

Avatars Grow Legs: Generating Smooth Human Motion from Sparse Tracking Inputs with Diffusion Model

Yuming Du* Robin Kips Albert Pumarola Sebastian Starke Ali Thabet Artsiom Sanakoyeu
Meta AI

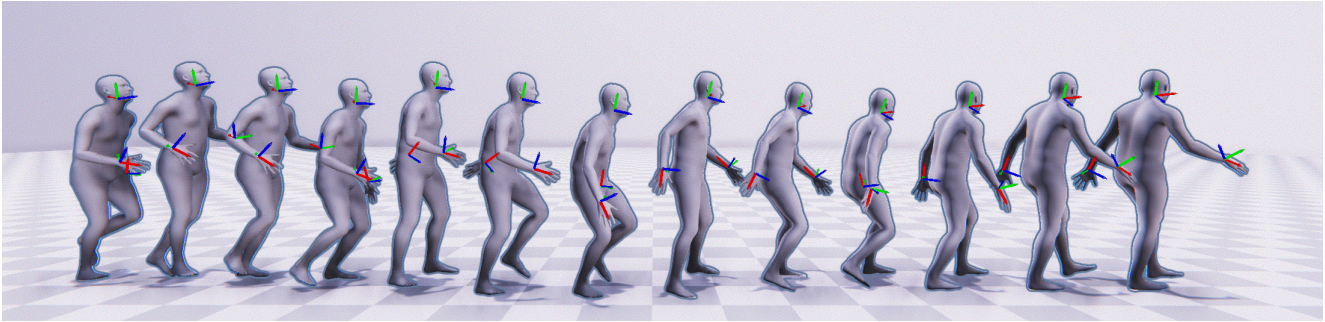


Figure 1. Full body motion synthesis based on HMD and hand controllers input. We show synthesis results of the proposed AGRoL method. RGB axes illustrate the orientation of the head and hands which serves as the input to our model.

Abstract

With the recent popularity spike of AR/VR applications, realistic and accurate control of 3D full-body avatars is a highly demanded feature. A particular challenge is that only a sparse tracking signal is available from standalone HMDs (Head Mounted Devices) and it is often limited to tracking the user’s head and wrist. While this signal is resourceful for reconstructing the upper body motion, the lower body is not tracked and must be synthesized from the limited information provided by the upper body joints. In this paper, we present AGRoL, a novel conditional diffusion model specially purposed to track full bodies given sparse upper-body tracking signals. Our model uses a simple multi-layer perceptrons (MLP) architecture and a novel conditioning scheme for motion data. It can predict accurate and smooth full-body motion, especially the challenging lower body movement. Contrary to common diffusion architectures, our compact architecture can run in real-time, making it usable for online body-tracking applications. We train and evaluate our model on AMASS motion capture dataset, and show that our approach outperforms state-of-the-art methods in generated motion accuracy and smoothness. We further justify our design choices through extensive experiments and ablations.

1. Introduction

Humans are the primary actors in AR/VR applications. As such, being able to track full-body movement is in high demand for these applications. Common approaches are able to accurately track upper bodies only [25, 58]. Moving to full-body tracking unlocks engaging experiences where users can interact with the virtual environment with an increased sense of presence. However, in the typical AR/VR setting there is no strong tracking signal for the entire human body – only the head and hands are usually tracked by means of Inertial Measurement Unit (IMU) sensors embedded in Head Mounted Displays (HMD) and hand controllers. Some works suggest adding additional IMUs to track the lower body joints [22, 25], those additions come at higher costs and the expense of the user’s comfort [24, 27]. In an ideal setting, we want to enable high-fidelity full-body tracking using the standard three inputs (head and hands) provided by most HMDs.

Given the position and orientation information of the head and both hands, predicting full-body pose, especially the lower body, is inherently an underconstrained problem. To address this challenge, different methods rely on generative models such as normalizing flows [46] and Variational Autoencoders (VAE) [11] to synthesize lower body motions. In the realm of generative models, diffusion models have recently shown impressive results in image and video generation [21, 40, 49], especially for conditional generation. This inspires us to employ the diffusion model to generate the fully-body poses conditioned on the sparse track-

*Work done during an internship at Meta AI.

ing signals. To the best of our knowledge, there is no existing work leveraging the diffusion model solely for motion reconstruction from sparse tracking information.

However, it is not trivial to employ the diffusion model in this task. Existing approaches for conditional generation with diffusion models are widely used for cross-modal conditional generation. Unfortunately, these methods can not be directly applied to the task of motion synthesis, given the disparity in data representations, *e.g.* human body joints feature *vs.* images.

In this paper, we propose a novel diffusion architecture – *Avatars Grow Legs* (AGRoL), which is specifically tailored for the task of conditional motion synthesis. Inspired by recent work in future motion prediction [18], which uses a simple MLP-based architecture, we find that a carefully designed MLP network can already achieve comparable performance to the state-of-the-art methods. However, we discovered that the predicted motions of MLP networks may contain jittering artifacts. To solve this problem and generate smooth realistic full body motion from sparse tracking signals, we design a novel lightweight diffusion model empowered by our MLP architecture. Diffusion models require time step embedding [21, 39] to be injected in the network during training and inference, while we found that our MLP architecture is not sensitive to the positional embedding in the input. To solve this issue, we further propose a novel strategy to effectively inject the time step embedding during the diffusion process. With the proposed strategy, we can significantly mitigate the jittering issues and further improve the model performance and its robustness to the loss of tracking signal. Our model accurately predicts full-body motions outperforming state-of-the-art approaches as we demonstrate by the experiments on AMASS [36], large motion capture dataset.

We summarize our contributions as follows:

- We propose AGRoL, a conditional diffusion model specifically designed for full-body motion synthesis based on sparse IMU tracking signals. AGRoL is a simple and yet efficient MLP-based diffusion model with a lightweight architecture. To enable gradual denoising and produce smooth motion sequences we propose a block-wise injection scheme that adds diffusion timestep embedding before every intermediate block of the neural network. With this timestep embedding strategy, AGRoL achieves state-of-the-art performance on the full-body motion synthesis task without any extra losses that are commonly used in other motion prediction methods.
- We show that our lightweight diffusion-based model AGRoL can generate realistic smooth motions while achieving real-time inference speed, making it suitable for online applications. Moreover, it is more robust

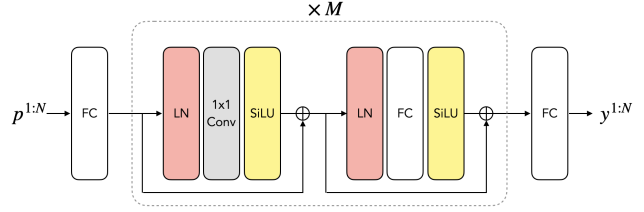


Figure 2. The architecture of our MLP-based network. *FC*, *LN*, and *SiLU* denote the fully connected layer, the layer normalization, and the SiLU activation layer respectively. 1×1 Conv denotes the 1D convolution layer with kernel size 1. Note that 1×1 Conv here is equivalent to a fully connected layer operating on the first dimension of the input tensor $\mathbb{R}^{N \times \hat{D}}$, while the *FC* layers operate on the last dimension. N denotes the temporal dimension and \hat{D} denotes the dimension of the latent space. The middle block is repeated M times. The first *FC* layer projects input data to a latent space $\mathbb{R}^{N \times \hat{D}}$ and the last one converts from latent space to the output space of full-body poses $\mathbb{R}^{N \times S}$.

against tracking signals loss than existing approaches.

2. Related Work

2.1. Motion Tracking from Sparse Tracking Inputs

Generating full-body poses from sparse tracking signals of body joints has recently drawn a lot of interest in the research community. Works like [22] track full bodies given 6 IMU inputs and using a bi-directional LSTM to predict SMPL body joints. In a similar approach, the method in [58] tracks with 4 IMU inputs (head, wrists, and pelvis). However, the more practical case in HMD settings is to have access to only 3 tracking signals: head and 2 wrists. To that extent, AvatarPoser [24] solves the 3-point problem using a transformer-based architecture. Other recent methods attempt to solve sparse input body tracking as a synthesis problem. To that extent, Aliakbarian *et al.* [4] proposed a flow-based architecture derived from [10], while Dittadi *et al.* [11] opted for a Variational Autoencoder (VAE) method. Finally, more complex methods resort to Reinforcement Learning [57], but require future frames as input, introducing latency in the tracking system.

In summary, all methods presented in this section either rely on more than 3 inputs or struggle to predict full body pose, in particular lower bodies. In contrast, our method leverages a custom diffusion model, using a simple MLP-based architecture, to accurately predict full body pose using only 3 IMU inputs.

2.2. Diffusion Models and Motion Synthesis

Diffusion models [21, 40, 49] are a class of likelihood-based generative models based on learning progressive noising and denoising of data. Diffusion models have recently drawn a large amount of interest in image genera-

tion [9], due to their ability to significantly outperform popular GAN architectures [7, 26]. Conditional generation is also possible with diffusion models, as seen in the classifier guidance proposed in [9], or the CLIP-based text conditional synthesis for diffusion models presented in [39].

More recently, concurrent works have also extended diffusion models to motion synthesis, with particular focus on the text-to-motion task [28, 52, 60]. However, these models are both complex in architecture and require multiple iterations at inference time. This hinders them unusable for real-time applications like VR body tracking. We circumvent this problem by designing a custom and efficient diffusion model. To the best of our knowledge, we present the first diffusion model solely purposed for solving motion reconstruction from sparse inputs. Our model leverages a simple MLP architecture, runs in real-time, and provides accurate pose predictions, particularly for lower bodies.

2.3. Human Motion Synthesis

Early works in human motion synthesis rose under the task of future motion prediction. Works around this task saw various modeling approaches ranging from sequence to sequence models [14], all the way to graph modeling of each body part [23]. These supervised models were later replaced by generative methods [17, 31] based on Generative Adversarial Networks (GANs) [16]. Despite their leap forward, these approaches tend to diverge from realistic motion and require access to all body joint positions, making them impractical for avatar animation in VR [19].

A second family of motion synthesis methods revolves around character control. In this setting, character motion must be generated according to user inputs and environmental constraints, such as the virtual environment properties. This line of work is lucrative for computer gaming, where controller input is used as guidance for character motion. Inspired by these constraints, the work in [56] formulates motion synthesis as a control problem given direction and speed input, using a GAN architecture. Similar efforts are found in [51], where the method learns fast and dynamic character interactions that involve contacts between the body and other objects, given user input from a controller. These methods are impractical in a VR setting, where users want to drive motion using their real body pose instead of a controller.

3. Method

3.1. Problem Formulation

Our goal is to predict the whole body motion given sparse tracking signals, i.e. the orientation and translation of the headset and two hand controllers. Given a sequence of N observed joint features $p^{1:N} = \{p^i\}_{i=1}^N \in \mathbb{R}^{N \times C}$, we aim to predict the whole body poses for the N frames

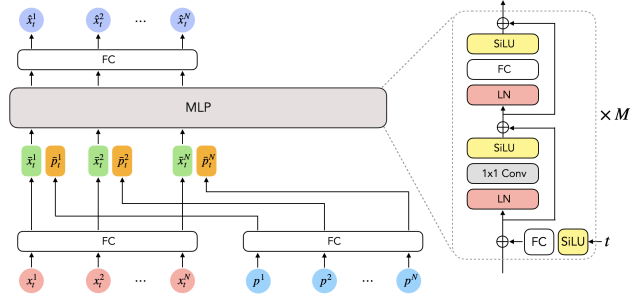


Figure 3. The architecture of our MLP-based diffusion model. t is the noising step. $x_t^{1:N}$ denotes the motion sequence of length N at step t , which is pure Gaussian noises when $t = 0$. $p^{1:N}$ denotes the sparse upper body signals of length N . $\hat{x}_t^{1:N}$ denotes the denoised motion sequence at step t .

$y^{1:N} = \{y^i\}_{i=1}^N \in \mathbb{R}^{N \times S}$, where C and S represent the dimension of the input/output joint features. In this paper, we adopt SMPL [34] model to represent the human poses and follow the setting of [11, 24] to only use the first 22 joints of SMPL model and ignore the joints on the hands. Thus, $y^{1:N}$ represents the global orientation of the pelvis and the relative rotation of each joint.

In the following section, we first introduce a simple MLP-based network for full-body motion synthesis based on sparse tracking signals. Then, we show how we further improve the performance by leveraging the proposed MLP-based architecture to power the conditional generative diffusion model, termed AGRoL.

3.2. MLP-based Network

Our network is composed of only 4 types of components widely used in the deep learning era: fully connected layer (LN), SiLU [43] activation layer, 1D convolutional layer [30] with kernel size 1 and layer normalization [5]. Note that the 1D convolutional layer with kernel size 1 can be also seen as a fully connected layer operating on a different dimension. The details of our network architecture are demonstrated in Figure 2. Each block of the MLP network contains one convolutional and one fully connected layer, which is responsible for temporal and spatial information merging respectively. We use skip-connections as in ResNets [20] with Layer Norm [6] as pre-normalization of the layers. First, we project the input data $p^{1:N}$ to a higher dimensional latent space using a linear layer. And the last layer of the network projects from the latent space to the output space of full-body poses $y^{1:N}$.

3.3. Diffusion Model

Diffusion model [21, 49] is a type of generative model which learns to reverse random Gaussian noise added by a Markov chain in order to recover desired data samples from the noise. In the forward diffusion process, given a sample

motion sequence $x_0^{1:N} \sim q(x_0^{1:N})$ from the data distribution, the Markovian noising process can be written as:

$$q(x_t^{1:N} | x_{t-1}^{1:N}) := \mathcal{N}(x_t^{1:N}; \sqrt{\alpha_t} x_{t-1}^{1:N}, (1 - \alpha_t)I), \quad (1)$$

where $\alpha_t \in (0, 1)$ is constant hyper-parameter and I is the identity matrix. $x_t^{1:N}$ tend to an isotropic Gaussian distribution when $t \rightarrow \infty$. Then, in the reverse diffusion process, a model p_θ with parameters θ is trained to generate samples from a Gaussian noise input $x_T \sim \mathcal{N}(0, I)$ with a fixed variance σ_t^2 . Formally,

$$p_\theta(x_{t-1}^{1:N} | x_t^{1:N}) := \mathcal{N}(x_{t-1}^{1:N}; \mu_\theta(x_t, t), \sigma_t^2 I), \quad (2)$$

where μ_θ could be reformulated as [21],

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{1 - \bar{\alpha}_t} \epsilon_\theta(x_t, t) \right), \quad (3)$$

where $\bar{\alpha}_t = \alpha_1 \dots \alpha_t$. So the model has to learn to predict noise $\epsilon_\theta(x_t, t)$ from x_t and timestep t .

In our case, we want to use the diffusion model to generate sequences of full-body poses conditioned on the sparse tracking of joint features $p^{1:N}$. Thus, the reverse diffusion process becomes conditional: $p_\theta(x_{t-1}^{1:N} | x_t^{1:N}, p^{1:N})$. Moreover, we follow [44] to directly predict the clean body poses $\hat{x}_0^{1:N}$ instead of predicting the residual noise $\epsilon_\theta(x_t, t)$. We denote by $\hat{x}_0^{1:N} := f_\theta(x^{1:N}, p^{1:N}, t)$ the output of our model f_θ . The objective function is then formulated as

$$\mathcal{L}_{dm} = \mathbb{E}_{x_0^{1:N} \sim q(x_0^{1:N}), t} [\| x_0^{1:N} - f_\theta(x^{1:N}, p^{1:N}, t) \|_2^2] \quad (4)$$

We use the MLP network that we proposed in Section 3.2 as our model f_θ that predicts the full-body poses. At time step t , the motion features $x_t^{1:N}$ and the observed joints feature $p^{1:N}$ are first passed separately through a fully connected layer to obtain the intermediate features $\bar{x}_t^{1:N}$ and $\bar{p}^{1:N}$.

$$\bar{x}_t^{1:N} = \text{FC}_0(x_t^{1:N}) \quad (5)$$

$$\bar{p}^{1:N} = \text{FC}_1(p^{1:N}) \quad (6)$$

Then these features are concatenated together and fed to the MLP network.

$$\hat{x}_0^{1:N} = f_\theta(\text{Concat}(\bar{x}_t^{1:N}, \bar{p}^{1:N}), t) \quad (7)$$

Block-wise Timestep Embedding. In diffusion models, embedding of the timestep t is usually fed to a network as an extra input. A common way to add time step embedding is to concatenate it with the input, similarly to positional embedding used in transformer-based methods [12, 55]. However, as our network uses MLPs, we find that our model is not very sensitive to the values of the timestep embedding

which hinders learning the denoising process and results in predicting motions with severe jittering issues, as shown in Section 4. To solve this problem, we propose a novel strategy that repetitively injects the time step embedding before every block of the MLP network. The details of our pipeline are shown in Figure 3. The timestep embedding is projected to match the input feature dimensions through a fully connected layer and a SiLU activation layer, then, differently from [21], which predicts a scale and a shift factor from the time step embedding for each block, we directly add the obtained feature to the input intermediate activations. As we show in Sect. 4, the proposed strategy can largely mitigate the jittering problem and enables synthesis of smooth motions.

4. Experiments

We train and evaluate our models on AMASS [36] dataset. We use two settings for training and testing to compare with previous methods. For the first setting, we follow [24] and use three subsets CMU [8], BMLr [53], and HDM05 [38]. For the second setting, we follow the data split in [4, 11, 45], which uses CMU [8], MPI Limits [3], Total Capture [54], Eyes Japn [13], KIT [37], BioMotion-Lab [53], BMLMovi [15], EKUT [37], ACCAD [1], MPI Mosh [33] SFU [2] and HDM05 [38] as training data, and HumanEval [48] and Transition [36] as testing data. In both settings, we adopt the SMPL [34] human model for the human pose representation and train our model to predict the global orientation of the root joint and relative rotation of the other joints.

4.1. Implementation details

We represent the joint rotations by the 6D reparametrization [61] due to its simplicity and continuity. Thus, for the sequences of body poses $y^{1:N} \in \mathbb{R}^{N \times S}$, $S = 22 \times 6$. And we set the frame number $N = 196$ if not stated otherwise.

MLP Network We build our MLP network using 12 blocks ($M = 12$). All latent features in the MLP network have the same shape of $N \times 512$. The network is trained with batch size 256 and Adam optimizer [29]. The learning rate is set to $3e-4$ at the beginning and drops to $1e-5$ after 200000 iterations. The weight decay is set to $1e-4$ for the entire training. During inference, we apply our model in an auto-regressive manner for the longer sequences.

MLP-based Diffusion Model (AGRoL) We keep the MLP network architecture unchanged in the diffusion model. To inject the time step embedding used in the diffusion process in the network, in each MLP block, we pass the time step embedding to a fully connected layer and a SiLU activation layer [43] and add it with the input feature. The network is trained with exactly the same hyperparameters as the MLP network, with the exception of using the

Method	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Root RE	Jitter	Upper Jitter	Lower Jitter
Final IK	16.77	18.09	59.24	-	-	-	-	-	-	-	-
LoBStR	10.69	9.02	44.97	-	-	-	-	-	-	-	-
VAE-HMD	4.11	6.83	37.99	-	-	-	-	-	-	-	-
AvatarPoser*	3.08	4.18	27.70	2.12	1.81	7.59	3.34	19.68	14.49	7.36	24.81
MLP (Ours)	2.69	3.93	<u>22.85</u>	2.62	1.89	<u>6.88</u>	<u>3.35</u>	19.61	13.01	9.13	<u>18.61</u>
AGRoL (Ours)	2.66	3.71	18.59	1.31	1.55	6.84	3.36	<u>19.62</u>	7.26	5.88	9.27
GT	0	0	0	0	0	0	0	0	4.00	3.65	4.52

Table 1. Comparison of our approach with state-of-the-art methods on a subset of AMASS dataset following [24]. We report *MPJPE* [cm], *MPJRE* [deg], *MPJVE* [cm/s], Jitter [10^2m/s^3] metrics. AGRoL achieves the best performance on *MPJPE*, *MPJRE* and *MPJVE*, and outperforms other models, especially on the *Lower PE* (Lower body Position Error) and *Jitter* metrics, which shows that our model generates accurate lower body movement and smooth motions.

Method	MPJRE	MPJPE	MPJVE	Root RE	Jitter
VAE-HMD [†] [11]	-	7.45	-	-	-
HUMOR [†] [45]	-	5.50	-	-	-
FLAG [†] [4]	-	4.96	-	-	-
AvatarPoser*	4.70	<u>6.38</u>	34.05	33.72	<u>10.21</u>
MLP (Ours)	<u>4.33</u>	6.66	<u>33.87</u>	<u>33.58</u>	21.74
AGRoL (Ours)	4.30	6.17	24.40	33.33	8.32
GT	0	0	0	0	2.93

Table 2. Comparison of our approach with state-of-the-art methods on AMASS dataset following the protocol of [4, 11, 45]. We report the MPJPE [cm], MPJRE [deg], MPJVE [cm/s], and Jitter [10^2m/s^3] metrics. The * denotes that we retrained the AvatarPoser using public code. † denotes methods that use pelvis location and rotation during inference, which are not directly comparable to our method, as we assume that the pelvis information is not available during the training and the testing. The best results are in bold, and the second-best results are underlined.

AdamW [35] as optimizer. The sampling step is set to 1000 with a cosine noise schedule [40] during the training. To accelerate the inference speed, instead of doing 1000 sampling steps during the inference. We use the DDIM [50] technique to speed up the sampling and only sample 5 steps during the inference.

All experiments are conducted with Pytorch framework [41] on a single NVIDIA V100 graphics card.

4.2. Evaluation Metrics

We use in total 10 metrics to evaluate our methods following previous works [11, 24, 45, 59]. The metrics can be divided into three types. The first type is rotation-related metrics, which includes the *MPJRE* (Mean Per Joint Rotation Error [degrees]) and *Root RE* (Root Rotation Error [degrees]), which measure the average relative rotation error of all joints and the rotation error of the root joint. The second type is velocity-related metrics including *MPJVE* (Mean Per Joint Velocity Error[cm/s]) and *Jitter*, *MPJVE* (Mean Per Joint Velocity Error[cm/s]) measures the average velocity error of all joints, *Jitter* measures the mean jerk (time derivative of acceleration) of all body joints in the global space in 10^2m/s [59], which reflects the smoothness of the motion. The third type is position-related metrics, which in-

cludes all the rest metrics. Specifically, *MPJPE* (Mean Per Joint Position Error[cm]) measures the average position error of all joints. *Root PE* evaluates the position error of the root. *Hand PE* measures the average position error for the two hands. *Upper PE* and *Lower PE* evaluate the average position error for joints in the upper body and lower body respectively.

4.3. Evaluation Results

We evaluate our method on AMASS dataset with two different protocols. As shown in Table 1 and Table 2, our MLP network solely can surpass most previous methods and achieves comparable results with the state-of-the-art method [24], which shows the effectiveness of our proposed simple network. With the help of the diffusion process, our AGRoL model further improves the performance of the MLP network and surpasses all previous methods. Moreover, the proposed AGRoL model significantly reduces the *Jitter* error, which means our generated motion is much smoother compared to the others. We visualize some examples in Figure 4 and Figure 5. In Figure 4 we show the comparison of the reconstruction error between AGRoL and AvatarPoser. In Figure 5, by visualizing the pose trajectories, we show the comparison of the smoothness between AGRoL and AvatarPoser, furthermore, the foot sliding issues can also be viewed by

4.4. Ablation Studies

In this section, we ablate our methods on AMASS dataset. We first compare our proposed MLP network with other networks in the case of the diffusion model in Section 4.4.1 to show the effectiveness of the MLP network. Then we ablate the importance of time step embedding for our diffusion model and ablate different strategies to add time step embedding in Section 4.4.2. We study the influence of extra losses in Section 4.4.3 and the number of sampling steps we used during the inference in Section 4.4.4.

4.4.1 Architecture

To validate the effectiveness of our proposed MLP network in a diffusion model setup, we replace our MLP network with other types of networks and compare their re-

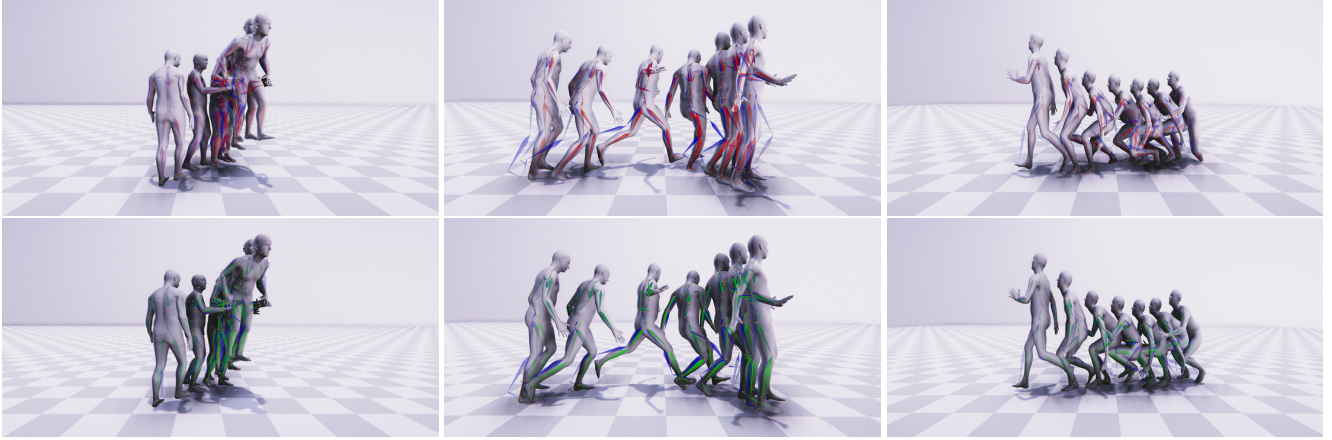


Figure 4. Qualitative comparison between AGRoL (lower) and AvatarPoser (upper) on test sequences from AMASS dataset. We visualize the predicted skeleton and human body meshes in the figures. The skeletons in **green** denote the motion predicted using our method. The skeletons in **red** denote the motion predicted using AvatarPoser [24]. The skeletons in **blue** denote the ground truth motion. As shown in the figure, our predicted motion is more accurate compared to the predicted motion of AvatarPoser.

Method	#Params	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Root RE	Jitter
AvatarPoser	2.89M	4.31	6.71	27.65	1.47	2.56	12.69	6.69	21.36	9.57
AvatarPoser-Large	7.63M	<u>2.86</u>	<u>4.04</u>	21.90	1.29	<u>1.62</u>	<u>7.53</u>	<u>3.64</u>	19.75	9.94
Transformer [52]	7.03M	3.01	4.41	<u>20.33</u>	2.97	2.13	7.71	3.88	<u>19.70</u>	6.45
AGRoL (Ours) - pred noise	7.51M	5.41	8.88	28.67	4.38	3.91	16.06	8.76	22.78	9.80
AGRoL (Ours)	7.51M	2.66	3.71	18.59	<u>1.31</u>	1.55	6.84	3.36	19.62	<u>7.26</u>

Table 3. Ablation study of network architectures in our diffusion model. We replace our MLP network with other networks and train them in the diffusion model fashion with the same hyperparameters. Our MLP network outperforms all other networks on most of the metrics. The *AvatarPoser-Large* denotes the network with the same architecture as AvatarPoser [24] but with more transformer layers. The best results are in bold, and the second-best results are underlined.

sults. We mainly consider two architectures, the network from AvatarPoser [24] and the transformer network [55]. In the case of transformer networks, instead of repetitively injecting the time positional embedding to every block, we concatenate the time positional embedding with the input features $\bar{x}^{1:N}$ and $\bar{p}^{1:N}$ before being fed to the transformer layers. We apply the same strategy to the network of AvatarPoser as this model also uses transformer blocks in the early stage. To establish a fair comparison to the AvatarPoser architecture, we train two versions of this model, one uses the same setting as in the original paper, and the other with more transformer layers to obtain a comparable size to our proposed model. We also perform the same experiment for the transformer network. As shown in Table 3, our proposed MLP network achieves superior results compared to these networks when trained in the diffusion fashion.

4.4.2 Diffusion Time Step Embedding

In this section, we study the importance of time step embedding. Time step embedding is often used in diffusion-based models [9, 60] to indicate the noising step t during the diffusion process. We use the sinusoidal position embedding [55] as the time step embedding. We first show the

results of AGRoL without time step embedding in Table 4, the network can still achieve decent performance on metrics related to position errors and rotation errors, while the performance on metrics related to velocity errors (*MPJVE* and *Jitter*) are severely degraded, which is reasonable, as the time step embedding is missing, the model does not know which step it locates thus can not denoise properly.

We then ablate three strategies for applying the time step embedding in our network: *Add*, *Concat*, and *RepIn*. In contrast to the *RepIn* (Repetitive Injection), which repetitively passes the time step embedding through a linear layer and injects them into every block of the MLP network, in *Add* and *Concat*, the time step embedding is only used once at the beginning of the network. Here, the time step embedding is first passed through a fully connected layer and a SiLU activation layer to obtain a latent feature $u \in \mathbb{R}^{1 \times K}$ before being fed to the network. In particular, *Add* sums up the u and the input features $\bar{x}^{1:N}$ and $\bar{p}^{1:N}$, the output of the network is therefore $\hat{x}_t^i = f_\theta(\text{Concat}(\bar{x}^i, \bar{p}^i) + u)$, $i \in [1, \dots, N]$. *Concat* concatenates the u with the input features $\bar{x}^{1:N}$ and $\bar{p}^{1:N}$, thus, the output of the network is $\hat{x}_t^{1:N} = f_\theta(\text{Concat}(\bar{x}^{1:N}, \bar{p}^{1:N}, u))$. *RepIn* represents our strategy for adding the time step embedding. Specifically,

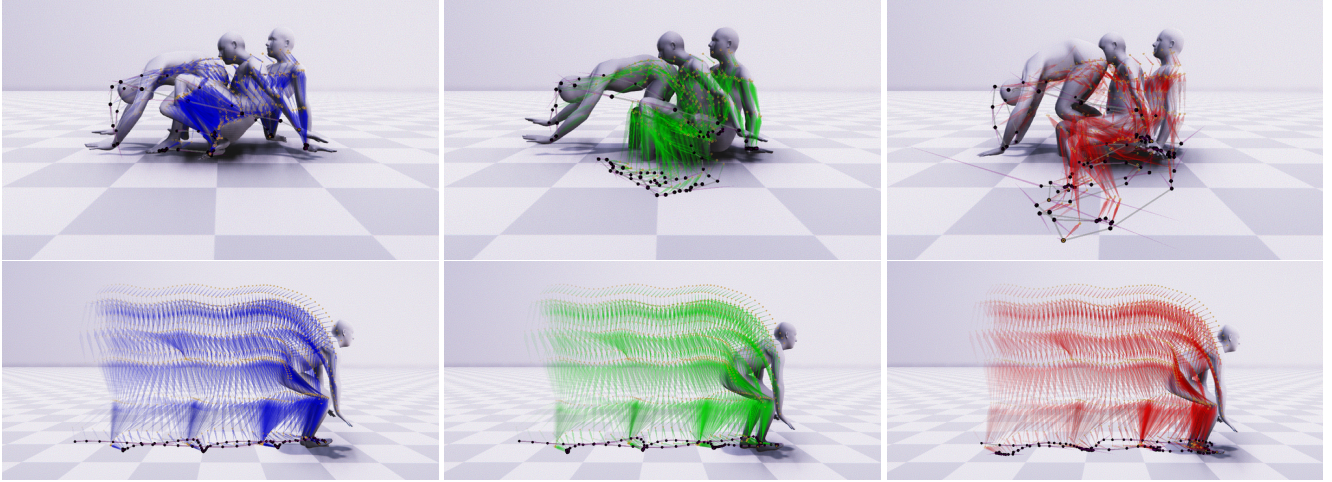


Figure 5. Motion trajectory visualization. We visualize the trajectories of the predicted motion in the figures. The images on the left show the ground truth motion with blue skeletons. The images in the middle show the predicted motion of AGRoL with green skeletons. The images on the right show the predicted motion of AvatarPoser with red skeletons. The light purple vectors in the figures denote the velocity vector of each joint. By visualizing the trajectories of the motion, the jittering issues and foot sliding issues can be better viewed from the figures. Smooth motion tends to have regular pose trajectories with the velocity vector of each joint changing steadily. The density of the pose trajectories will change along with the walking speed, the trajectories will become denser when the person slows down. Thus, if there is no foot sliding, we should occasionally see the change of density in pose trajectories.

Method	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Root RE	Jitter
w/o Time	<u>2.68</u>	3.63	22.80	1.36	1.54	6.67	3.25	19.38	15.23
Add	2.80	4.01	23.60	1.40	1.64	7.44	3.59	19.61	15.02
Concat	2.72	3.79	21.99	1.31	1.57	7.00	3.43	19.66	13.30
RepIn (Ours)	2.66	<u>3.71</u>	18.59	1.31	<u>1.55</u>	<u>6.84</u>	<u>3.36</u>	19.62	7.26

Table 4. Ablation of the time step embedding. *w/o Time* denotes the results of AGRoL without time step embedding. *Add* sums up the features from time step embedding with the input features. *Concat* concatenates the features from time step embedding with the input features. In *Add* and *Concat*, the time step embedding is only fed once at the top of the network. *RepIn* (Repetitive Injection) denotes our strategy to inject the time step embedding into every block of the network. As shown in the table, the time step embedding mainly affects the *MPJVE* and *Jitter* metrics. Without time step embedding or adding the time step embedding improperly will result in high errors for velocity-related metrics, causing severe jittering issues.

for each block of the MLP network, we project the time step embedding separately through a fully connected layer and a SiLU activation layer, then we add the obtained feature $u_j, j \in [0, \dots, M]$ to the input features of its correspondent block. As shown in Table 4, our proposed strategy can largely improve the velocity-related metrics and alleviate the jittering issues and generate smooth motion.

4.4.3 Additional Losses

In addition to \mathcal{L}_{dm} , we explore three other geometric losses during the training like previous works [42, 47]:

$$\mathcal{L}_{pos} = \frac{1}{N} \sum_{i=1}^N \| \text{FK}(y_0^i) - \text{FK}(\hat{x}_0^i) \|_2^2 \quad (8)$$

$$\mathcal{L}_{vel} = \frac{1}{N-1} \sum_{i=1}^{N-1} \| (\text{FK}(y_0^{i+1}) - \text{FK}(y_0^i)) - (\text{FK}(\hat{x}_0^{i+1}) - \text{FK}(\hat{x}_0^i)) \|_2^2 \quad (9)$$

$$\mathcal{L}_{foot} = \frac{1}{N-1} \sum_{i=1}^{N-1} \| (\text{FK}(y_0^i) - \text{FK}(\hat{x}_0^i)) \cdot m_i \|_2^2, \quad (10)$$

where $\text{FK}(\cdot)$ is the forward kinematics function which takes local human joint rotations as input and outputs these joint positions in the global coordinate space. \mathcal{L}_{pos} represents the position loss of joints, \mathcal{L}_{vel} represents the velocity loss of the joints in 3D space and \mathcal{L}_{foot} represents the foot contact loss, which enforces static feet when there is no feet movement. $m_i \in \{0, 1\}$ denotes the binary mask and equals to 0 when the feet joints have zero velocity.

We train our model with different combinations of extra losses, setting their weights equal to 1. As shown in Table 5, in contrast to previous works [24], the extra geometric losses do not bring additional performance to our diffusion model. Our model can achieve good results when

\mathcal{L}_{pos}	\mathcal{L}_{vel}	\mathcal{L}_{foot}	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Root RE	Jitter
			2.66	3.71	18.59	1.31	1.55	6.84	3.36	19.62	7.26
		✓	2.83	4.07	20.66	1.58	1.70	7.49	3.66	19.74	9.20
✓			2.81	4.06	21.85	1.75	1.73	7.43	3.72	19.77	12.16
✓	✓		2.73	3.92	20.55	1.72	1.68	7.15	3.52	19.59	10.16
✓	✓	✓	2.89	4.16	20.58	1.73	1.76	7.63	3.81	19.85	8.98

Table 5. Ablation of the additional losses used during training.

# Sampling Steps	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Root RE	Jitter
2	3.17	4.93	20.03	2.19	2.12	8.98	4.61	20.29	6.90
5	2.66	3.71	18.59	1.31	1.55	6.84	3.36	19.62	7.26
10	<u>2.68</u>	3.69	<u>19.55</u>	<u>1.39</u>	1.55	6.77	3.31	19.61	7.51
100	2.84	3.93	23.50	1.62	1.67	7.19	3.51	19.84	9.64
1000	2.97	4.14	27.25	1.82	1.78	7.55	3.66	20.10	12.79

Table 6. Ablation of the # sampling steps during inference. The input and output length is fixed to $N = 196$.

Methods	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Root RE	Jitter
AvatarPoser	5.69	10.34	572.58	8.98	5.49	17.34	8.83	27.27	762.79
MLP	5.37	10.76	107.82	12.43	6.48	16.94	8.74	25.38	92.51
Transformer	4.44	8.62	135.99	7.29	5.28	13.44	10.32	21.37	147.09
AGRoL(Ours)	4.20	6.38	96.85	5.27	3.86	10.03	6.67	20.90	33.35

Table 7. Robustness of the models to joints tracking loss. We evaluate the methods by randomly masking a portion(10%) of input frames during the inference on AMASS dataset. We test each method 5 times and take the average results. AGRoL achieves the best performance among all the methods, which shows the robustness of our method against joint tracking loss.

trained solely with the denoising objective function Eq.(4). We believe the reason that extra losses do not improve the performance of AGRoL is due to the inner working of the reverse diffusion process, which does not interplay with extra geometrical losses without proper tuning.

4.4.4 Number of Sampling Steps during Inference

In Table 6 we ablate the number of sampling steps that we used during the inference. We use the same model trained with 1000 sampling steps and test with a subset of steps in the diffusion process. We use 5 DDIM sampling steps, allowing our model to achieve superior performance on most of the metrics while being fast.

4.5. Robustness to Tracking Loss

In this section, we study the robustness of our model against tracking loss of the input joints. In practice, it is a common problem in VR applications that the joint tracking signal is lost on some frames, due to hands or controllers going out of the field of view, creating temporal discontinuity in the inputs. We evaluate the performance of all available methods on tracking loss by randomly masking 10% of input frames during the inference. The results are shown in Table 7. The performance of all current methods is largely degraded, which indicates that they are not robust against the tracking loss problem. In comparison, our network accuracy is less degraded which tends to indicate that our network can accurately model motion given highly sparse tracking inputs.

4.6. Inference Speed

Our AGRoL model achieves real-time inference speed due to a lightweight architecture combined with DDIM sampling. A single AGRoL generation, that runs 5 DDIM sampling steps, produces 196 output frames in 35ms on a single NVIDIA V100 GPU. Our predictive MLP model takes 196 frames as input and predicts a final result of 196 frames in a single forward pass. It is even faster and requires only 6ms on a single NVIDIA V100 GPU.

5. Conclusion and Limitations

In this paper, we introduced a simple yet efficient MLP-based architecture with carefully designed building blocks which achieves competitive performance on the full-body motion synthesis task. Then we proposed AGRoL, a conditional diffusion model for full-body motion synthesis based on sparse tracking signal. AGRoL leverages a simple yet efficient conditioning scheme for structured human motion data. We show that our lightweight diffusion-based model generates realistic and smooth human motions while achieving real-time inference speed, making it suitable for online AR/VR applications. A notable limitation of our and related approaches is sometimes occurring floor penetration artefacts. The future work would be investigating this problem and incorporating extra physical constraints in the model.

References

- [1] Osu accad. <https://accad.osu.edu/research/motion-lab/system-data>. 4
- [2] Sfu motion capture database. <https://mocap.cs.sfu.ca/>. 4
- [3] Ijaz Akhter and Michael J Black. Pose-conditioned joint angle limits for 3d human pose reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1446–1455, 2015. 4
- [4] Sadegh Aliakbarian, Pashmina Cameron, Federica Bogo, Andrew Fitzgibbon, and Thomas J Cashman. Flag: Flow-based 3d avatar generation from sparse observations. In *CVPR*, pages 13253–13262, 2022. 2, 4, 5
- [5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3
- [6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3
- [7] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 3
- [8] Carnegie Mellon University. CMU MoCap Dataset. 4
- [9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 34:8780–8794, 2021. 3, 6
- [10] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ICLR*, 2016. 2
- [11] Andrea Dittadi, Sebastian Dziadzio, Darren Cosker, Ben Lundell, Thomas J Cashman, and Jamie Shotton. Full-body motion from a single head-mounted device: Generating smpl poses from partial observations. In *ICCV*, pages 11687–11697, 2021. 1, 2, 3, 4, 5
- [12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 4
- [13] Eyes, JAPAN Co. Ltd. Eyes, Japan. 4
- [14] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *ICCV*, pages 4346–4354, 2015. 3
- [15] Saeed Ghorbani, Kimia Mahdavian, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F Troje. Movi: A large multipurpose motion and video dataset. *arXiv preprint arXiv:2003.01888*, 2020. 4
- [16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 3
- [17] Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and José MF Moura. Adversarial geometry-aware human motion prediction. In *ECCV*, pages 786–803, 2018. 3
- [18] Wen Guo, Yuming Du, Xi Shen, Vincent Lepetit, Alameda-Pineda Xavier, and Moreno-Noguer Francesc. Back to mlp: A simple baseline for human motion prediction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023. 2
- [19] Ikhsanul Habibie, Daniel Holden, Jonathan Schwarz, Joe Yearsley, and Taku Komura. A recurrent variational autoencoder for human motion synthesis. In *British Machine Vision Conference*, 2017. 3
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3
- [21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020. 1, 2, 3, 4, 11
- [22] Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J Black, Otmar Hilliges, and Gerard Pons-Moll. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM TOG*, 37(6):1–15, 2018. 1, 2
- [23] Ashesh Jain, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, June 2016. 3
- [24] Jiayi Jiang, Paul Strelci, Huajian Qiu, Andreas Fender, Larissa Laich, Patrick Snape, and Christian Holz. Avatarposer: Articulated full-body pose tracking from sparse motion sensing. *ECCV*, 2022. 1, 2, 3, 4, 5, 6, 7, 11, 12, 13, 14
- [25] Yifeng Jiang, Yuting Ye, Deepak Gopinath, Jungdam Won, Alexander W Winkler, and C Karen Liu. Transformer inertial poser: Attention-based real-time human motion reconstruction from sparse imu. *arXiv preprint arXiv:2203.15720*, 2022. 1
- [26] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pages 4401–4410, 2019. 3
- [27] Manuel Kaufmann, Yi Zhao, Chengcheng Tang, Lingling Tao, Christopher Twigg, Jie Song, Robert Wang, and Otmar Hilliges. Em-pose: 3d human pose estimation from sparse electromagnetic trackers. In *ICCV*, pages 11510–11520, 2021. 1
- [28] Jihoon Kim, Jiseob Kim, and Sungjoon Choi. Flame: Free-form language-based motion synthesis & editing. *arXiv preprint arXiv:2209.00349*, 2022. 3
- [29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4
- [30] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 3
- [31] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. Convolutional sequence to sequence model for human dynamics. In *CVPR*, pages 5226–5234, 2018. 3
- [32] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13401–13412, 2021. 12

- [33] Matthew Loper, Naureen Mahmood, and Michael J Black. Mosh: Motion and shape capture from sparse markers. *ACM Transactions on Graphics (ToG)*, 33(6):1–13, 2014. 4
- [34] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM TOG*, 34(6):1–16, 2015. 3, 4, 12
- [35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5
- [36] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *ICCV*, pages 5442–5451, Oct. 2019. 2, 4, 11, 12, 14
- [37] Christian Mandery, Ömer Terlemez, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. The kit whole-body human motion database. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 329–336. IEEE, 2015. 4
- [38] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation mocap database HDM05. Technical Report CG-2007-2, Universität Bonn, June 2007. 4
- [39] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 2, 3
- [40] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 1, 2, 5
- [41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5
- [42] Mathis Petrovich, Michael J Black, and Gül Varol. Action-conditioned 3d human motion synthesis with transformer vae. In *CVPR*, pages 10985–10995, 2021. 7
- [43] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7(1):5, 2017. 3, 4
- [44] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 4, 11
- [45] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11488–11499, 2021. 4, 5
- [46] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 1
- [47] Mingyi Shi, Kfir Aberman, Andreas Aristidou, Taku Komura, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Motionet: 3d human motion reconstruction from monocular video with skeleton consistency. *ACM TOG*, 40(1):1–15, 2020. 7
- [48] Leonid Sigal, Alexandru O Balan, and Michael J Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International journal of computer vision*, 87(1):4–27, 2010. 4
- [49] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 1, 2, 3
- [50] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 5
- [51] Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. Local motion phases for learning multi-contact character movements. *ACM TOG*, 39(4):54–1, 2020. 3
- [52] 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. 3, 6
- [53] Nikolaus F. Troje. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision*, 2(5):2–2, Sept. 2002. 4
- [54] Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. Total capture: 3d human pose estimation fusing video and inertial sensors. In *Proceedings of 28th British Machine Vision Conference*, pages 1–13, 2017. 4
- [55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4, 6
- [56] Zhiyong Wang, Jinxiang Chai, and Shihong Xia. Combining recurrent neural networks and adversarial training for human motion synthesis and control. *IEEE transactions on visualization and computer graphics*, 27(1):14–28, 2019. 3
- [57] Alexander Winkler, Jungdam Won, and Yuting Ye. Questsim: Human motion tracking from sparse sensors with simulated avatars. *ACM TOG*, 2022. 2
- [58] Dongseok Yang, Doyeon Kim, and Sung-Hee Lee. Lobstr: Real-time lower-body pose prediction from sparse upper-body tracking signals. In *Comput. Graph. Forum*, volume 40, pages 265–275. Wiley Online Library, 2021. 1, 2
- [59] Xinyu Yi, Yuxiao Zhou, Marc Habermann, Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Feng Xu. Physical inertial poser (pip): Physics-aware real-time human motion tracking from sparse inertial sensors. In *CVPR*, pages 13167–13178, 2022. 5
- [60] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiandiffuse: Text-driven human motion generation with diffusion model. *arXiv preprint arXiv:2208.15001*, 2022. 3, 6
- [61] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, pages 5745–5753, 2019. 4

We first show extra ablation experiments of our method on AMASS [36] dataset following the protocol proposed in [24]. Then we show extra qualitative comparison between our method and the state-of-the-art method [24].

A. Extra Ablation Experiments

Sampling Steps In Table I we ablate the number of sampling steps T during training. Surprisingly, even when training with merely 10 sampling steps, the model can achieve decent performance. Although we notice that the model converges to a worse local minimum when only a few sampling steps is used. To achieve the best results, more sampling steps is required.

Input/Output length The proposed AGRoL model takes a sequence of sparse tracking signals as input and predicts the full body motion of the same length. In Table II we ablate the input & output length N of the AGRoL model. Our model benefits from longer input sequences, especially decreasing the mean per joint velocity error and jitter. But the performance saturates after the length of $N = 196$. In Table III we further compare our method with AvatarPoser [24] by varying its input length. Note that with longer input sequences our model can achieve significantly lower errors on velocity-related metrics like MPJVE and jitter, while AvatarPoser still has large MPJVE and jitter even with longer input length, thus failing to fully leverage the temporal information to generate smooth motions.

# Sampling Steps	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Jitter
10	2.69	3.70	19.41	1.47	1.55	6.80	7.63
100	2.65	3.62	18.74	1.33	1.52	6.66	6.71
1000 (Ours)	2.66	3.71	18.59	1.31	1.55	6.84	7.26

Table I. Ablation of the number of sampling steps during training the AGRoL model. The results become worse when the number of sampling steps is too small. More sampling steps is beneficial during training the network.

Input & Output Length	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Jitter
41	2.59	3.64	23.24	1.28	1.50	6.73	13.67
98	2.61	3.70	20.71	1.58	1.57	6.76	10.59
196 (Ours)	2.66	3.71	18.59	1.31	1.55	6.84	7.26
256	2.81	3.81	19.05	1.27	1.57	7.03	7.76

Table II. Ablation of the input & output length of the AGRoL model. Our model can benefit from larger input length.

Methods	Input Length	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Jitter
AvatarPoser [24]	41	3.08	4.18	27.70	2.12	1.81	7.59	14.49
AGRoL	41	2.59	3.64	23.24	1.28	1.50	6.73	13.67
AvatarPoser [24]	196	3.05	4.20	28.71	1.61	1.70	7.82	16.96
AGRoL (Ours)	196	2.66	3.71	18.59	1.31	1.55	6.84	7.26

Table III. Comparison between AGRoL and AvatarPoser [24] while varying the number of input frames. Our method can benefit from longer inputs and generate smoother motion. In contrast, AvatarPoser fails to gain consistent improvement from longer input sequences and even degrades in some metrics, including MPJVE, Lower PE, and Jitter.

Predicting noise Our diffusion model AGRoL follows [44] and directly predicts the clean signal $\hat{x}_0^{1:N}$ in contrast to the original Denoising Diffusion Probabilistic Model (DDPM) formulation [21], where the model predicts residual noise $\epsilon_\theta(x_t, t)$ at every step. In this subsection, we further discuss the experiment presented in Table 3 of the main paper, where we implemented a version of AGRoL model (“AGRoL - pred noise”) that predicts the residual noise $\epsilon_\theta(x_t, t)$. Similar to [44], we also find it better to predict the unnoised $\hat{x}_0^{1:N}$ directly, which is demonstrated by the results in Table IV. Since our simple MLP network (see Sect. 3.2 of the main paper) can already produce reasonable estimations of the full body motion using only one forward pass (see Table 1 in the main paper), we hypothesize that the DDPM formulation of Ramesh et al. [44] allows to exploit the full capacity of the network *at every sampling step*, in contrast to the original formulation of [21].

Method	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Root RE	Jitter
AGRoL - pred noise ϵ_θ	5.41	8.88	28.67	4.38	3.91	16.06	8.76	22.78	9.80
AGRoL (Ours)	2.66	3.71	18.59	1.31	1.55	6.84	3.36	19.62	7.26

Table IV. Ablating different formulations of the diffusion model: Predicting clean signal directly (Ours) vs predicting noise $\epsilon_\theta(x_t, t)$. The AGRoL model that learns to predict clean body motion at every diffusion step is substantially better in every metric.

Method	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Root RE	Jitter
AvatarPoser	4.37	9.11	97.24	4.31	3.32	17.47	8.11	8.03	65.18
MLP (Ours)	3.63	7.33	74.90	3.86	2.69	14.03	5.48	6.85	47.16
AGRoL (Ours)	3.56	6.83	65.58	2.17	2.04	13.74	4.91	6.82	41.95
GT	0	0	0	0	0	0	0	0	30.48

Table V. Comparison of our approach with others on AIST++ [32] dataset.

B. Extra Datasets

In addition to the AMASS [36] dataset, we also evaluate the performance of our approach on AIST++ [32] dataset. AIST++ dataset contains in total 5.1 hours of dancing movements performed by professional dancers. The dataset has 10 genres of dances, including some dances containing complicated movements like breakdancing, jazz etc. We follow the train/test splits proposed in [32]. The global rotation and translation of the hands and head are calculated using the SMPL human model [34] with the provided model parameters. Compared to the AMASS dataset, which contains mostly everyday life motions, the motions in the AIST++ dataset are much more diverse and challenging. As shown in Table V, the AGRoL achieves superior performance in all the metrics and produces smoother motions compared to the AvatarPoser and the predictive MLP model. While there is still room for improvement on such a challenging dataset, the proposed AGRoL method significantly reduces the MPJVE, Jitter and lower body positional error (Lower PE) compared to the AvatarPoser.

C. Extra Qualitative Results

In Figure I we show extra qualitative comparisons between our method and AvatarPoser [24]. Please refer to our video in the supplementary material for more qualitative results. As shown in the video, our method reconstructs the full body poses more accurately, it can generate smoother motions and alleviate the jittering issue compared to AvatarPoser.

We also demonstrate failure cases for the proposed AGRoL model in Figure II. We can see that our method fails when we test it on irregular poses, that were not well covered in the training set, or when the lower body pose does not have strong correlation with the upper body. For example, during the break dance motion (Fig. II, bottom row) the upper body may stay static, while legs move which makes it very challenging to predict legs accurately. Increasing the size and diversity of the training set plus incorporating extra physical or geometrical priors to prevent floor penetration could be a potential solution for the failure cases. We plan to further investigate it in future work.



Figure I. Qualitative comparison on test sequences from AMASS dataset: **(left)** GT, **(middle)** AvatarPoser, **(right)** AGRoL. We visualize the predicted skeleton and human body meshes in the figures. We visualize the predicted skeletons and human body meshes in the figures. The skeletons in **blue** denote the ground truth, in **red** – predicted by AvatarPoser [24], and in **green** – predicted by our method. As shown in the figure, our predicted motion is more accurate compared to the predicted motion of AvatarPoser. RGB axes illustrate the location and orientation of the head and hands provided as input to the models. *Please refer to our video for more qualitative results.*



Figure II. Failure test cases: (left) GT, (middle) AvatarPoser, (right) AGRoL. We visualize the predicted skeletons and human body meshes in the figures. The models were trained on the subset of AMASS [36] following the protocol of [24]. The skeletons in blue denote the ground truth, in red – predicted by AvatarPoser [24], and in green – predicted by our method. RGB axes illustrate the location and orientation of the head and hands provided as input to the models. We can see that our method fails (inaccurate poses, floor penetration) when we test it (a) on irregular poses, that were not very common in the training set, or (b) when the lower body pose does not have strong correlation with the upper body as we see in the break dance motion in the bottom row. Increasing the diversity of training motions and incorporating extra physical constraints may help to address such failure cases.