE and RE. We calculate the positional error (PE) and rotational error (RE) when reaching the goal location of the object like previous works [19, 6]. We do not calculate them on the COUCH dataset [30] as the objects from the COUCH dataset are not annotated by the target goal position, which is the target pelvis position when the character sits on the object. From the released code of COUCH [30], they directly use the position of the object but the position of the object is not identical to the pelvis. As a result, the PE and RE in the COUCH dataset cannot represent the precision of the character's pelvis.

Penetration ratio. The penetration ratio between the character and the object is calculated in a similar way as the non-collision score proposed by [31]. Since there is no scene geometry, we do not calculate the contact score [31, 29, 25, 24]. Instead, we use foot sliding to measure the physical plausibility.

Percentage of frames with penetration. We calculate the penetration ratio of each frame in a sequence to indicate whether the character collides with the environment [6]. Then the percentage of frames with penetration is the percentage that the penetration ratio is larger than 0.

Foot sliding. Similar to [20], foot sliding is calculated by finding the closest foot joint of the body to the ground and measuring its velocity. We only calculate foot sliding when the character is walking since it may slide its feet during sitting and lying down.

4. Details and Analysis of Baselines

4.1. Implementation details of baselines

We compare our approach with online methods including NSM [19], SAMP [6], and COUCH [30] as they are the most relevant ones to this setting. Most other autoregressive methods are not designed for human-object interactions [11, 32]. We also compare with offline methods including SLT [25] and TDNS [24] as they focus on long-term motion generation while others are short-term [3, 26]. Because they are conducted in different settings, we modify their inputs and outputs to our settings.

NSM. We directly use the released models from NSM for the experiments on the NSM dataset [19]. We train NSM on the COUCH dataset [30] and the input and output are the same as the ones on the NSM dataset. We do not run NSM on the SAMP dataset [6] as the SAMP dataset does not provide the phase labels.

SAMP. We directly use the released models from SAMP for the experiments on the SAMP dataset [19]. We train SAMP on the NSM dataset [19] and COUCH dataset [30], where the input and output are the same as the ones on the SAMP dataset.

COUCH. As COUCH [30] only releases its code and data without pre-trained models, we train COUCH using their released code. We only conduct COUCH on the COUCH dataset as it requires the input of hand contacts. The hand contact model is trained from the COUCH dataset, so we do not run COUCH on other datasets.

SLT. SLT [25] conducts experiments on the PROX dataset in a very different setting from ours. The original SLT [25] requires the path points provided by users and only tests for at most 6 seconds. We modify it [25] to our setting and use A* [5] to generate paths. We select points along the path, forming the provided path points for SLT. As the original input is the point cloud of the scene, we replace it with the same object representation as our method. As SLT [25] assumes SMPL [13, 17, 15] representation, we do not conduct it on the NSM dataset as the character from the NSM dataset is different.

TDNS. TDNS [24] conducts experiments on the PROX dataset [7] and did not release the code. The original setting of TDNS [24] uses the point cloud of the scene. We replace it with the same object representation as our method. They employ POSA [8] to find the location of the synthesized interaction anchor in the scene. However, we only have one object so we use the goal predicted by GoalNet [6] or provided by datasets [19, 30], and keep POSA [8] for post-processing. As TDNS [24] also assumes SMPL [13, 17, 15] representation, we do not conduct it on the NSM dataset as the character from the NSM dataset is different.

4.2. Analysis of the baselines

We further analyze the results of SLT [25] and TDNS [24] to show that our implementation is promising. As indicated by the results in Tab. 4, the FD of motion quality increases by using A* and Neural Mapper [24]. This is consistent in Tab. 1 in the main paper that SLT [25] and TDNS [24] have much higher FD. Furthermore, SLT and TDNS employ cVAE-based architectures while we employ DDPM, which leads to another FD gap. Consequently, we think the margins between our method and baselines are promising.

5. Details and Analysis of Ablations

5.1. Implementation details of ablations

Impact of each sub-module. We provide more details about how we construct the variants.

Without GP: We remove the goal pose module in our framework. Therefore, we directly synthesize the milestones based on the object and the starting point. Then we generate the trajectories and infill the motions.

Without MP: We remove the milestone pose generation module. Then we directly generate the motions based on the trajectory in a sequence-level auto-regressive way. Specifically, we still employ the same network architecture as the motion infilling module and remove the condition of the pose at the last frame. We first synthesized motions along the trajectory with a sequence length of 61. Then we input the pose of the last frame and generate another sequence of motions with length 61. We keep this process auto-regressively and obtain the final results.

Without MT: We remove the milestone point generation module. We directly synthesize the whole trajectory with hundreds of frames. Then we synthesize the motions in the same way as the variant without MP. The GPU memory of this variant is large and directly training it with the batchsize 256 leads to GPU explosion. We are forced to use a small batchsize 16 and train this variant for 6 days while the milestone point generation costs around 12 hours.

Without TC: We remove the trajectory completion module. Instead, we generate the motions and trajectory in one network together.

Variants	FD ↓	APD$_M$ ↑	APD$_P$ ↑	APD$_T$ ↑	Penetration↓	Sliding↓
cVAE	27.06	3.36	3.27	90.52	4.36	**0.47**
DDPM	34.22	3.75	3.37	84.76	4.31	0.49
VQ-VAE	24.77	3.78	3.87	89.19	4.27	0.48
Part VQ-VAE	**22.34**	**4.06**	**4.52**	**91.38**	**4.00**	0.50

Table 2. **Ablation study of goal pose generation.** We implement the goal pose module with different architectures.

Variants	FD↓	APD$_M$ ↑	APD$_P$ ↑	APD$_T$ ↑	Penetration↓	Sliding↓
Part VQ-VAE	28.71	3.85	3.62	83.51	4.32	0.89
DDPM	**22.34**	**4.06**	**4.52**	**91.38**	**4.00**	**0.50**

Table 3. **Ablation study of part VQ-VAE for motion infilling.** We replace the DDPM as part VQ-VAE to predict motions.

Variants	FD↓	APD$_M$ ↑	APD$_P$ ↑	APD$_T$ ↑	Penetration↓	Sliding↓
A*	40.59	4.07	4.42	42.98	4.24	0.88
NM [24]	40.54	**4.15**	4.31	52.74	4.15	0.96
MT	**22.34**	4.06	**4.52**	**91.38**	**4.00**	**0.50**

Table 4. **Ablation study of milestone generation.** We compare our method with the variant based on the path generated by A* path planning [5]. NM: Neural Mapper [24]. MT: milestone point generation.

5.2. More analysis of ablations

Goal pose generation. For the variant that employs part VQ-VAE for motion infilling, we encode the continuous pose into the part VQ representation and employ the transformer to predict the codes. As shown in Tab. 2 and Tab. 3, the part VQ-VAE achieves better performance for goal pose generation but inferior performance for motion infilling than DDPM. We think the reason might be that the motion data is sufficient for continuous methods like DDPM but the goal pose data is not enough as most frames in a sequence are walking instead of sitting or lying.

Milestone generation. We use A* to plan the path instead of directly connecting the object and the starting point because connecting two points without A* may give colliding paths when the start point is behind the object (as shown in Fig. 6). Although A* is deterministic, the trajectories are not deterministic because the GoalNet [6] generates diverse goals on the object. As demonstrated in Tab. 4, the motion quality and the diversity of trajectories of these variants are worse. The reason why these variants perform poorly might be the low diversity of trajectory that affects the distribution of generated motions for calculating FD.

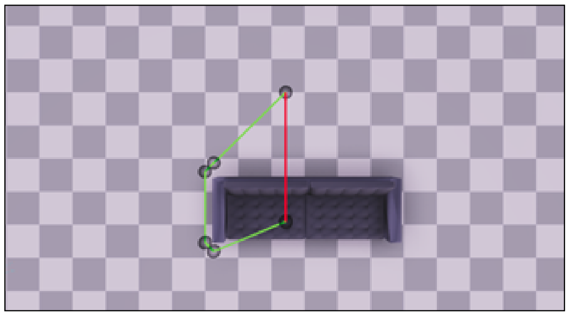


Figure 6. **The necessity of applying A*.** A* (green lines) vs. connecting two points (red lines) when it is behind the object.

Variants	FD↓	APD$_M$ ↑	APD$_P$ ↑	APD$_T$ ↑	Penetration↓	Sliding↓
Auto-regressive	33.39	**4.07**	3.66	76.48	6.87	0.98
Ours	**22.34**	4.06	**4.52**	**91.38**	**4.00**	**0.50**

Table 5. **Ablation study of the hierarchical generation framework.** Auto-regressive means we remove the hierarchical motion generation pipeline and employ an auto-regressive model to generate motions.

Order	FD↓	APD$_M$ ↑	APD$_P$ ↑	APD$_T$ ↑	Penetration↓	Sliding↓
A	22.80	4.01	4.35	**94.44**	4.09	**0.47**
B	22.93	**4.12**	4.46	94.21	4.02	0.52
Ours	**22.34**	4.06	**4.52**	91.38	**4.00**	0.50

Table 6. **Part order.** Different orders have similar results. A order: *left leg*, *right leg*, *right hand*, *left hand*. B order: *right hand*, *left hand*, *body*, *left leg*, *right leg*.

6. More Ablations

6.1. Design choice of the framework

Hierarchical motion generation. We build a variant without multi-stage motion generation to study the effect of our hierarchical motion generation pipeline. Specifically, we only employ the goal pose module to explicitly generate the goal pose. And we input the goal pose to SAMP as the condition to generate the motions auto-regressively. As demonstrated in Tab. 5, without hierarchical modeling, the generated motion quality is much worse than ours, where the FD increases.

6.2. Design choice of the submodules

Part orders. Different orders do not make significant differences as shown by Tab. 6.

Length prediction. To explore the effect of length prediction, we compare it with the variant that determines the length by the distance to the goal as $N = \lceil D/h \rceil$ where h is the distance the character may walk in 2 seconds (61 frames). As demonstrated in Tab. 7, the diversity of generated trajectories of the variant is much worse. We find that the character is forced to walk directly toward the object due to the length restriction. The motion quality also drops. This experiment shows that the diversity of the trajectory is important to the motion quality.

Variants	FD↓	APD$_M$ ↑	APD$_P$ ↑	APD$_T$ ↑	Penetration↓	Sliding↓
Distance	28.96	3.95	4.19	49.22	4.08	0.65
LP	**22.34**	**4.06**	**4.52**	**91.38**	**4.00**	**0.50**

Table 7. **Abaltion study of length prediction.** We compare our method with the variant that uses distance as the length. LP: Length prediction.

Variants	FD↓	APD$_M$ ↑	APD$_P$ ↑	APD$_T$ ↑	Penetration↓	Sliding↓
S [25]	29.44	**4.33**	4.37	88.02	4.30	0.53
Shared	23.14	4.06	4.52	90.74	4.16	0.52
MP	**22.34**	4.06	**4.52**	**91.38**	**4.00**	**0.50**

Table 8. **Ablation study of the impact of milestone pose generation.** S: generating the milestone pose separately. Shared: a shared model to generate milestone points and poses together.

Method	FD↓	Parameters (M) ↓	Training (hour)↓	Inference (second)↓
MoE	74.33	15.9	**8**	**1.30**
SAMP	57.34	18.5	9	1.49
SLT*	68.83	**6.0**	11	177.32 (0.04)
TDNS*	46.60	23.7	36	180.51 (3.24)
Ours	**22.34**	28.3	35	7.13

Table 9. **Complexity.** The training and inference time (720 frames) on a TITAN Xp GPU. The numbers in brackets are inference time without optimization.

Milestone pose generation. To further analyze the effect of milestone pose generation, we compare it with the variant that generates the static poses separately like SLT [25], which does not consider the temporal dependency of the poses. Here, we employ our proposed part VQ-VAE instead of cVAE in SLT to avoid the effect of the architecture of the pose generation. As demonstrated in Tab. 8, generating the poses separately shows a little higher motion diversity as indicated by APD$_M$. The reason is that generating poses separately makes the results more diverse, but this variant does not consider the temporal dependency which leads to much worse motion quality. We also implement a variant that employs a shared model to generate milestone points and poses together. As the results of this variant are worse, we apply separate models to generate milestone points and poses like previous works [25, 24].

Parameters. Tab. 9 compares the parameters, training time, and inference time on the SAMP dataset [6]. Our method exhibits comparable parameters and training time to TDNS [24] while achieving a much lower FD. Compared with offline methods such as TDNS and SLT [25], our method produces superior results without the need for inference-time optimization.

7. Qualitative Results

7.1. Visualization of generated trajectories.

Fig. 7 compares the generated trajectories by our method and other baselines. As shown in the figure, our generated trajectories are more diverse than baseline methods. Specifically, the baseline methods are prone to walk straightly toward the object while our method is able to reach the target object in two directions.

7.2. Visualization of the length of the generated sequences.

Fig. 8 compares the length of the sequences. We calculate the length from the character starts to walk to the character finishes sitting. As shown in the figure, our results are more similar to the dataset.

7.3. More qualitative results on the SAMP dataset.

Fig. 9 compares the poses that the character interacts with the object on the SAMP dataset. As shown by the figure, our results are more diverse than the baselines.

7.4. Qualitative results on the COUCH dataset.

Fig. 10 compares the poses that the character interacts with the object on the COUCH dataset. We achieve more diverse results than the baselines.

7.5. Qualitative results on the NSM dataset.

Fig. 11 compares our method with SAMP and NSM on the NSM dataset. Fig. 12 compares the poses that the character interacts with the object on the NSM dataset. As the character in the dataset interacts with the object of the same category in a similar way, the baselines generate similar results. Our approach can generate more diverse results than the baselines.

7.6. Visualization of generated motions.

We provide a video to show more qualitative results on the NSM [19], SAMP [6], and COUCH [30] datasets. The comparisons with these baselines are also included in the video. We also demonstrate some case studies and failure cases in the video. To better evaluate the motion quality, we highly recommend readers watch the video.

8. Discussions

8.1. Online vs. offline.

Although it seems that online methods [19, 6, 30] can tackle most scenarios, our offline method achieves better performance in our setting. Offline methods [25, 24] are suitable for 3D content creation and the movie industry. However, offline methods cannot replace online methods as they cannot be applied to interactive applications like games. Furthermore, we assume the scene is static and our method cannot handle moving objects. The choice of different methods depends on the demand.

8.2. Case study.

Distance to the object. When the object is close to the character, the character would walk with small steps and sits on the object. However, it may generate unrealistic motions with severe foot sliding if the object is too far from the character as the dataset lacks such data.

Distance between milestones. We follow previous works [25, 24] to set the frame number between milestones as a static hyperparameter, and the arbitrary length generation is accomplished by the length prediction head. We find that the distances between generated milestones are not equal. If the distance between milestones is large, the character would walk faster. If the distance is small, the character would walk slower. The provided video also shows that the character's walking speed is not constant. However, if the predicted distance is too large, the model would generate motions with more sliding. Because our method predicts the length of milestones, this situation hardly ever happens unless the character is too far from the object which is out of the dataset distribution.

Starting actions. Although the training sequences in the dataset is that a character approaches the object, sits on it, and then leaves for the endpoint, we find our method and SAMP [6] can synthesize motions that start from sitting and sits on another object.

Other failure cases. Although we apply a transformer to learn the distribution of the codes of goal pose, our model may predict a wrong combination of codes. However, the overall performance is better than continuous methods and the continuous methods may also generate unrealistic poses.

8.3. Limitations.

As discussed above, our method cannot handle moving objects and the slow inference speed makes it hard for a real-time generation. The longest length of the generated motions is restricted by the dataset as the ground-truth data of the length prediction head is from the dataset. Furthermore, because the trajectory completion module and the motion infilling module do not consider the motion dependency between non-consecutive milestones, minor discontinuity exists. We try to eliminate this by interpolation. Specifically, for each subsequence, we predict additional

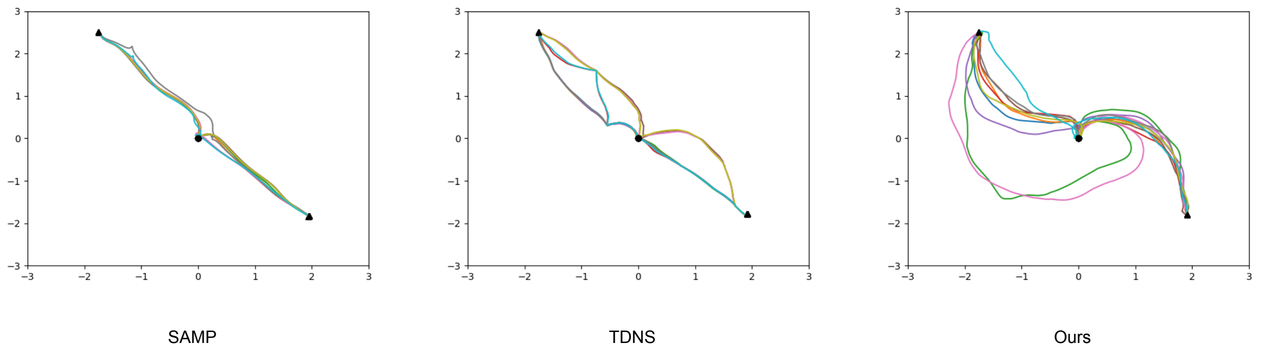


Figure 7. **The visualization of** 10 **generated trajectories when given the same starting point and the endpoint.** Compared with baseline methods [6, 24], our method generates more diverse trajectories. Triangles indicate starting point and the endpoint. Squares indicate the target object.

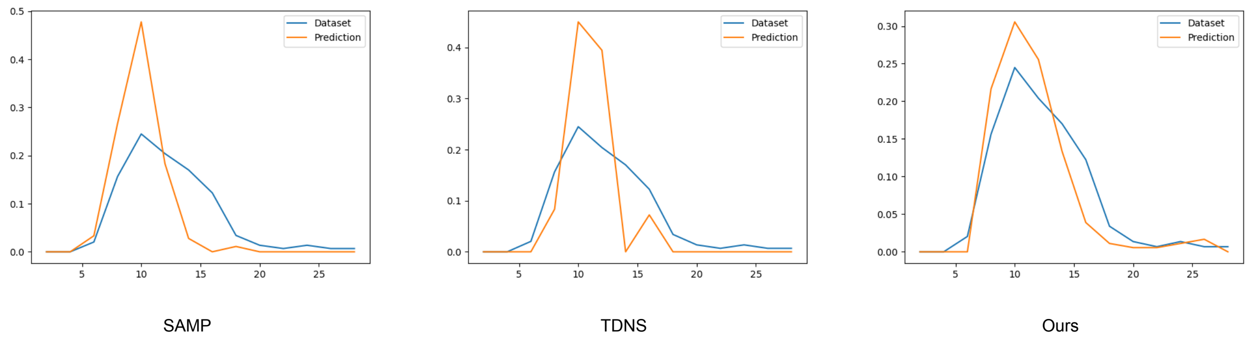


Figure 8. **The distribution of the length of the generated sequences.** We calculate the length of the motions that the character starts to walk and finishes the sitting action. Our results are more similar to the dataset distribution.

frames beyond the milestones. Then, we interpolate the two poses in the overlapped frames of two subsequences for smoother transitions. After processing, the FD decreases from 22.42 to 22.34, and the foot sliding increases from 0.49 to 0.50. We find this strategy shows better visual quality with only a little sacrifice of foot sliding. However, we can still observe some foot sliding in our results. Methods like [27] might mitigate this by simulating motions in physical engines with the guide of our generated kinematic results.

References

[1] Helmut Alt and MICHAEL GODAU. Computing the fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 03 1995. 4

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. 1

[3] Zhe Cao, Hang Gao, Karttikeya Mangalam, Qi-Zhi Cai, Minh Vo, and Jitendra Malik. Long-term human motion prediction with scene context. In *Computer Vision ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 387–404, Berlin, Heidelberg, 2020. Springer-Verlag. 4

[4] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883, June 2021. 1

[5] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 4, 5

[6] Mohamed Hassan, Duygu Ceylan, Ruben Villegas, Jun Saito, Jimei Yang, Yi Zhou, and Michael J. Black. Stochas-

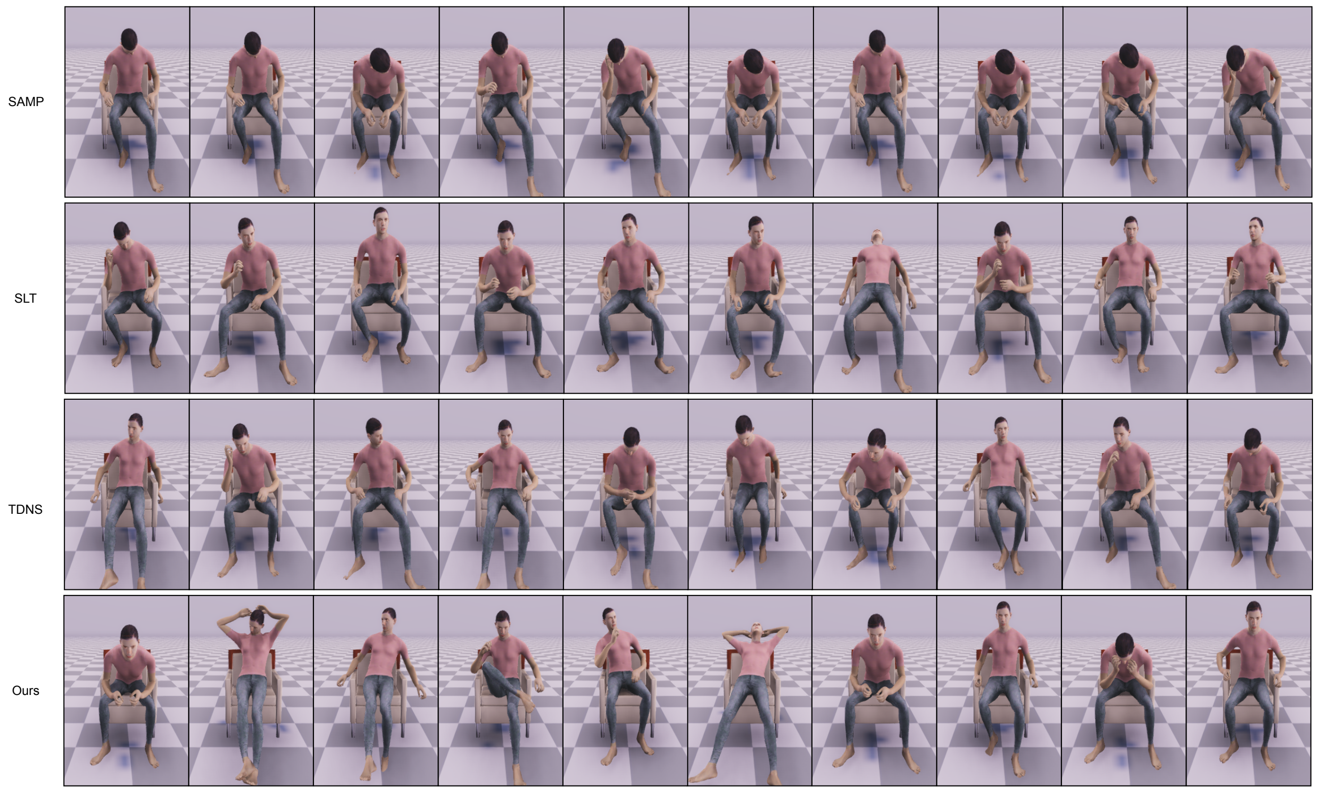


Figure 9. **Visualization of the poses that the character interacts with the object on the SAMP dataset.** We show 10 generated results that the character interacts with the same object. Our approach generates more diverse results than other methods.

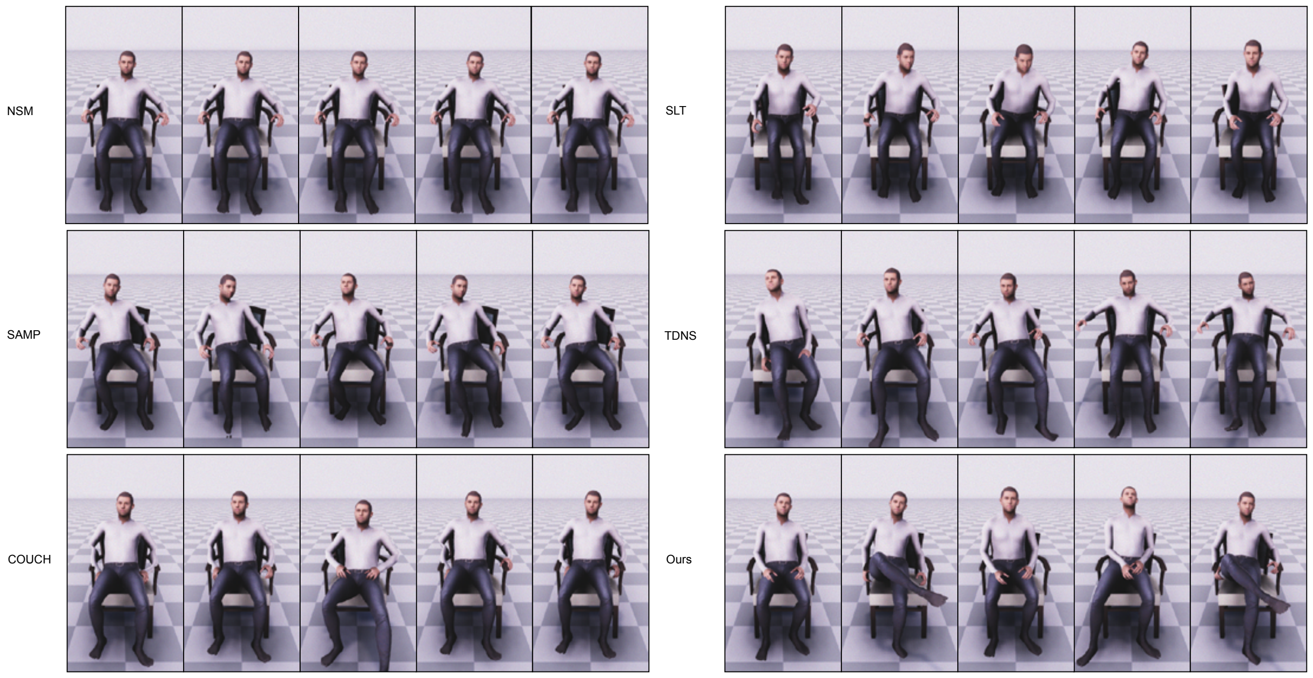


Figure 10. **Visualization of the poses that the character interacts with the object on the COUCH dataset.** We show 5 generated results that the character interacts with the same object. Our approach generates more diverse results than other methods.

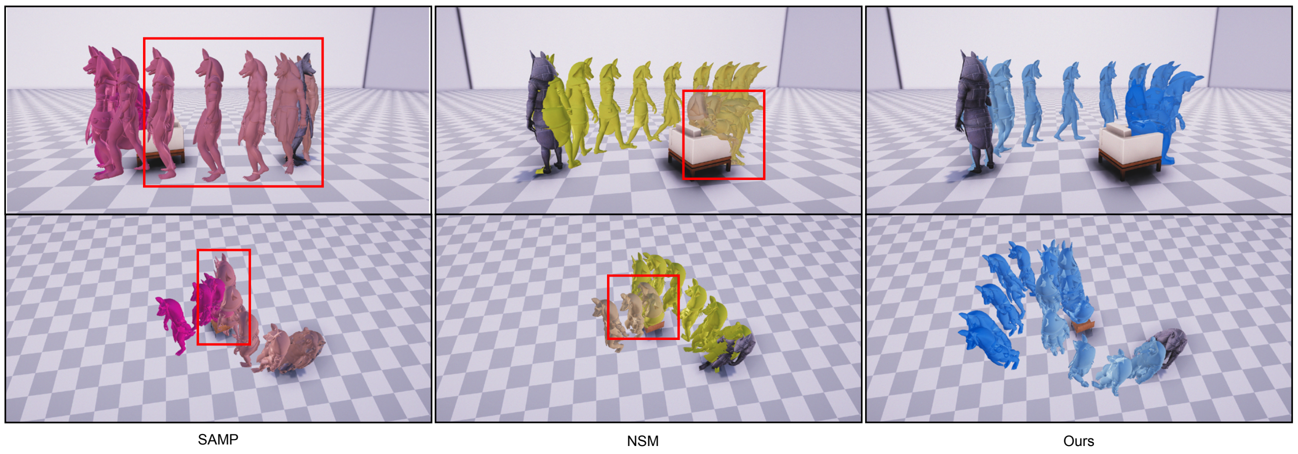


Figure 11. **Qualitative results on the NSM dataset.** We compare our method with the baseline SAMP [6] and NSM [19]. The failure cases are pointed out by the red rectangles. Specifically, the first row indicates that SAMP starts to walk with sliding and NSM has more penetration when the character is very close to the object. The second row shows that SAMP and NSM may stand up unnaturally. The human with clothes indicates the starting point. Darker color denotes later frames in the sequence.

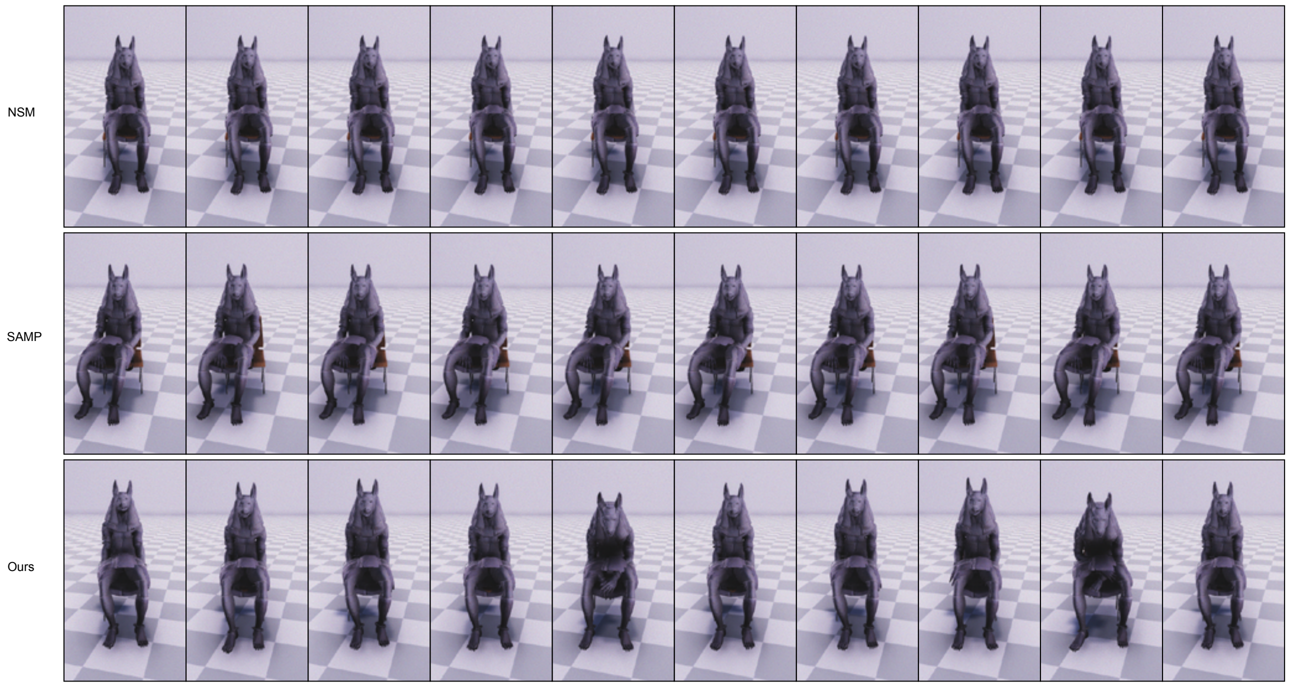


Figure 12. **Visualization of the poses that the character interacts with the object on the NSM dataset.** We show 10 generated results that the character interacts with the same object. Our approach generates more diverse results than other methods.

tic scene-aware motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11374–11384, October 2021. 2, 3, 4, 5, 6, 7, 8, 10

[7] Mohamed Hassan, Vasileios Choutas, Dimitrios Tzionas, and Michael J. Black. Resolving 3d human pose ambiguities with 3d scene constraints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3, 5

[8] Mohamed Hassan, Partha Ghosh, Joachim Tesch, Dimitrios Tzionas, and Michael J. Black. Populating 3d scenes by learning human-scene interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14708–14718, June 2021. 5

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference*

on Neural Information Processing Systems, NIPS'17, page 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. 3, 4

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. 1

[11] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Trans. Graph.*, 36(4), jul 2017. 4

[12] Deok-Kyeong Jang, Soomin Park, and Sung-Hee Lee. Motion puzzle: Arbitrary motion style transfer by body part. *ACM Trans. Graph.*, 41(3), jun 2022. 1

[13] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015. 4, 5

[14] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021. 1

[15] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4, 5

[16] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 1

[17] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), Nov. 2017. 4, 5

[18] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 1

[19] Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. Neural state machine for character-scene interactions. *ACM Trans. Graph.*, 38(6), nov 2019. 2, 3, 4, 5, 7, 10

[20] Omid Taheri, Vasileios Choutas, Michael J. Black, and Dimitrios Tzionas. Goal: Generating 4d whole-body motion for hand-object grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13263–13273, June 2022. 4

[21] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Amit H Bermano, and Daniel Cohen-Or. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022. 2

[22] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1

[24] Jingbo Wang, Yu Rong, Jingyuan Liu, Sijie Yan, Dahua Lin, and Bo Dai. Towards diverse and natural scene-aware 3d human motion synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20460–20469, June 2022. 3, 4, 5, 6, 7, 8

[25] Jiashun Wang, Huazhe Xu, Jingwei Xu, Sifei Liu, and Xiaolong Wang. Synthesizing long-term 3d human motion and interaction in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9401–9411, June 2021. 4, 5, 6, 7

[26] J. Wang, S. Yan, B. Dai, and D. Lin. Scene-aware generative network for human motion synthesis. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12201–12210, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. 4

[27] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. *arXiv preprint arXiv:2212.02500*, 2022. 8

[28] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *arXiv preprint arXiv:2208.15001*, 2022. 2

[29] S. Zhang, Y. Zhang, Q. Ma, M. J. Black, and S. Tang. Place: Proximity learning of articulation and contact in 3d environments. In *2020 International Conference on 3D Vision (3DV)*, pages 642–651, Los Alamitos, CA, USA, nov 2020. IEEE Computer Society. 4

[30] Xiaohan Zhang, Bharat Lal Bhatnagar, Sebastian Starke, Vladimir Guzov, and Gerard Pons-Moll. Couch: Towards controllable human-chair interactions. In *European Conference on Computer Vision (ECCV)*. Springer, October 2022. 2, 3, 4, 5, 7

[31] Yan Zhang, Mohamed Hassan, Heiko Neumann, Michael J. Black, and Siyu Tang. Generating 3d people in scenes without people. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4

[32] Yan Zhang and Siyu Tang. The wanderings of odysseus in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20481–20491, 2022. 4

[33] Yi Zhou, Connelly Barnes, Lu Jingwan, Yang Jimei, and Li Hao. On the continuity of rotation representations in neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3