

# Learning to Control Physically-simulated 3D Characters via Generating and Mimicking 2D Motions

## Supplementary Material

### A. Preliminary of Reinforcement Learning

We approach the single-view 2D motion tracking using reinforcement learning, which formulates a Markov Decision Process (MDP) defined by the tuple  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ , where  $\mathcal{S}$  represents states,  $\mathcal{A}$  actions,  $\mathcal{T}$  environment dynamics, and  $\mathcal{R}$  reward functions. The tracking controller is modeled as a policy  $\pi(a_t|s_t)$ , which samples actions  $\mathbf{a}_t \in \mathcal{A}$  based on the current state  $\mathbf{s}_t \in \mathcal{S}$ . We collect state-action-reward trajectories  $\tau = \{(\mathbf{s}_t, \mathbf{a}_t, r_t)\}_{t=1}^T$  in the simulator, following the policy's actions, the simulation dynamics  $\mathcal{T}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ , and the reward function  $r_t = \mathcal{R}(\mathbf{s}_t, \mathbf{a}_t)$ . The training objective is to learn an optimal policy  $\pi^*$  that maximizes the expected cumulative return  $R = \mathbb{E}_\pi [\sum_t \gamma^t r_t]$  over the collected trajectories.

### B. Implementation Details

#### B.1. Data Preprocessing

We adopt a pinhole camera model for reprojection computation, where each camera view  $C$  is parameterized by  $C = (R_C, \tau_C, f_C)$ , with  $R_C$  and  $\tau_C$  denoting the extrinsic rotation and translation, and  $f_C$  representing the intrinsic parameters.

For in-the-wild videos, we estimate the camera parameters  $C$  and the initial states  $s_0$  directly from the extracted 2D motion sequences. We jointly optimize the camera extrinsics and the 3D character poses by minimizing a reprojection loss, while keeping the intrinsics fixed. The resulting 3D poses are then used as the initial states for single-view tracking training. To avoid ground penetration, we include an additional regularization term that penalizes any body points falling below the ground plane.

#### B.2. Dataset Statistics.

Table 4 shows the statistics of our self-collected 'in-the-wild' video data.

Table 4. Statistics of our curated datasets.

Dataset	Length(s)	Range(s)	Usage
Soccer Dribble	1033	3.5–17.9	Tracking / Generator
Animal	294	0.9–5.3	Tracking

#### B.3. Proprioceptive States Representation

The proprioceptive states  $o_{\text{prop}}$  consists of the following elements: root height  $\mathbf{h}_t \in \mathbb{R}$ , root velocity in the local co-

ordinate frame  $\mathbf{v}_t \in \mathbb{R}^3$ , root angular velocity in the local frame  $\mathbf{w}_t \in \mathbb{R}^3$ , joint rotations in local joint coordinates  $\mathbf{q}_t \in \mathbb{R}^{6 \times J}$ , joint velocities in local joint coordinates  $\dot{\mathbf{q}}_t \in \mathbb{R}^{3 \times J}$ , and the Cartesian positions of key body links in the local root frame  $\mathbf{p}_t \in \mathbb{R}^K$ .

#### B.4. Algorithm of Adaptive State Initialization

The details of the proposed adaptive state initialization strategy are provided in Algorithm 1.

---

#### Algorithm 1 Initial State Distribution Update and Sampling

---

```

1: Input: Initial state distribution  $d(s_0)$ , Reference motion frames  $\mathbf{X} = \{\mathbf{x}_0, \dots, \mathbf{x}_T\}$ , Critic network  $V_\phi$ 
2: Output: Updated initial state buffer  $\mathcal{B}$ 
3: Initialize: Create state buffer  $\mathcal{B} = \{\mathcal{B}_t\}_{t=0}^T$ , where each  $\mathcal{B}_t$  is dedicated to frame  $\mathbf{x}_t$ .
4: _____
5: function SAMPLEINITIALSTATE( $\mathcal{B}_t$ )
6:   Scores  $\leftarrow \{\text{Score}(s) \mid s \in \mathcal{B}_t\}$ 
7:   Probabilities  $\leftarrow \text{Normalize}(\exp(\beta \cdot \text{Scores}))$   $\triangleright \beta$  controls prioritization
8:   return Sample( $\mathcal{B}_t$ , Probabilities)
9: end function
10: _____
11: function UPDATESATEBUFFER( $\mathcal{B}$ , Rollouts)
12:   for each state  $\{s_t, \mathbf{x}_t\}$  at frame  $\mathbf{x}_t$  in Rollouts do
13:     CriticScore  $\leftarrow V_\phi(s_t)$   $\triangleright$  Estimate value/tracking performance
14:     if CriticScore > Threshold or  $|\mathcal{B}_t| < \text{MinSize}$  then
15:       Score( $s_t$ )  $\leftarrow$  CriticScore
16:        $\mathcal{B}_t \leftarrow \mathcal{B}_t \cup \{s_t\}$   $\triangleright$  Store state in the buffer
17:     if  $|\mathcal{B}_t| > \text{MaxSize}$  then
18:        $\mathcal{B}_t \leftarrow \text{RemoveLowestScore}(s', \mathcal{B}_t)$   $\triangleright$  Maintain buffer size limit
19:     end if
20:   end if
21: end for
22: end function

```

---

#### B.5. 2D Motion Tokenizer

We employ a VQ-VAE as the motion tokenizer. The VQ-VAE comprises an encoder  $E(\cdot)$  and a decoder  $D(\cdot)$ . The encoder converts the relative 2D motion sequence  $\mathbf{x}^{\text{rel}}$  into a sequence of lower-dimensional latent vectors,  $\mathbf{z} =$

$[z_0, z_1, \dots, z_{T/l}]$ , where  $l$  denotes the temporal downsampling factor. Each latent vector  $z_i$  is then quantized to its nearest entry in a learned codebook of discrete embeddings  $\mathbf{C} = \{c_k\}_{k=1}^K$  by  $\hat{z}_i = \arg \min_{c_k \in \mathbf{C}} \|z_i - c_k\|_2$ .

The decoder reconstructs the original relative 2D motions from the quantized latent representations, i.e.,  $\hat{\mathbf{x}}^{\text{rel}} = D(\hat{\mathbf{z}})$ , where  $D(\cdot)$  denotes the decoder and  $\hat{\mathbf{z}}$  are the quantized latents.

The VQ-VAE is trained by minimizing the following loss function:

$$\mathcal{L}_{\text{vq}} = \mathcal{L}_{\text{recon}} + \|\text{sg}[\mathbf{Z}] - \hat{\mathbf{Z}}\|_2^2 + \beta \|\mathbf{Z} - \text{sg}[\hat{\mathbf{Z}}]\|_2^2, \quad (9)$$

where  $\mathcal{L}_{\text{recon}}$  is the reconstruction loss, the second term encourages the embedding vectors to be close to the encoder outputs, and the third term penalizes the encoder for deviating from the quantized embeddings. Here,  $\text{sg}[\cdot]$  denotes the stop-gradient operator, and  $\beta$  is a hyperparameter that balances the commitment loss. We stabilize training through exponential moving average (EMA) updates on codebook embeddings [62].

## B.6. Autoregressive Generator

We use a Transformer decoder architecture for the autoregressive generator to predict motion tokens. Each discrete token index is first mapped into an embedding, and a conditional embedding, which can be derived from the given control input or set to zero for unconditional generation, is prepended to the sequence. Absolute positional embeddings are added to encode the temporal order.

For generative tasks that are more sensitive to depth ambiguity in single-view 2D inputs, such as dance motion generation, we train a multi-view 2D generator to supply an additional viewpoint, helping to resolve missing depth cues. To construct training data for the multi-view generator, we collect 3D motion states from the simulated character while it tracks the original single-view 2D sequences, then project them into multiple 2D views to obtain paired multi-view motion samples. We represent the resulting multi-view 2D motions as sequential data, where each frame contains multiple motion tokens ordered by view ID.

## B.7. Key Hyperparameters

The key hyperparameters for policy learning are listed in Table 5. Our PPO implementation is based on RL-Games [21]. The hyperparameters for the VQ-VAE and the autoregressive generator are provided in Table 6 and Table 7, respectively.

## C. Additional Results

**Investigation of View-Aggregation Strategies.** We investigate an alternative view-aggregation strategy, *Action-Blending*, which performs blending in the action space

Table 5. Key Hyperparameters for the Policy Learning (PPO).

Hyperparameter	Value
Optimizer	Adam
Batch Size	16,384
Learning rate	$2 \times 10^{-5}$
Total Training Frames	$5 \times 10^{10}$
Number of Environments	16,384
Rollout Length	32
PPO Epochs ( $N_{\text{epochs}}$ )	3
Value Loss Coefficient	5
Discount Factor $\gamma$	0.99
GAE $\tau$	0.95
PPO clipping	0.2

Table 6. Key Hyperparameters for the VQ-VAE Motion Tokenizer.

Hyperparameter	Value
<b>Training Configuration</b>	
Optimizer	Adam
Learning Rate	$5 \times 10^{-5}$
Batch Size	128
Training Steps	600,000
<b>Codebook Parameters</b>	
Codebook Size ( $K$ )	512
Embedding Dimension ( $D_{\text{embed}}$ )	128
<b>Architecture (ResNet Encoder/Decoder)</b>	
Model Width (Channels)	512
Number of ResNet Blocks	2
Temporal Downsampling Factor	2

Table 7. Key Hyperparameters for the Autoregressive Generator.

Hyperparameter	Value
<b>Training Configuration</b>	
Optimizer	Adam
Learning Rate	$5 \times 10^{-5}$
Batch Size	128
Training Steps	600,000
<b>Architecture (Transformer Decoder)</b>	
Transformer Layers ( $N_{\text{layers}}$ )	4
Attention Heads ( $N_{\text{head}}$ )	4
Embedding Dimension	256
<b>Data &amp; Sequence Parameters</b>	
Motion Clip Length (Frames)	60

rather than in the network feature space. Results on the AIST++ train and test splits are reported in Table 8, where  $V$  denotes the number of views. Views are uniformly sam-

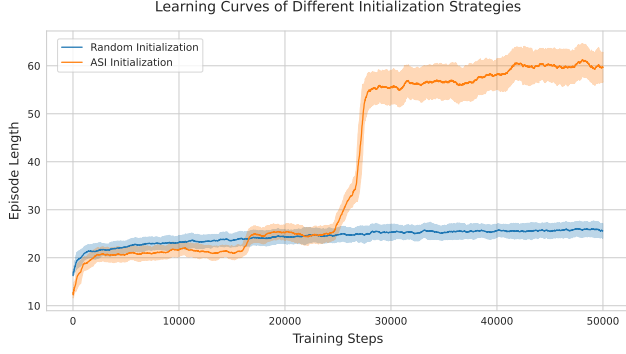


Figure 8. **Learning curve of 2D motion imitation with and without adaptive state initialization.** Adaptive optimization of the initial state distribution leads to significantly faster convergence in 2D motion imitation.

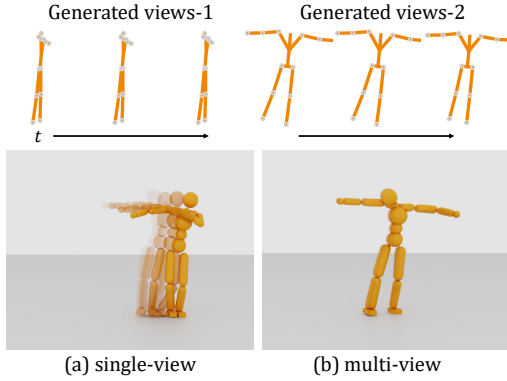


Figure 9. **Impact of multi-view guidance on resolving depth ambiguity.** Top: Reference keyframes generated from two view-points. Bottom: (a) Tracking guided only by View-1 results in instability and sliding due to depth ambiguity. (b) Tracking guided by both views successfully resolves these ambiguities, allowing the controller to recover stable and physically plausible 3D poses.

Table 8. Zero-Shot Multi-View Aggregation on AIST++.  $V$  denotes the number of aggregated views.

Method	$V$	Train		Test	
		$E_{3D} \downarrow$	$SR \uparrow$	$E_{3D} \downarrow$	$SR \uparrow$
Action-Blending	2	178.5	80.2	195.6	80.2
	3	182.1	80.2	196.6	80.1
<b>Ours</b>	2	147.8	90.1	164.9	88.0
	3	<b>144.2</b>	<b>90.4</b>	<b>161.5</b>	<b>88.9</b>

pled over a  $180^\circ$  range.

**State Initialization Strategy** We compare the learning curves of average tracking duration for a highly challenging animal jumping motion, with and without our proposed adaptive state initialization strategy. As shown in Fig. 8,

our method enables effective convergence of policy learning, while the compared one struggles due to physically implausible initial states that hinder the training process.

**Impact of Multi-view Guidance** While our view-agnostic 2D tracking policy leverages diverse training view-points to acquire a robust 3D understanding, the inherent depth ambiguity of single-view inputs can still persist during inference for complex motions, causing irregular movements. To investigate this, we conducted an experiment using two orthogonal 2D reference views synthesized by our proposed 2D motion generator. We compared the policy’s performance when tracking only one of these views versus tracking both simultaneously. As illustrated in Fig. 9, the policy tracking a single generated view suffers from severe depth ambiguity, resulting in artifacts like foot sliding. In contrast, the policy integrating both views successfully resolves this ambiguity, maintaining stable poses and plausible motion.

(R2)Ablation on VQVAE Reconstruction Loss. We measure the reconstruction error of global 2D poses (Recon. G), which accounts for translation and scale variations, as well as the error of local 2D poses (Recon. L), which considers only human-centric, scale-invariant poses. Results with and without the absolute loss term are reported in Table 9, indicating that the introduced loss helps reconstruct 2D motions with more accurate global movements.

Table 9. Ablation study of the loss in the VQ-VAE training.

Configuration	AIST++		Dribble	
	Recon. L	Recon. G	Recon. L	Recon. G
w/o Absolute Loss	<b>149.5</b>	2979.5	<b>11.71</b>	933.71
<b>Ours</b>	203.51	<b>345.06</b>	16.04	<b>52.73</b>

## D. Limitation and Future Work

In our current approach, the 2D tracking policy relies exclusively on reprojection-error-based rewards. This geometric objective may be insufficient for learning precise contact dynamics required for fine-grained object interactions, such as dexterous manipulation or tool use. Future work could address this by introducing additional semantic rewards to explicitly ground these contact constraints. Moreover, our framework currently assumes that camera parameters are estimated with reasonable accuracy. In practice, however, inaccurate camera estimation can significantly degrade the performance of the 2D tracking policy. To mitigate this, future iterations could incorporate camera parameters directly into the learning procedure, allowing for the joint optimization of motion control and camera estimation.