Walk Like This: Stylized Locomotion for Physics-Based Characters through Example-Driven Diffusion

Joshua Aurand[1,2] **Agon Serifi**[1] **Sammy Christen**[1] **Ruben Grandia**[1]
Espen Knoop[1] **Marco Hutter**[2] **Moritz Bächer**[1]
[1]Disney Research Zürich, Switzerland [2]ETH Zürich, Switzerland
joshua.aurand@disneyresearch.com



Figure 1: We introduce a real-time controller for physics-based characters that lets users steer the character with joystick commands while conditioning on style from unseen examples of arbitrary duration and various sources (e.g., MoCap, video).

Abstract: We present a real-time framework for expressive legged locomotion that enables independent control over velocity and style. Unlike existing data-driven controllers, which typically require explicit style labels for training, our method efficiently leverages large-scale unstructured motion data and generalizes to unseen styles. We disentangle style from functional locomotion through a tailored sampling strategy used to train a style encoder whose output conditions an autoregressive diffusion model. During inference, the user provides a style reference clip and a joystick velocity command. The diffusion model then generates a kinematic reference tracked by a reinforcement learning controller. To improve user controllability and stylization strength, we propose an attention biasing mechanism and a velocity command correction strategy. We demonstrate our method on a physics-based humanoid character in simulation and on a physical robot, showcasing robust real-time control and continuous style modulation. The results can be best seen in our supplemental video[1].

Keywords: Motion Generation, Physics-Based Control, Stylized Locomotion

1 Introduction

A central challenge in physics-based humanoid control is to move beyond functional locomotion and to perform tasks in an expressive gait *style*. Although some stylized locomotion controllers exist, they are typically designed around fixed objectives or limited to imitating a narrow distribution of motions [1, 2, 3, 4, 5], offering little flexibility for stylistic variation and generalization after training. Similarly, while CAMDM [6] achieves real-time stylized character control, its conditioning on explicit style labels prevents generalization to new styles. Style transfer offers an alternative by mapping style onto target motions [7, 8, 9], but these methods are not designed for real-time robotic tracking or interactive control. On the other hand, motion generators [10, 11, 12, 13] can be grounded in physics but lack expressive control over style. Hence, a real-time framework that adapts to unseen styles on humanoid robots is currently missing.

[1] https://www.youtube.com/watch?v=roFnS6N7nJ4

10th Conference on Robot Learning (CoRL 2026), Austin, Texas, USA.

Table 1: **Comparison to Related Work.** Real-time (RT), generative (Gen), unseen styles (Uns Sty), physics-based model (Phys), joystick control (Joy Ctrl). *Legend:* ✓ Yes ∼ Partially ✗ No

Approach	RT	Gen	Uns Sty	Phys	Joy Ctrl
Style Transfer [7, 8, 9, 15, 16, 17, 18, 19]	✗	∼	✓	✗	✗
Implicit Rewards [1, 2]	✓	∼	✗	✓	∼
"Physical Diffusion" [10, 11, 12, 13]	∼	✓	✗	✓	∼
CAMDM [6]	✓	✓	✗	✗	✓
Ours	✓	✓	✓	✓	✓

To fill this gap, we propose a framework for the interactive control of humanoid robots that enables real-time user control via joystick commands and an example motion clip that contains the desired style characteristics. This gives users intuitive control over both *what* the robot does and *how* it performs these movements. To enable this, we build a stylized generative motion model based on autoregressive diffusion [6]. Crucially, we condition the model on a style embedding that is extracted from a style example through a transformer-based encoder. We jointly train the diffusion model and style encoder on unlabeled data, and use a tailored sampling strategy to disentangle style from functional locomotion. Our method both generalizes to new styles and gives users the flexibility to provide style examples of any duration. We highlight both aspects in our experiments by extracting a style directly from a recorded human video demonstration.

To enhance style adherence and enable rapid style transitions, we introduce an inference-time technique that re-weights the attention scores within the diffusion model, outperforming standard classifier-free guidance (CFG) [14] both in efficiency and effectiveness. Furthermore, we add a velocity correction scheme that mitigates drift when following joystick commands. Finally, we bridge the gap between generated motion and hardware dynamics by pairing the diffusion model with a reinforcement learning (RL) tracking controller. This combines the *expressive power* of a motion generator with the *robustness* of an RL low-level controller, as we validate with closed-loop simulation and hardware deployment.

Our core contributions: **(i) A generative motion-control framework** enabling user control over locomotion and style, trained from large-scale unstructured datasets and demonstrated on hardware. **(ii) An autoregressive diffusion model and style encoder** that disentangle style from functional locomotion through an effective sampling strategy during training. **(iii) An inference-time attention biasing technique** to modulate stylization strength alongside **a velocity correction scheme** to mitigate drift. We validate each design choice through ablations.

2 Related Work

Our work builds upon advancements in kinematic motion synthesis and physics-based control. Tab. 1 contrasts our work with prior approaches.

Stylized Motion Generation Stylized motion generation traditionally relies on *style transfer* from reference to content motion [20], with modern approaches utilizing network activation matching [7, 15, 16, 17] or diffusion models [9, 18, 19]. While they produce high-quality results, these methods are usually too slow for real-time use and depend heavily on the source content motion. Even when coupled with generative models to mitigate this dependency [9, 19], they still lack the performance and controllability required for interactive applications. In contrast, purely generative approaches are better suited for real-time interaction [6, 21, 22, 23, 24, 25, 26]. Closest to our work is CAMDM [6], a state-of-the-art interactive generator that uses an autoregressive diffusion model conditioned on discrete labels. While CAMDM achieves real-time, high-quality synthesis, it is restricted to seen training styles, provides purely kinematic motion, and lacks explicit control over style intensity or transitions. We overcome these limitations by introducing a more general style conditioning mechanism, explicit control over stylization strength, and an RL tracker to enforce physicality.

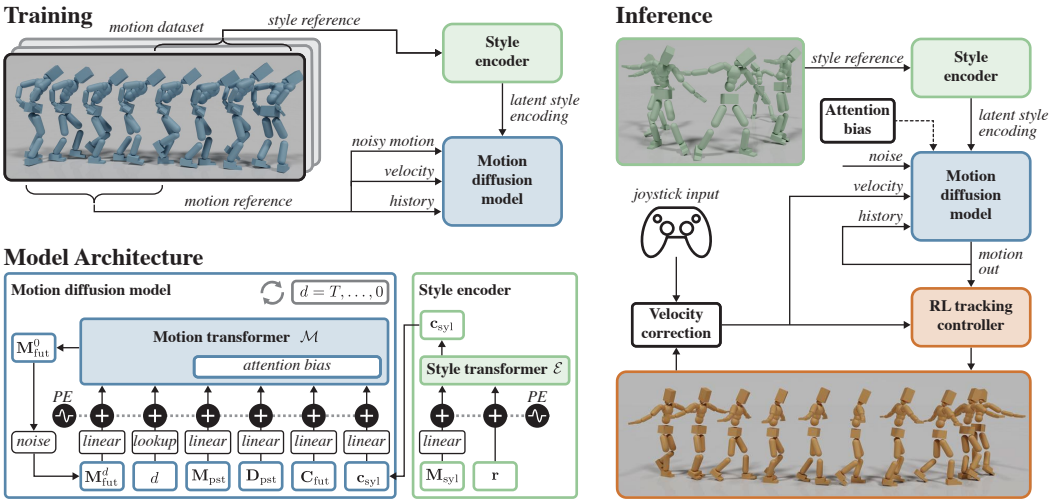

Figure 2: **Overview.** We *train* an autoregressive diffusion model on unstructured motion datasets. We explicitly encode style in the style encoder, enabling generalization to unseen style references of varying length at inference. During *inference*, users can control the robot's velocity through joystick commands, while selecting a desired style clip. The motion diffusion model's output is autoregressively rolled out and passed to an RL-based tracking controller.

Physics-Based Character Control Physics-based RL is well-established for robust humanoid locomotion [27, 28, 29, 30, 31]. Compared to command-based strategies [27, 28, 32], tracking-based approaches offer superior style control [31, 33, 34], scaling from single references [34] to large datasets via mixture-of-experts [35, 36] or implicit rewards [1, 2, 3]. However, frameworks like CALM [37] fail to explicitly disentangle style from skills, and character-specific gait-design tools [4, 5, 38, 39] lack generalization. To enforce physical feasibility, recent trends combine kinematic diffusion with RL through iterative training [13], joint state-action prediction [40], or closed-loop fine-tuning [11], though these are often limited to simulation. RobotMDM [10] tackles the sim-to-real gap by fine-tuning the diffusion model. We instead train a tracking policy from scratch on a large dataset of diffusion-generated motions.

3 Background

We build on the framework of Motion Diffusion Models (MDMs) [41, 42]. Given a noisy motion $\mathbf{M}^d$, whose noise level depends on the diffusion step d, an MDM $\mathcal{M}$ predicts the clean motion $\mathbf{M}^0$. Particularly relevant to our work is CAMDM [6], an autoregressive MDM that predicts the next F frames of motion $\mathbf{M}^0_{\text{fut}}$, conditioned on the past P frames of motion $\mathbf{M}_{\text{pst}}$, as well as future root trajectory $\mathbf{C}_{\text{fut}}$ and a discrete style label ℓ_{syl}:

$$\mathbf{M}^0_{\text{fut}} = \mathcal{M}(\mathbf{M}^d_{\text{fut}}, d; \mathbf{M}_{\text{pst}}, \mathbf{C}_{\text{fut}}, \ell_{\text{syl}}). \tag{1}$$

The future root trajectory contains $x-y$ coordinates and the root orientation of the character in all F future frames, computed by integrating world-frame velocity commands. $\mathcal{M}$ itself is a transformer-encoder [43] where all the inputs are tokenized individually. During inference, a pure noise sample is gradually denoised over multiple steps to reach the final clean prediction. We refer to [6, 41] for details.

4 Method

We develop a real-time interactive controller for stylized locomotion of physics-based characters, as illustrated in Fig. 2. Our framework contains two models: a kinematic, autoregressive diffusion model (Sec. 4.1) and an RL-based tracking controller (Sec. 4.3).

The diffusion model generates kinematic motions based on user velocity commands and a stylized motion clip. A style encoder learns to extract the style from the motion in a compressed latent space. Both the generator and the style encoder are trained jointly from an unstructured motion dataset. During inference, we propose two simple yet effective techniques to improve stylization and adherence to user commands (Sec. 4.2). First, we modify bias terms in the generator transformer's attention layers. Second, we enable intuitive user control through a velocity correction module that mitigates global drift. To transfer the generated kinematic motions onto a physics-based character, we use an RL-based tracking controller. We train this controller on a large dataset generated through autoregressive rollouts of our diffusion model.

4.1 Kinematic Diffusion Model

We use an autoregressive diffusion model for predicting the character's next F states $\mathbf{M}_{\text{fut}}$, inspired by CAMDM [6] (see Sec. 3). We condition the model on the past P states $\mathbf{M}_{\text{pst}}$ and motion differences $\mathbf{D}_{\text{pst}}$, future root states $\mathbf{C}_{\text{fut}}$ and a style condition $\mathbf{c}_{\text{syl}}$:

$$\mathbf{M}_{\text{fut}}^{0} = \mathcal{M}(\mathbf{M}_{\text{fut}}^{d}, d; \mathbf{M}_{\text{pst}}, \mathbf{D}_{\text{pst}}, \mathbf{C}_{\text{fut}}, \mathbf{c}_{\text{syl}}), \tag{2}$$

where $\mathcal{M}$ is a transformer-encoder. Inputs are tokenized individually per timestep through linear maps before being fed to the transformer, except the diffusion step d, for which we use a learned lookup table. We use a motion representation similar to [6], where the state at each timestep consists of the root position, orientation, joint orientations as well as link positions and orientations (details can be found in Appendix A.1). Unlike CAMDM, the future root states $\mathbf{C}_{\text{fut}}$ contain desired root yaw angles and not full rotations so that the character's roll and pitch are fully controlled through the imposed style. Furthermore, we add past motion differences $\mathbf{D}_{\text{pst}}$ as an additional condition, computed as position and rotation differences of consecutive frames in $\mathbf{M}_{\text{pst}}$. While the information in $\mathbf{D}_{\text{pst}}$ is redundant, we found it to improve coherence during autoregressive rollouts, e.g., consistent swing directions of arms.

Encoding Style We compute style embeddings $\mathbf{c}_{\text{syl}}$ by compressing a stylized motion clip $\mathbf{M}_{\text{syl}}$ to a latent representation using our style transformer $\mathcal{E}$, i.e., $\mathbf{c}_{\text{syl}} = \mathcal{E}(\mathbf{M}_{\text{syl}})$. Contrary to discrete style labels [6], such inputs contain explicit information about the style and allow generalization outside the set of training styles [7]. The style transformer $\mathcal{E}$ is also a transformer-encoder, where all frames of the style motion are tokenized through a linear projection. The transformer architecture is well suited to represent the various timespans S of different styles. In addition, we enforce $\|\mathbf{c}_{\text{syl}}\| = 1$ through normalization which facilitates generalization [3, 44] and allows mixing styles through spherical linear interpolation. Inspired by the use of register tokens in vision transformers [45], we use an input-independent register token $\mathbf{r}$ in the inputs of the style transformer. It serves as a working space to capture relevant style information and we use its value after the final transformer layer as the output $\mathbf{c}_{\text{syl}}$. See Fig. 2 for a visualization of the style encoder in the full model architecture.

Training Both $\mathcal{M}$ and $\mathcal{E}$ are trained jointly. We first sample a motion sequence from the training data and then sample $P + F$ consecutive frames to get $\mathbf{M}_{\text{pst}}$ and $\mathbf{M}_{\text{fut}}$. The future root states $\mathbf{C}_{\text{fut}}$ are directly read from the motion. The model is then trained to reconstruct $\mathbf{M}_{\text{fut}}$ from a noisy version of it. For the style encoder, a *different* window of length $S \in [S_{\min}, S_{\max}]$ is sampled from the *same* motion sequence to yield $\mathbf{M}_{\text{syl}}$. Crucially, this dual sampling allows the style transformer to disentangle the style of the sequence from its content. We define content as the ground-projected root trajectory, while style encompasses all residual motion dynamics. The loss function combines a reconstruction loss with velocity, joint consistency, and foot sliding losses, adopted from [6].

4.2 Diffusion Model Inference

We deploy the diffusion model autoregressively, conditioning the prediction of the next F frames on its own past predictions, the style $\mathbf{c}_{\text{syl}}$, and the future root states $\mathbf{C}_{\text{fut}}$. The autoregressive nature of the model, however, comes with a number of limitations, which we address through inference-time adjustments.

Attention Bias Conditioning on past motion gives the model a strong signal for how future motion should be generated, often leading it to ignore style conditioning and get stuck in a single style. To address this, we propose biasing the attention mechanism during inference. Specifically, attention in the transformer layers is computed as

$$\text{attention}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}; \mathbf{B}\right) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D}} + \mathbf{B}\right)\mathbf{V}, \tag{3}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times D}$ are the query, key and value matrices, and $\mathbf{B} \in \mathbb{R}^{N \times N}$ is the bias matrix with N being the number of tokens and D the model dimension. Typically, the bias matrix is set to 0, leaving the model full control over how different tokens attend to each other. We propose adding a bias $w_{\text{syl}} \geq 0$ to the attention scores of all queries toward the style token and a bias $w_{\text{pst}} \leq 0$ to those toward the past-motion tokens. This forces the model to respect the style token, facilitating style transitions and improving stylization strength.

Velocity Correction The user controls the character through joystick commands $(\mathbf{v}_{x,y}, v_\psi)$ comprising heading, lateral, and yaw velocities, which are integrated over timesteps $t \in \{1, \ldots, F\}$ to form the trajectory conditioning signal $\mathbf{C}_{\text{fut}}$ for the diffusion model. During inference, the character can drift from the intended trajectory due to accumulation of prediction errors. Furthermore, the character is controlled through velocity commands expressed in the root heading frame. As the root naturally oscillates during movement, the generated trajectory depends heavily on where exactly in that cycle the diffusion model is triggered.

To alleviate these problems, we keep track of the character's current position and yaw direction, $\mathbf{p}^{\text{cur}}_{x,y}$ and ψ^{cur}, as well as the desired position and direction, $\mathbf{p}^{\text{des}}_{x,y}$ and ψ^{des}. The current state is updated according to the diffusion model predictions whereas the desired state is computed by integrating the given joystick velocity commands $(\mathbf{v}_{x,y}, v_\psi)$. Whenever a new diffusion call is triggered, we apply a feedback correction to the velocity command to compute corrected joystick commands:

$$\tilde{\mathbf{v}}_{x,y} = \mathbf{v}_{x,y} + k_{\mathbf{p}}\mathbf{R}_\psi(\psi^{\text{cur}})^\top(\mathbf{p}^{\text{des}}_{x,y} - \mathbf{p}^{\text{cur}}_{x,y}) \tag{4}$$

$$\tilde{v}_\psi = v_\psi + k_\psi(\psi^{\text{des}} - \psi^{\text{cur}}). \tag{5}$$

Here $k_{\mathbf{p}}$ and k_ψ are feedback gains and $\mathbf{R}_\psi$ is the 2D rotation matrix about the yaw axis. These corrected commands $\tilde{\mathbf{v}}$ are then used to generate the future root trajectory which is used as conditioning input for the diffusion model. By treating the deviation from the intended trajectory as an error signal, this formulation introduces a closed-loop correction that continuously steers the character back toward the user's original commands. We prevent excessive corrections by projecting the desired state into a neighborhood of the current state during each timestep. Specifically, the desired position is clipped to be at most $0.5\,\text{m}$ away from the current position and the desired yaw is clipped into a $0.5\,\text{rad}$ range around the current orientation. We refer to Appendix A.1 for further details.

4.3 RL-Based Tracking Controller

We use RL to train a policy that tracks the generated kinematic motions, largely following related work [33, 34]. The policy predicts actions given observations and goal states. We refer to Appendix A.2 for details on the design of the action space, rewards, observations, and goal states.

Training Data We avoid the computational overhead of querying the diffusion model during RL by generating a large motion dataset offline via autoregressive rollouts. To do so, we run the diffusion model conditioned on style clips randomly sampled from the diffusion model's training data paired with randomly sampled joystick commands. This ensures that we cover the whole space of motions that can be generated by the diffusion model.

Deployment We deploy the policy and diffusion model in an open loop: the diffusion model is conditioned on its own past predictions, rather than the real state of the character, which we found to improve stylization quality. Furthermore, an open-loop scheme allows training on a precomputed

Table 2: **Quantitative Evaluation.** We compare our method to CAMDM [6] and evaluate the effects of attention biasing and CFG.

Method	FID ↓	Style Acc. ↑ [%]	Ft. Slid. ↓ [cm/frame]	Pos. Err. ↓ [cm/frame]	Yaw Err. ↓ [deg/frame]
CAMDM [6]	0.27	86.0	0.71	0.70	0.60
Ours	0.22	88.1	**0.66**	0.70	0.65
CAMDM [6] + Bias	0.14	93.5	0.90	**0.69**	**0.58**
Ours + CFG	0.17	91.5	0.98	0.71	0.68
Ours + Bias	**0.07**	**96.3**	0.93	0.71	0.67
Ours, Unseen Styles	0.34	93.0	0.90	0.71	0.69

dataset instead of running the diffusion model online. This is feasible because we do not deal with precise object or environment interactions that require the diffusion model to adjust to the real-world data, as in other works [11, 46]. To avoid computational overhead from the style encoder, we precompute a set of style encodings and cache them during deployment.

5 Experiments and Results

We first compare our method to CAMDM [6] and analyze our style encoder. We then evaluate the attention biasing and velocity correction. Finally, we ablate our RL tracking policy and transfer our method onto a real robot. We refer to the Appendix for implementation details.

5.1 Diffusion Model

Quantitative Evaluation We compare our diffusion model against CAMDM [6], the current state of the art in generative modeling of stylized motion and the only model directly applicable to our setting when evaluating seen styles (Tab. 1). Both models are trained on the 100STYLE dataset [23]. We report FID score [47], style accuracy, foot sliding, and positional and yaw errors with respect to the condition trajectory (see Appendix A.3). We summarize results in Tab. 2 and provide a qualitative comparison in Appendix A.3.1. Compared to CAMDM, our model generates motion that is closer to the real motion capture data, as indicated by the lower FID and higher style accuracy, while achieving similar results in terms of physicality and alignment to the trajectory conditions. Furthermore, we find that adding attention biasing at inference improves overall motion quality for both CAMDM and our model, which we discuss in more detail in Sec. 5.2.

Style Encoder We highlight the interpolation and generalization capabilities of our style encoder. First, we encode styles from the 100STYLE dataset and spherically interpolate between the corresponding style latents. Next, we evaluate the generalization of our method by capturing videos of a person performing a given style. We then extract 120 frames of motion features from that video using an open-source pose estimation model [48] and pass them as input to the style encoder. Notably, no further processing after pose estimation is needed before feeding the clips to the style encoder. As seen in Fig. 3 and our supplemental video, our model interpolates smoothly between styles and even generalizes to a highly dynamic style such as jumping. For this experiment, we train the model on both the 100STYLE [23] and AMASS [49] datasets to improve generalization capabilities. Furthermore, Tab. 2 shows quantitative evaluations on a set of 10 styles of 100STYLE not included during training. Importantly, CAMDM is not applicable here, as it is inherently limited to seen styles. On unseen styles, our model reproduces their distinctive features and reaches style accuracy comparable to that on seen styles, but it struggles with finer details, leading to a higher FID. Visualizations of the motions generated for the test split can be found in Appendix A.3.1. Finally, we apply three additional probes to the style latents for further evidence of disentanglement between style and content (the ground-projected root trajectory). First, linear discriminant analysis

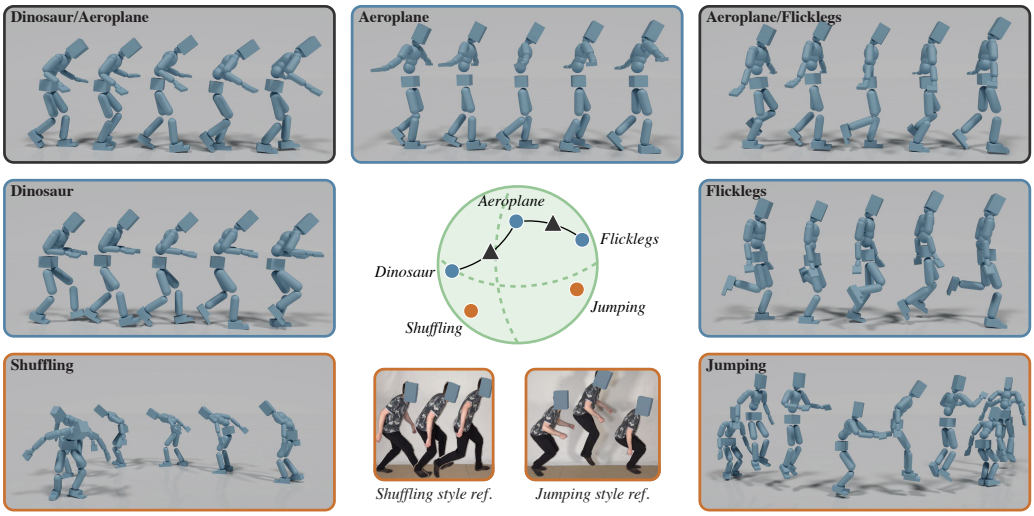

Figure 3: **Style Encoding.** Generated motions conditioned on interpolated style latents of styles from the 100STYLE dataset (top and middle row) and latents computed from unseen motions extracted from videos (bottom row).

[50] recovers style identity from the style latents with 95% accuracy across all styles of 100STYLE. Second, within-style variation of the style latents, computed across different clips of the same style, accounts for only 19% of the total style latent variance. Third, regressing content from the style latents alone yields $R^2 = 0.07$, barely better than predicting the mean content.

Style Clip Sampling Strategy We verify the importance of our style encoder sampling strategy by training a model where we always sample the *same* motion window for both style reference and past and future motion. Using the *same* motion window results in a sample diversity of $0.8\,\mathrm{deg}$ for joint angles across different generated motion windows, while using *different* motion windows (our approach) results in $5.5\,\mathrm{deg}$. We refer to Appendix A.3.1 for qualitative results, which show that without our sampling scheme, the model collapses, causing all generated motion windows to repeat the reference motion. In contrast, with our sampling scheme, we can disentangle the style from the locomotion and generate diverse motion.

5.2 Inference Techniques

Attention Bias We compare our proposed attention bias method to CFG [14] on past motion where the predicted motion is a weighted average of a fully conditioned and a masked prediction (see Appendix A.3). Tab. 2 shows how attention biasing significantly improves quality and stylization of generated motions. As it is a pure inference adjustment, we can apply it to both models, *Ours + Bias* and *CAMDM + Bias*, though the relative improvement is smaller for CAMDM because its discrete style labels are less informative than our style encodings. Improved stylization comes at the cost of more foot sliding, as negative weighting of past motion makes it more challenging to create coherent foot swings. Compared to CFG, attention biasing performs better in all metrics. We conjecture that this performance gap arises because the bias modifies the pre-softmax weights, so attention stays a normalized distribution over the same tokens and thus closer to the training regime. In contrast, CFG interpolates two separate in-distribution predictions at every denoising step, and the result of this guidance step may no longer lie in the training distribution. Qualitative results and a sensitivity analysis with respect to the bias weights can be found in the Appendix (Fig. A.5 and Fig. A.6 respectively).

Table 3: **RL Tracking Policy.** We evaluate the effect of training data and loop mode on the performance of the tracking policy. Additionally, we compare open- and closed-loop performance under strong external disturbances.

Training Data	**Loop Mode**	**Disturbances**	FID ↓	Style Acc. [%] ↑	MPJAE [deg] ↓	Falls [#/min] ↓
MoCap	Open	✗	0.16	92.0	5.2	1.104
Generated	Open	✗	0.35	85.1	8.3	0.001
Generated	Closed	✗	0.45	79.6	7.8	0.001
Generated	Open	✓	0.84	62.9	16.0	3.812
Generated	Closed	✓	0.95	58.2	15.9	4.431

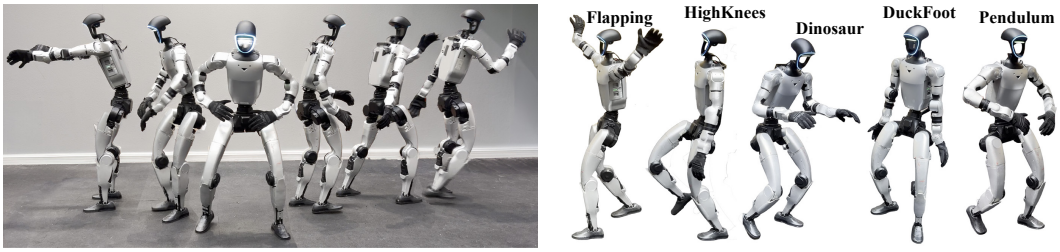

Figure 4: **G1 Robot Deployment.** Left: Continuous joystick control with dynamic style modulation. Right: Demonstration of the framework generating five distinct motion styles.

We also investigate style transition behavior between the different methods more closely. As shown in the inset table, both our attention biasing and CFG achieve similar improvements over regular diffusion rollouts in transition success rates [6]. However, our method incurs much less computational overhead than CFG, which requires two model evaluations in each diffusion step.

Method	Succ. Rate ↑	Inf. Time ↓
None	83.8%	7.69 ms
CFG	98.7%	14.30 ms
Bias	98.6%	8.50 ms

Velocity Correction We evaluate how closely the generated motion adheres to the intended trajectories, with and without velocity correction. To that end, we deploy the diffusion model with velocity commands corresponding to a number of predefined trajectories, such as squares, circles, figure eights, and randomized velocity commands. The quantitative results, reported in the inset table, highlight that our correction scheme significantly improves adherence to the given commands, indicated by the reduced drift and mean velocity error. We refer to Appendix A.3.1 and Fig. A.8 for qualitative results and details on the test trajectories.

Velocity Correction	Drift per meter ↓	Mean vel. error ↓
✗	0.117	$0.271\,\mathrm{m\,s^{-1}}$
✓	0.003	$0.075\,\mathrm{m\,s^{-1}}$

5.3 Physics-Based Tracking

RL Tracking Policy We evaluate how the choice of training data affects the RL policy's tracking of diffusion-generated motions. Tab. 3 shows that training a policy on the original, non-diffusion-generated data has a significantly higher failure rate (falls per minute) when tracking diffusion output than when training on generated data. This happens because the policy trained on the diffusion output learns to correct artifacts, such as foot sliding, yielding improved stability at the cost of higher tracking errors and worse motion quality. We further compare our open-loop approach, where the diffusion model is conditioned on its own past predictions, with a closed-loop approach, where the diffusion model is conditioned on states estimated from the robot. The open-loop approach leads to better stylization since, in closed loop, the past simulated states differ in distribution from the past motions the diffusion model was trained on. Closed-loop execution leads to lower tracking errors, because the tracking errors enter the diffusion conditioning, pulling the generated motion

toward what the policy executes rather than toward the intended style. Finally, we apply perturbation forces of $600\,\mathrm{N}$ to the character root, resulting in lower motion quality, higher tracking errors, and significantly more failures for both open and closed loop. Open loop nevertheless remains more stable than closed loop. Recovery states can enter the diffusion conditioning in closed-loop mode, pushing the diffusion model out of distribution. In contrast, with an open-loop deployment the diffusion model's generation stream stays uncorrupted and provides clean recovery targets for the policy. We found no degradation of motion quality over the whole time horizon of 20 minutes per rollout. The tracking evaluations discussed so far were performed for styles contained in the training set of both the diffusion model and RL policy. For styles that are generated by interpolating between style latents of training styles, as seen in Fig. 3, tracking performance stays consistent at an MPJAE of $7.9\,\mathrm{deg}$ and 0 falls. FID and style accuracy cannot be computed for such interpolated styles, as no ground-truth data exists.

Transfer to Physical Robot We apply our method to a Unitree G1 humanoid robot ($1.32\,\mathrm{m}$, $35\,\mathrm{kg}$, 29 DoF). After retargeting the human data to the G1 with ReActor [51], we train our model and deploy it zero-shot on the robot. Qualitative results, provided in Fig. 4 and our supplemental video, show that a single controller can robustly track diverse motion styles using different joystick inputs. The robustness of the open-loop mode carries over to hardware, including under disturbances. Throughout $\sim 4\,\mathrm{h}$ of deployments, the robot successfully executed various trajectories and velocity commands across different styles without any failures. The diffusion model runs in real time on an onboard NVIDIA Jetson Thor at $\sim 11\,\mathrm{ms}$ per inference step and $\sim 17\,\mathrm{ms}$ command-to-motion output latency, compared to $8\,\mathrm{ms}/12\,\mathrm{ms}$ for simulation on an NVIDIA RTX 5090.

6 Limitations

While our work presents a framework for general, stylized locomotion control, several limitations remain. The RL policy can correct artifacts of the diffusion model (foot sliding, hovering, ground penetration). However, there can be a mismatch between the kinematic motions and the commanded velocity, which manifests as the tracking policy taking extra in-between steps to maintain stability. Introducing implicit rewards to achieve natural gaits could present a solution to these issues [1, 2]. Our method works with styles that are agnostic to the direction of movement, such as flailing the arms or walking in a wide stance. For styles that are tightly coupled to the direction of movement, such as a moonwalk or cartwheels, our method requires matching velocity commands. Furthermore, styles that are largely described through a distinct root trajectory, such as rotating clockwise while walking, specify the same quantity as the joystick, creating an inherent conflict in the interface, which we resolve in favor of the user command. Finally, our current framework lets the model learn the style implicitly. A more explicit parameterization could allow users to partially stylize, for example, focusing only on the lower or upper body behavior of a style reference.

7 Conclusion

We presented a real-time system for expressive legged locomotion that enables simultaneous user control over velocity and style via an example-driven, autoregressive diffusion model. Through a style encoder that is trained with an effective sampling strategy on short, unstructured motion clips rather than fixed style labels, we enable physics-based characters to generalize to completely unseen styles, including those extracted from real-world videos. This flexible interface opens the door to various other style input modalities, such as artist-created animations.

Acknowledgments

We would like to thank Ioannis Dadiotis for his support with robot deployments.

References

[1] Y. Mu, Z. Zhang, Y. Shi, D. Yang, M. Matsumoto, K. Imamura, G. Tevet, C. Guo, M. Taylor, C. Shu, P. Xi, and X. B. Peng. SMP: Reusable Score-Matching Motion Priors for Physics-Based Character Control. *ACM Trans. Graph.*, 45(4), July 2026. ISSN 0730-0301. doi:10.1145/3811282.

[2] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *ACM Trans. Graph.*, 40(4), July 2021. ISSN 0730-0301. doi:10.1145/3450626.3459670.

[3] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler. ASE: Large-Scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters. *ACM Trans. Graph.*, 41(4), July 2022. ISSN 0730-0301. doi:10.1145/3528223.3530110.

[4] R. Grandia, E. Knoop, M. A. Hopkins, G. Wiedebach, J. Bishop, S. Pickles, D. Müller, and M. Bächer. Design and Control of a Bipedal Robotic Character. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi:10.15607/RSS.2024.XX.103.

[5] M. Piseno, Z. Li, A. Escontrela, P. Abbeel, K. Sreenath, X. B. Peng, and A. Iscen. Stylized Locomotion Control for Bipedal Humanoid Robots with Adversarial Motion Priors, 2025. URL `https://www.michaelpiseno.com/biped_ctrl.github.io/`. Project page.

[6] R. Chen, M. Shi, S. Huang, P. Tan, T. Komura, and X. Chen. Taming Diffusion Probabilistic Models for Character Control. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705250. doi:10.1145/3641519.3657440.

[7] K. Aberman, Y. Weng, D. Lischinski, D. Cohen-Or, and B. Chen. Unpaired Motion Style Transfer from Video to Animation. *ACM Trans. Graph.*, 39(4), Aug. 2020. ISSN 0730-0301. doi:10.1145/3386569.3392469.

[8] D.-K. Jang, S. Park, and S.-H. Lee. Motion Puzzle: Arbitrary Motion Style Transfer by Body Part. *ACM Trans. Graph.*, 41(3), June 2022. ISSN 0730-0301. doi:10.1145/3516429.

[9] L. Zhong, Y. Xie, V. Jampani, D. Sun, and H. Jiang. SMooDi: Stylized Motion Diffusion Model. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part I*, pages 405–421, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73231-7. doi:10.1007/978-3-031-73232-4_23.

[10] A. Serifi, R. Grandia, E. Knoop, M. Gross, and M. Bächer. Robot Motion Diffusion Model: Motion Generation for Robotic Characters. In *SIGGRAPH Asia 2024 Conference Papers*, SA '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400711312. doi:10.1145/3680528.3687626.

[11] G. Tevet, S. Raab, S. Cohan, D. Reda, Z. Luo, X. B. Peng, A. H. Bermano, and M. van de Panne. CLoSD: Closing the Loop between Simulation and Diffusion for Multi-Task Character Control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=pZISppZSTv`.

[12] Y. Yuan, J. Song, U. Iqbal, A. Vahdat, and J. Kautz. PhysDiff: Physics-Guided Human Motion Diffusion Model. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. doi:10.1109/ICCV51070.2023.01467.

[13] M. Xu, Y. Shi, K. Yin, and X. B. Peng. PARC: Physics-Based Augmentation with Reinforcement Learning for Character Controllers. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, SIGGRAPH Conference Papers '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715402. doi:10.1145/3721238.3730616.

[14] J. Ho and T. Salimans. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

[15] W. Song, X. Jin, S. Li, C. Chen, A. Hao, X. Hou, N. Li, and H. Qin. Arbitrary Motion Style Transfer with Multi-Condition Motion Latent Diffusion Model. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 821–830, 2024. doi:10.1109/CVPR52733.2024.00084.

[16] B. Kim, J. Kim, H. J. Chang, and J. Y. Choi. MoST: Motion Style Transformer between Diverse Action Contents. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1705–1714, 2024. doi:10.1109/CVPR52733.2024.00168.

[17] H. Y. Au, S. Wang, H. Sheng, J. Jiang, and J. Chen. Motion AutoStyling via Seperable Modelling of Posing and Dynamic LoRAs. In *Proceedings of the SIGGRAPH Asia 2025 Technical Communications*, SA Technical Communications '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400721366. doi:10.1145/3757376.3771387.

[18] S. Raab, I. Gat, N. Sala, G. Tevet, R. Shalev-Arkushin, O. Fried, A. H. Bermano, and D. Cohen-Or. Monkey See, Monkey Do: Harnessing Self-Attention in Motion Diffusion for Zero-Shot Motion Transfer. In *SIGGRAPH Asia 2024 Conference Papers*, SA '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400711312. doi:10.1145/3680528.3687579.

[19] L. Zhong, Y. Yang, and C. Li. SMooGPT: Stylized Motion Generation Using Large Language Models, 2025. URL https://arxiv.org/abs/2509.04058.

[20] S. Xia, C. Wang, J. Chai, and J. Hodgins. Realtime Style Transfer for Unlabeled Heterogeneous Human Motion. *ACM Trans. Graph.*, 34(4), July 2015. ISSN 0730-0301. doi:10.1145/2766999.

[21] G. E. Henter, S. Alexanderson, and J. Beskow. MoGlow: Probabilistic and Controllable Motion Synthesis Using Normalising Flows. *ACM Trans. Graph.*, 39(6), Nov. 2020. ISSN 0730-0301. doi:10.1145/3414685.3417836.

[22] S. Starke, I. Mason, and T. Komura. DeepPhase: Periodic Autoencoders for Learning Motion Phase Manifolds. *ACM Trans. Graph.*, 41(4), July 2022. ISSN 0730-0301. doi:10.1145/3528223.3530178.

[23] I. Mason, S. Starke, and T. Komura. Real-Time Style Modelling of Human Locomotion via Feature-Wise Transformations and Local Motion Phases. *Proc. ACM Comput. Graph. Interact. Tech.*, 5(1), May 2022. doi:10.1145/3522618.

[24] H. Zhang, S. Starke, T. Komura, and J. Saito. Mode-Adaptive Neural Networks for Quadruped Motion Control. *ACM Trans. Graph.*, 37(4), July 2018. ISSN 0730-0301. doi:10.1145/3197517.3201366.

[25] S. Starke, P. Starke, N. He, T. Komura, and Y. Ye. Categorical Codebook Matching for Embodied Character Controllers. *ACM Trans. Graph.*, 43(4), July 2024. ISSN 0730-0301. doi:10.1145/3658209.

[26] Y. Shi, J. Wang, X. Jiang, B. Lin, B. Dai, and X. B. Peng. Interactive Character Control with Auto-Regressive Motion Diffusion Models. *ACM Trans. Graph.*, 43(4), July 2024. ISSN 0730-0301. doi:10.1145/3658140.

[27] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-World Humanoid Locomotion with Reinforcement Learning. *Science Robotics*, 9(89):eadi9579, 2024. doi:10.1126/scirobotics.adi9579.

[28] X. Gu, Y.-J. Wang, X. Zhu, C. Shi, Y. Guo, Y. Liu, and J. Chen. Advancing Humanoid Locomotion: Mastering Challenging Terrains with Denoising World Model Learning. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi:10.15607/RSS.2024.XX.058.

[29] Z. Zhuang, S. Yao, and H. Zhao. Humanoid Parkour Learning. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 1975–1991. PMLR, 06–09 Nov 2024. URL https://proceedings.mlr.press/v270/zhuang25a.html.

[30] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement Learning for Versatile, Dynamic, and Robust Bipedal Locomotion Control. *The International Journal of Robotics Research*, 44(5):840–888, 2025. doi:10.1177/02783649241285161.

[31] Z. Luo, Y. Yuan, T. Wang, C. Li, F. Castañeda, S. Chen, Z.-A. Cao, J. Li, D. Minor, Q. Ben, J. Park, D. Sami, Z. Wang, X. Da, R. Ding, C. Hogg, L. Song, E. Lim, E. Jeong, T. He, H. Xue, W. Xiao, S. Yuen, J. Kautz, Y. Chang, U. Iqbal, L. J. Fan, and Y. Zhu. SONIC: Supersizing Motion Tracking for Natural Humanoid Whole-Body Control. *Science Robotics*, 11(117): eaed4592, 2026. doi:10.1126/scirobotics.aed4592.

[32] J. Long, J. Ren, M. Shi, Z. Wang, T. Huang, P. Luo, and J. Pang. Learning Humanoid Locomotion with Perceptive Internal Model. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9997–10003, 2025. doi:10.1109/ICRA55743.2025.11128333.

[33] A. Serifi, R. Grandia, E. Knoop, M. Gross, and M. Bächer. VMP: Versatile Motion Priors for Robustly Tracking Motion on Physical Characters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '24, pages 1–11, Goslar, DEU, 2024. Eurographics Association. doi:10.1111/cgf.15175.

[34] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Trans. Graph.*, 37(4), July 2018. ISSN 0730-0301. doi:10.1145/3197517.3201311.

[35] J. Won, D. Gopinath, and J. Hodgins. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Trans. Graph.*, 39(4), Aug. 2020. ISSN 0730-0301. doi:10.1145/3386569.3392381.

[36] Z. Luo, J. Cao, A. Winkler, K. Kitani, and W. Xu. Perpetual Humanoid Control for Real-time Simulated Avatars. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. doi:10.1109/ICCV51070.2023.01000.

[37] C. Tessler, Y. Kasten, Y. Guo, S. Mannor, G. Chechik, and X. B. Peng. CALM: Conditional Adversarial Latent Models for Directable Virtual Characters. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701597. doi:10.1145/3588432.3591541.

[38] D. Müller, E. Knoop, D. Mylonopoulos, A. Serifi, M. A. Hopkins, R. Grandia, and M. Bächer. Olaf: Bringing an Animated Character to Life in the Physical World. *IEEE Robotics and Automation Letters*, 11(6):7580–7587, 2026. doi:10.1109/LRA.2026.3685938.

[39] M. A. Hopkins, G. Wiedebach, K. Cesare, J. Bishop, E. Knoop, and M. Bächer. Interactive Design of Stylized Walking Gaits for Robotic Characters. *ACM Trans. Graph.*, 43(4), July 2024. ISSN 0730-0301. doi:10.1145/3658227.

[40] X. Huang, T. Truong, Y. Zhang, F. Yu, J. P. Sleiman, J. Hodgins, K. Sreenath, and F. Farshidian. Diffuse-CLoC: Guided Diffusion for Physics-based Character Look-ahead Control. *ACM Trans. Graph.*, 44(4), July 2025. ISSN 0730-0301. doi:10.1145/3731206.

[41] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-Or, and A. H. Bermano. Human Motion Diffusion Model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SJ1kSyO2jwu.

[42] M. Zhang, Z. Cai, L. Pan, F. Hong, X. Guo, L. Yang, and Z. Liu. MotionDiffuse: Text-Driven Human Motion Generation with Diffusion Model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46 (6):4115–4128, June 2024. ISSN 0162-8828. doi:10.1109/TPAMI.2024.3355414.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. 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. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[44] T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(12):4217–4228, Dec. 2021. ISSN 0162-8828. doi:10.1109/TPAMI.2020.2970919.

[45] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski. Vision Transformers Need Registers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2dnO3LLiJ1.

[46] Z. Zhang, K. Wen, M. Xu, J. He, C. Li, T. Miki, C. Schwarke, C. Zhang, X. B. Peng, and M. Hutter. Learning Whole-Body Humanoid Locomotion via Motion Generation and Motion Tracking. *IEEE Robotics and Automation Letters*, 11(9):10194–10201, 2026. doi:10.1109/LRA.2026.3710365.

[47] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. 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, pages 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964. URL https://dl.acm.org/doi/10.5555/3295222.3295408.

[48] Z. Shen, H. Pi, Y. Xia, Z. Cen, S. Peng, Z. Hu, H. Bao, R. Hu, and X. Zhou. World-Grounded Human Motion Recovery via Gravity-View Coordinates. In *SIGGRAPH Asia 2024 Conference Papers*, SA '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400711312. doi:10.1145/3680528.3687565.

[49] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. AMASS: Archive of Motion Capture as Surface Shapes. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5441–5450, 2019. doi:10.1109/ICCV.2019.00554.

[50] R. A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188, 1936. doi:10.1111/j.1469-1809.1936.tb02137.x.

[51] D. Müller, A. Serifi, S. Christen, R. Grandia, E. Knoop, and M. Bächer. ReActor: Reinforcement Learning for Physics-Aware Motion Retargeting. *ACM Trans. Graph.*, 45(4), July 2026. ISSN 0730-0301. doi:10.1145/3811378.

[52] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the Continuity of Rotation Representations in Neural Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5738–5746, 2019. doi:10.1109/CVPR.2019.00589.

[53] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*, 2015. URL https://arxiv.org/abs/1412.6980.

[54] P. Dobrowolski. Swing-Twist Decomposition in Clifford Algebra, 2015. URL https://arxiv.org/abs/1506.05481.

[55] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[56] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018. doi:10.1109/ICRA.2018.8460528.

[57] Y. Wang, S. Zhu, P. Zhi, Y. Li, J. Li, Y.-L. Li, Y. Xiao, X. Wang, B. Jia, and S. Huang. OmniXtreme: Breaking the Generality Barrier in High-Dynamic Humanoid Control, 2026. URL https://arxiv.org/abs/2602.23843.

A Appendix

A.1 Kinematic Diffusion Model

Motion Representation We represent the motion state, similarly to CAMDM [6], as

$$\mathbf{m} = (\mathbf{x}, \mathbf{r}, \mathbf{J}, \mathbf{X}, \mathbf{R}), \tag{6}$$

with the position $\mathbf{x} \in \mathbb{R}^3$ and orientation $\mathbf{r} \in \mathbb{R}^6$ of the character's root, the orientations of all its N_J joints $\mathbf{J} \in \mathbb{R}^{N_J \times 6}$, and the positions $\mathbf{X} \in \mathbb{R}^{N_B \times 3}$ and orientations $\mathbf{R} \in \mathbb{R}^{N_B \times 6}$ of all its N_B links. We represent all rotations in 6D [52]. For joint orientations, we represent a child's orientation in the kinematic chain by its orientation in the parent's coordinate frame.

To train a model that is agnostic to the character's global state, we transform all quantities, except the joint orientations, in $\mathbf{M}_{\text{pst}}$ and $\mathbf{M}_{\text{fut}}$, to the heading frame of the character, extracted from the current state $\mathbf{m}_t$. The origin of this frame is set to the character's ground-projected root position, its x-axis points in the direction of the root's yaw, and the z-axis is aligned with the global up vector.

Model and Training Parameters Both the diffusion model $\mathcal{M}$ and style encoder $\mathcal{E}$ are realized as transformer-encoders [43] with the same model parameters which can be seen in the inset table. These parameters are used across all experiments for both the humanoid and robotic characters. We train all models with 8 diffusion steps with a cosine noise schedule. Training is performed for 100 epochs and the model weights are optimized using Adam [53] with default parameters except for the learning rate, which is set to 10^{-4}. During training we randomly mask past motion by setting it to 0 with probability 0.15. For the humanoid character we represent motion at 30 frames per second and set the future frames $F = 20$ with past frames $P = 10$. The robotic character uses 50 frames per second with $F = 30$ and $P = 15$. To ensure responsiveness to velocity commands, we trigger early autoregression after 10 and 15 frames, respectively, if the velocity commands change. For both characters, we sample style clips of lengths 1 s to 5 s. An important consideration is that the humanoid uses spherical joints while the robot uses revolute joints. For the diffusion model itself, we treat both joint types as if they were spherical joints, which allows the diffusion model to predict infeasible orientations for the revolute joints. We then post-process the diffusion model predictions by projecting the predicted orientations onto the set of feasible orientations using a swing-twist decomposition [54].

Parameter	Value
d_{model}	256
FeedForward Size	1024
# Layers	4
# Heads	4
Dropout Rate	0.2
Activation Fn.	GELU

Motion Smoothing To alleviate discontinuities between consecutive diffusion predictions, we apply a smoothing scheme to generate motion. Specifically, rather than generating a new motion every F frames, we trigger the diffusion model after only $F - K$ frames, creating a K-frame overlap between consecutive predictions, as visualized in Fig. A.1. Then, the overlapping frames are blended together, using linear and spherical interpolation for positions and rotations, respectively. In practice, we set $K = 3$.

Velocity Correction For the feedback gains we set $k_{\mathbf{p}} = k_{\psi} = \frac{1}{F\Delta t}$ where F is the number of future frames predicted by the diffusion model and Δt is the duration of each timestep, i.e., the inverse frame rate. This ensures that the error is corrected over the span of one diffusion prediction, without under-correcting or overshooting. Finally, we use the mean yaw error over the last P frames when computing the feedback update. This prevents the error correction itself from depending on when exactly in the gait cycle diffusion is triggered.

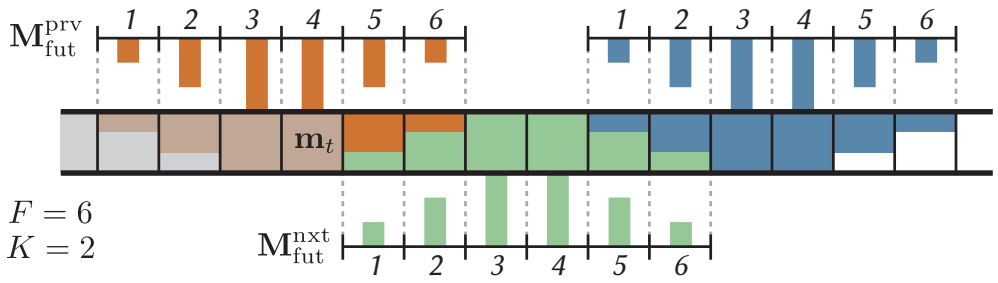

Figure A.1: **Smoothing Scheme.** To achieve smooth transitions between consecutive diffusion calls, we predict the next F states $\mathbf{M}^{\text{nxt}}_{\text{fut}}$ at the current timestep t, then blend the first K states in $\mathbf{M}^{\text{nxt}}_{\text{fut}}$ with the last K states of the previous prediction $\mathbf{M}^{\text{prv}}_{\text{fut}}$, then copy the next $F - 2K$ states from $\mathbf{M}^{\text{nxt}}_{\text{fut}}$ before incrementing the timestep $t := t + (F - K)$ and repeating the process. More formally, we weight state $\mathbf{m}^{\text{nxt}}_i$ by $w = {}^{i}/_{(K+1)}$ and state $\mathbf{m}^{\text{prv}}_{(F-K)+i}$ by $1 - w$ to form the blended states $\mathbf{m}_{t+i}$ for $i = 1, \ldots, K$. The visualization shows the $F = 6, K = 2$ case.

A.2 Reinforcement Learning

We use RL to train a policy $\pi(\mathbf{a}_t \,|\, \mathbf{o}_t, \mathbf{g}_t)$ to track the generated motion. The policy predicts joint setpoints $\mathbf{a}_t$ for proportional-derivative (PD) controllers based on observations $\mathbf{o}_t$ and goal conditioning $\mathbf{g}_t$. The conditioning $\mathbf{g}_t$ contains the character's target state at timestep t as predicted by the diffusion model, converted into the character's heading frame of step $t - 1$. Additionally, we explicitly concatenate the joystick command to the target state. Proprioceptive observations consist of joint positions and velocities, root angular velocity, projected gravity and a history of previous actions.

Observations and Goals In our tracking policy, we use the goal condition

$$\mathbf{g}_t := (\mathbf{x}_t, \mathbf{r}_t, \mathbf{q}_t, \mathbf{X}_t, \mathbf{R}_t, \dot{\mathbf{x}}_t, \boldsymbol{\omega}_t, \tilde{\mathbf{v}}_t), \tag{7}$$

containing the set of reference motion features $\mathbf{m}_t$ (see Eq. (6)). However, note that here we represent the joint angles in minimal coordinates $\mathbf{q}_t$ instead of using $\mathbf{J}_t$, as is common for the simulation of articulated bodies. For the robot with revolute joints, $\mathbf{J}$ is projected to a feasible $\mathbf{q}_t$. In addition, $\mathbf{g}_t$ includes the target root linear velocity $\dot{\mathbf{x}}_t$ and angular velocity $\boldsymbol{\omega}_t$, computed from $\mathbf{M}_{\text{fut}}$, as well as the velocity command after correction $\tilde{\mathbf{v}}_t$ (Sec. 4.2). All rigid body positions and orientations are expressed in the root's heading frame, making the policy independent of the global state. The proprioceptive observations $\mathbf{o}_t$ are

$$\mathbf{o}_t := (\hat{\mathbf{q}}_t, \hat{\dot{\mathbf{q}}}_t, \hat{\boldsymbol{\omega}}_t, \hat{\mathbf{g}}^{\text{proj}}_t, \mathbf{a}_{t-1}, \mathbf{a}_{t-2}), \tag{8}$$

where the hat symbol $\hat{(\cdot)}$ indicates measured values from the simulator or the real system. These observations include the projected gravity $\hat{\mathbf{g}}^{\text{proj}}_t$, angular velocity $\hat{\boldsymbol{\omega}}_t$, joint angles $\hat{\mathbf{q}}_t$, joint velocities $\hat{\dot{\mathbf{q}}}_t$, as well as the previous two actions $\mathbf{a}_{t-1}$ and $\mathbf{a}_{t-2}$.

Rewards We use a combination of three groups of reward functions, consisting of tracking rewards $\mathbf{r}_{\text{track}}$, regularization rewards $\mathbf{r}_{\text{reg}}$, and task rewards $\mathbf{r}_{\text{task}}$. Tracking rewards ensure that the policy follows the prescribed motion, while the regularization rewards ensure that it does so in a smooth and natural manner that is deployable on real hardware. The task reward incentivizes following the joystick commands and helps with resolving unphysical target states, for example when the diffusion model generates motion with foot sliding. In such cases, the task reward encourages the policy to give less priority to exact tracking, resulting in more natural and stable motion. Tab. A.1 shows all the individual reward terms with their respective weights.

Training Environment We follow standard practices in the training of tracking controllers [33, 34]. During each episode we sample a random motion from the training set and iterate

Table A.1: **RL Reward Terms.** Reward terms and weights for the Humanoid and the G1 robot. Tracking and joystick rewards use an exponential kernel $\exp(-e/\sigma)$ on the listed squared error e. We drop the time index t for conciseness. Norms are Euclidean unless specified otherwise. h_f measures the foot-ground distance while $c_f^{\text{sim}}, c_f^{\text{ref}}$ are simulation or reference provided contact flags.

Name	**Error / Term**	σ	**Humanoid**	**G1 Robot**
	Tracking			
Root position	$\|\mathbf{x} - \hat{\mathbf{x}}\|^2$	0.1	1.0	2.0
Root orientation	$\|\mathbf{r} \ominus \hat{\mathbf{r}}\|_F^2$	0.5	1.0	2.0
Root lin. velocity	$\|\dot{\mathbf{x}} - \hat{\dot{\mathbf{x}}}\|^2$	1.0	1.0	2.0
Root ang. velocity	$\|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}\|^2$	1.0	1.0	2.0
Upper body joints	$\|\mathbf{q}^u - \hat{\mathbf{q}}^u\|^2$	0.5	5.0	5.0
Lower body joints	$\|\mathbf{q}^l - \hat{\mathbf{q}}^l\|^2$	0.5	1.0	3.0
Ankle joints	$\|\mathbf{q}^a - \hat{\mathbf{q}}^a\|^2$	1.0	1.0	1.0
Body positions	$\|\mathbf{X} - \hat{\mathbf{X}}\|^2$	1.0	1.0	1.0
Body orientations	$\|\mathbf{R} \ominus \hat{\mathbf{R}}\|_F^2$	2.5	1.0	1.0
Body lin. velocity	$\|\dot{\mathbf{X}} - \hat{\dot{\mathbf{X}}}\|^2$	5.0	–	2.0
Alive bonus	1	–	1.0	2.0
	Joystick commands			
Joystick lin. velocity	$\|\tilde{\mathbf{v}}_{x,y} - \hat{\mathbf{v}}_{x,y}\|^2$	0.25	5.0	3.0
Joystick yaw rate	$(\tilde{v}_\psi - \hat{v}_\psi)^2$	0.25	5.0	5.0
Lin. velocity direction	$\cos\angle(\tilde{\mathbf{v}}_{x,y}, \hat{\mathbf{v}}_{x,y}) \cdot \mathbb{1}[\|\tilde{\mathbf{v}}_{x,y}\| > 0.2]$	–	–	5.0
Yaw rate direction	$\text{sgn}(\tilde{v}_\psi)\,\text{sgn}(\hat{v}_\psi) \cdot \mathbb{1}[\lvert\tilde{v}_\psi\rvert > 0.2]$	–	–	5.0
Stuck (lin. velocity)	$\mathbb{1}[\|\hat{\mathbf{v}}_{x,y}\| < 0.2 \wedge \|\tilde{\mathbf{v}}_{x,y}\| > 0.2]$	–	–	-10.0
Stuck (yaw rate)	$\mathbb{1}[\lvert\hat{v}_\psi\rvert < 0.2 \wedge \lvert\tilde{v}_\psi\rvert > 0.2]$	–	–	-10.0
	Regularization			
Action rate	$\|\mathbf{a}_t - \mathbf{a}_{t-1}\|^2$	–	-0.01	-0.1
Action accel.	$\|\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\|^2$	–	-0.01	-0.05
Joint accel.	$\|\ddot{\mathbf{q}}\|^2$	–	-10^{-5}	-10^{-6}
Torque	$\|\boldsymbol{\tau}\|^2$	–	-10^{-5}	-10^{-5}
Joint position limits	$\sum_j \max(0,\ \lvert q_j\rvert - q_j^{\max})$	–	–	-10.0
Joint velocity limits	$\sum_j \max(0,\ \lvert\dot{q}_j\rvert - 0.95\,\dot{q}_j^{\max})$	–	–	-1.0
Foot impact	$\sum_f \max(0,\ \|\mathbf{F}_f\| - 400)$	–	–	-0.01
Simulation-contact slip	$\sum_f \|\hat{\dot{\mathbf{X}}}_{f,xy}\| \cdot c_f^{\text{sim}}$	–	–	-0.3
Reference-contact slip	$\sum_f \|\hat{\dot{\mathbf{X}}}_{f,xy}\| \cdot c_f^{\text{ref}}$	–	–	-5.0
Swing clearance	$\sum_f (1 - c_f^{\text{ref}}) \cdot \max(0,\ 0.02 - h_f)$	–	–	-200.0

through it to get the individual goals of each timestep. Here, we also randomly sample the starting frame inside the goal motion and combine it with reference state initialization where the character is initialized in that exact state [34]. To allow the policy to reach any target state from any starting state we choose the reference state during initialization from a different motion than the actual target motion with a probability of 0.2. Episodes terminate early if the pelvis or head contacts the ground. The policy itself is trained with PPO [55] with the hyperparameters given in the inset table.

Parameter	Value
$\varepsilon_{\text{clip}}$	0.2
Value Loss Coefficient	1.0
Entropy Coefficient	0.005
Discount Factor γ	0.99
GAE Factor λ	0.99
Desired KL Divergence	0.01
# Environments	4096
Env. Steps per PPO Iter.	32
Learning Epochs per PPO Iter.	5
# Mini Batches	4
Schedule	Adaptive

Training Data To generate the training data for the policy we generate a large-scale dataset using diffusion model rollouts with randomized commands. For each clip of motion in the training set of the diffusion model, we generate 10 minutes

Table A.2: **Domain Randomization.** Randomization ranges applied during policy training. A dash indicates that the term is not randomized for the respective embodiment.

Term	Unit	Humanoid	Robot
Additive Observation Noise			
Projected Gravity	rad	–	$\mathcal{U}(-0.05, 0.05)$
Root Angular Velocity	$\mathrm{rad\,s^{-1}}$	–	$\mathcal{U}(-0.2, 0.2)$
Joint Position	rad	–	$\mathcal{U}(-0.01, 0.01)$
Joint Velocity	$\mathrm{rad\,s^{-1}}$	–	$\mathcal{U}(-1.5, 1.5)$
Physics Randomization			
Root Mass Offset	kg	–	$\mathcal{U}(-1.0, 3.0)$
Root/Link CoM Offset	m	–	$\mathcal{U}(-0.05, 0.05)$
Foot Friction	–	$\mathcal{U}(0.4, 0.6)$	$\mathcal{U}(0.2, 1.0)$
K_p, K_d scale	–	–	$\mathcal{U}(0.9, 1.1)$
Disturbances			
Push Root	$\mathrm{m\,s^{-1}}$	$\mathcal{U}(-0.5, 0.5)$	$\mathcal{U}(-0.5, 0.5)$
Push Ankle Force	N	5.0	5.0
Push Ankle Torque	$\mathrm{N\,m}$	2.5	2.5

of motion. To do so, we randomly resample velocity commands every $1.5\,\mathrm{s}$ to $4.5\,\mathrm{s}$ and randomly resample the chosen style reference window from the clip every $5\,\mathrm{s}$. Additionally, we generate another set of 10 hours of motion where we sample velocity commands in the same manner while also randomly sampling the style reference over the whole training set, to ensure that the policy can transition between different styles.

Robustness and Sim-to-Real In order to improve the robustness of the policies and ensure feasibility of transferring them to the real world we apply disturbances and domain randomization [56] with their respective values shown in Tab. A.2. A curriculum is applied to the magnitude of disturbances, which ramps linearly from 0 to the specified maximum values between PPO iterations 300 and 2000. Finally, the policies are trained on rough terrain where height field values in the range $[-2, 2]\ \mathrm{cm}$ are uniformly sampled and placed on a grid with spacing of $10\,\mathrm{cm}$. The grid points are then linearly connected to generate the ground terrain. For the G1 robot, we use the same actuator parameters as in [57].

A.3 Evaluations

Evaluation Setup We train the diffusion model for 100 epochs. When training on 100STYLE [23] this corresponds to roughly two days of training on an NVIDIA RTX 4090. With the inclusion of AMASS [49], training time increases to roughly five days. To ensure convergence, the RL policies are trained for five days on an NVIDIA RTX 4090. However, reasonable performance is already achieved after a few hours of training. For all experiments involving attention biasing we set $w_{\text{syl}} = 2.0, w_{\text{pst}} = -1.0$ and for CFG we set $\gamma_{\text{CFG}} = 0.6$. To assess the diffusion model quality, we generate 20 minutes of motion for each of the 100 styles in 100STYLE. For the transitions, we generate 5 transition motions from each style to every other style in 100STYLE. Here, we first generate $5\,\mathrm{s}$ of motion in style i, then switch the style reference to style j and generate another $5\,\mathrm{s}$ of motion. Velocity commands are randomly sampled for both the diffusion model and transition evaluations. For the evaluations of the RL policies we generate and track 20 hours of motions with randomized commands.

Evaluation Metrics For the evaluation of the diffusion model we rely on a number of metrics to assess motion quality, adherence to commands as well as physicality of the motion. First of all, we use the Fréchet Inception Distance (FID) [47] as a measure of distributional similarity between

original MoCap and generated motions. The FID value is computed as

$$\text{FID} = \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\|_2^2 + \text{Tr}\left(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2(\boldsymbol{\Sigma}_r\boldsymbol{\Sigma}_g)^{\frac{1}{2}}\right) \tag{9}$$

where $\boldsymbol{\mu}_r, \boldsymbol{\mu}_g$ and $\boldsymbol{\Sigma}_r, \boldsymbol{\Sigma}_g$ are empirical means and covariances of latent features of the real and generated data respectively. We extract these latent features from the activations of a classification model. This classification model is trained to classify $0.25\,\text{s}$ motion windows of the original 100STYLE dataset into its corresponding style and we use the final activations, prior to the classification head of the model, as latent representations. Importantly, FID score is computed on a per-style basis so that it captures stylistic features of motion rather than just generic similarity to the MoCap clips. Style accuracy is computed as the percentage of generated motions for which the style classifier predicts the style label that corresponds to the motion that was used as style reference. Position and yaw errors are simply computed as the difference between predicted position/yaw and their imposed condition counterparts. We compute foot sliding by measuring foot movement under estimated contact where a foot is classified as in contact if its heel height is less than $1\,\text{cm}$. For transitions we compute the success rate based on the style classifier: when the model should transition from style i to style j, we count it as a success if the style classifier predicts label j after the transition is initiated. We do not impose a maximum transition time, so a transition counts as successful even if style j is only reached by the end of the predicted motion.

CFG In our experiments, we compare against CFG on past motion, where each diffusion step is computed as

$$\mathbf{M}_{\text{fut}}^0 = \gamma_{\text{CFG}}\mathcal{M}(\mathbf{M}_{\text{fut}}^d, d; \mathbf{M}_{\text{pst}}, \mathbf{D}_{\text{pst}}, \mathbf{C}_{\text{fut}}, \mathbf{c}_{\text{syl}}) + (1-\gamma_{\text{CFG}})\mathcal{M}(\mathbf{M}_{\text{fut}}^d, d; \varnothing, \varnothing, \mathbf{C}_{\text{fut}}, \mathbf{c}_{\text{syl}}) \tag{10}$$

where $\gamma_{\text{CFG}} \in [0, 1]$ defines the guidance strength, with lower values prioritizing style adherence over past-motion consistency.

A.3.1 Qualitative Results

In this section, we show more qualitative results of our method.

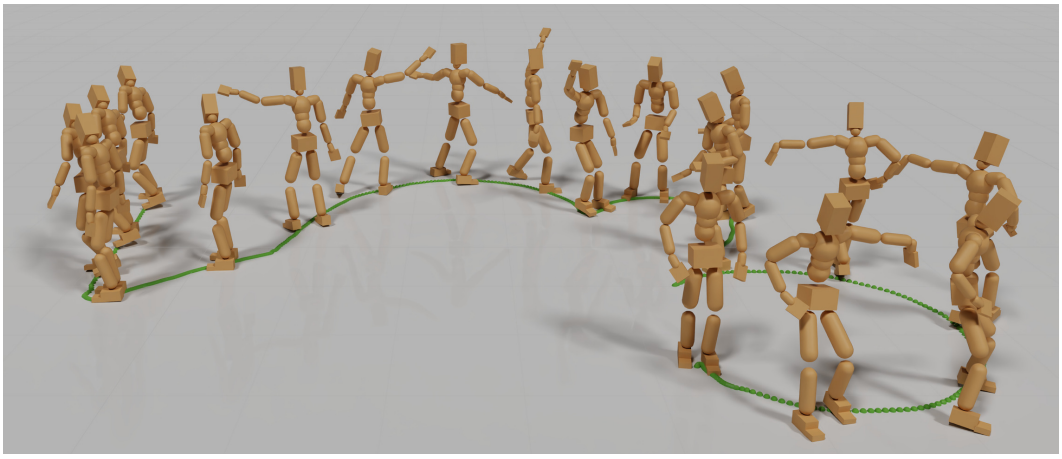

Figure A.2: **Interactive Control.** Example of interactive control where a physics-based humanoid follows velocity commands in various styles. The green dots indicate the trajectory generated from the given velocity commands.

Diffusion Model and Tracking Fig. A.2 shows an example of a longer sequence of motion generated interactively. A user gives joystick and style commands as conditions for the diffusion model's generation which is then tracked by an RL policy.

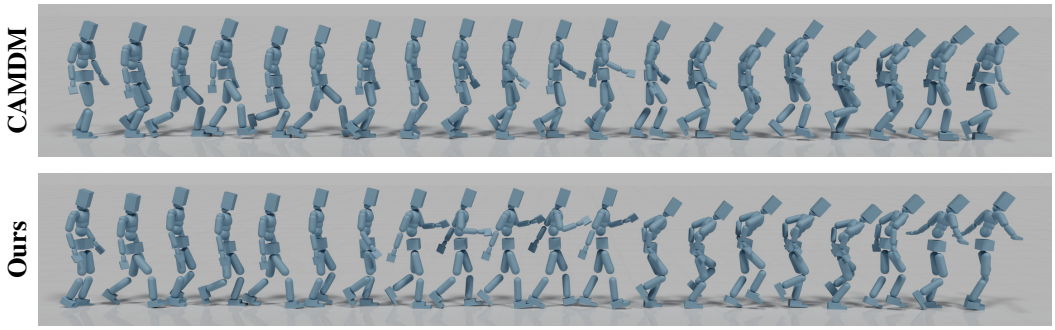

Figure A.3: **Baseline Comparison.** We show two motion sequences generated by the same commands with CAMDM [6] and our model.

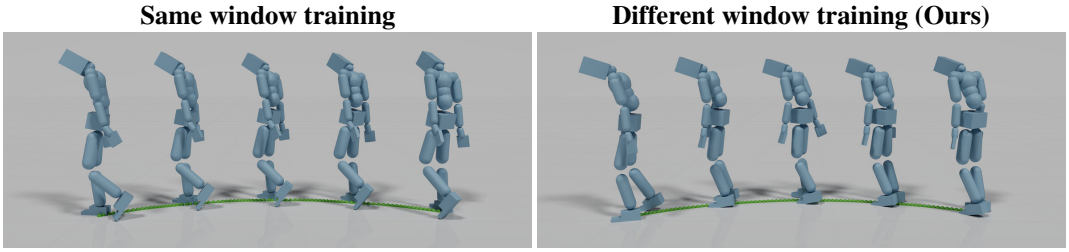

Figure A.4: **Sampling Scheme.** We show the first frame of 5 consecutively predicted motion windows of models trained with and without our proposed sampling scheme. Without our sampling approach ("same window training"), the model collapses and keeps repeating the reference motion. With our sampling approach ("different window training"), the model learns to extract the style characteristic from the reference motion.

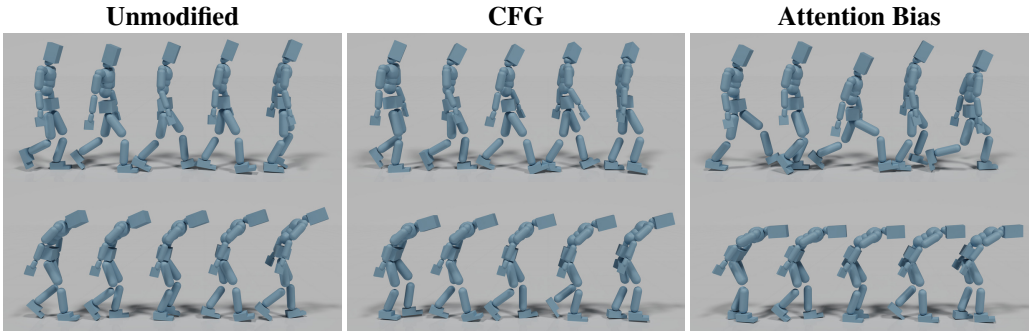

Figure A.5: **Attention Bias Quality.** We compare unmodified inference with CFG and attention biasing for improved stylization.

Qualitative Comparison to CAMDM Fig. A.3 shows a comparison of motions generated by [6] and our model for the same set of joystick commands and style conditions. For CAMDM we select the labels corresponding to the clips we choose as conditioning for our model. We can see that while the style is recognizable for both models, our method performs it more expressively, for example, by reaching the arms out widely in the aeroplane style (right).

Sampling Scheme We generate 5 consecutive predictions with and without our sampling scheme, given a fixed style reference as input. Fig. A.4 shows how the model collapses without our sampling scheme, repeating the same motion at every prediction. With our sampling the model adapts and predicts a continuous motion that follows the general style characteristics.

Attention Bias Fig. A.5 shows qualitative comparisons of generating motions without any inference adjustments, with CFG and with our proposed attention biasing technique. Stylization is

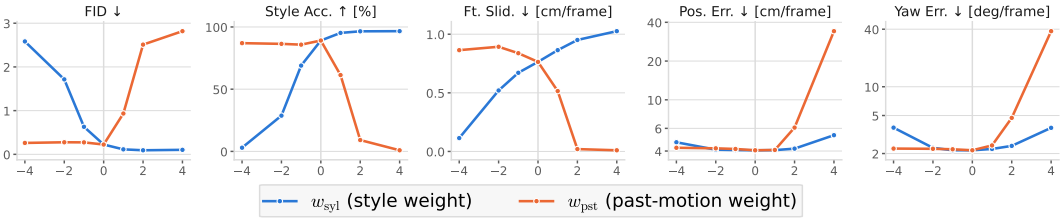

Figure A.6: **Attention Bias Sensitivity.** We show how varying bias parameters affect performance by varying one weight at a time while keeping the other fixed at 0. Increasing either weight to large magnitudes worsens command adherence while increasing generation artifacts. Large positive w_{pst} weights specifically result in frozen poses, which is why foot sliding is at zero.

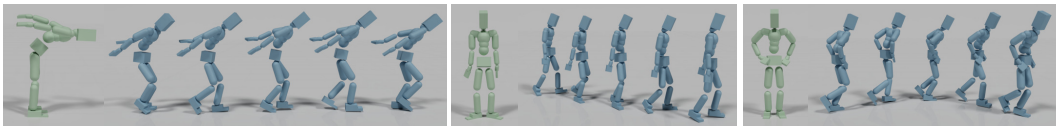

Figure A.7: **Unseen Style.** Generation of motion with unseen styles from 100STYLE, shown as the green snapshots.

noticeably increased when using our biasing method. Additionally, Fig. A.6 shows how varying bias parameters affects performance. Increasing either weight's magnitude leads to worse command adherence, as the model is forced to attend less to the command conditions. Increasing w_{syl} improves FID and style accuracy at the cost of increased foot sliding. Positive values of w_{pst} reduce motion quality significantly, as they induce frozen poses, leading to foot sliding values of 0. Our defaults of $w_{\text{syl}} = 2.0$ and $w_{\text{pst}} = -1.0$ present a good trade-off between improved stylization quality and increased generation artifacts. We found these default values to work well across styles, embodiments and different models.

Unseen Styles To showcase generalization capabilities of our method, we select a random subset of the styles in 100STYLE as a test set (10 styles), which we do not include during training. Our model can still generate motion that matches these styles during inference, as shown in Fig. A.7.

Velocity Correction Fig. A.8 shows qualitative results of our assessment of the velocity correction mechanism. All trajectories correspond to roughly 30 s of motion. The periodic trajectories (circle, figure eights, square, rounded square) are repeated multiple times. In all cases, the character stays significantly closer to the commanded trajectory when the velocity correction scheme is applied.

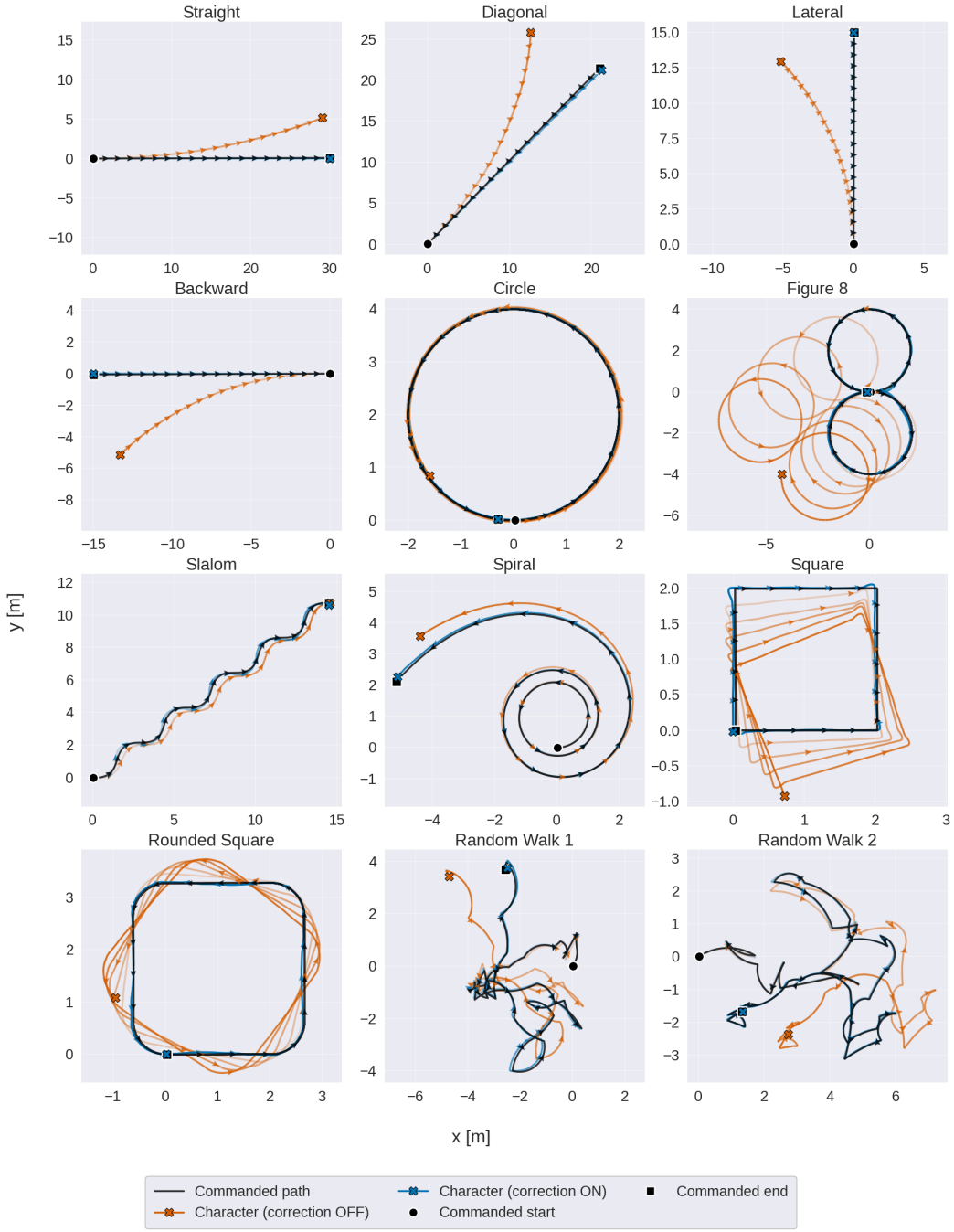

Figure A.8: **Velocity Correction.** Qualitative analysis of applying velocity correction over a number of different trajectories.