UNSUPERVISED LATENT ACTIONS FOR TERRAIN-AWARE INTERACTIVE MOTION GENERATION

Anonymous authors
Paper under double-blind review

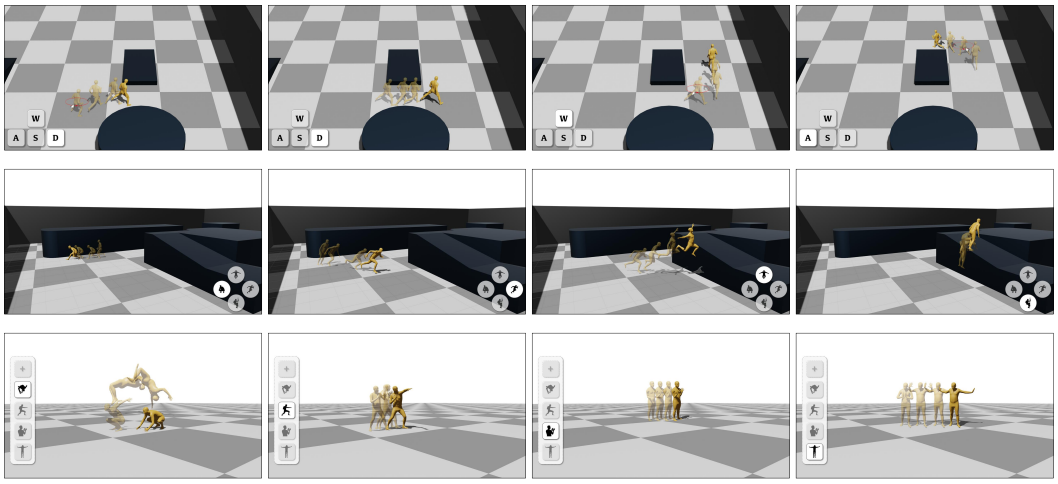

Figure 1: We propose TINMO, a framework for terrain-aware interactive motion generation using unsupervised latent action codes. TINMO supports real-time keyboard–mouse control for basic directional locomotion (1st row) and predefined skill control (2nd row). By further incorporating the proposed Action Library, the system enables plug-and-play expansion of motion skills, allowing newly added behaviors to be invoked interactively as additional semantic controls (3rd row).

ABSTRACT

Generating responsive and terrain-aware character motion in real time is challenging due to the mismatch between high-frequency user inputs and the structured dynamics of human movement, as well as the need for stable closed-loop control. We present **TINMO**, a framework for **T**errain-aware **IN**teractive **MO**tion generation based on unsupervised *Latent Action Codes*. These discrete codes are learned directly from unlabeled motion data and represent high-level motion intentions, enabling real-time control without manual action annotations. TINMO models motion as transitions between spatiotemporal primitives indexed by Latent Action Codes, allowing coherent and efficient autoregressive generation. A lightweight multimodal controller maps user inputs, terrain features, and motion history to the latent codes and root trajectories, producing context-aware behavior in closed-loop settings. To support extensibility, TINMO includes an *Action Library* that stores new motion skills as action-code sequences and local trajectories for plug-and-play invocation through the frozen generator without retraining. Experiments on terrain-aware path-following tasks show that TINMO outperforms the evaluated baselines in motion tracking while supporting interactive responsiveness, demonstrating the effectiveness of unsupervised action modeling for real-time character control. Video demonstrations are available in the supplementary materials.

1 INTRODUCTION

Real-time interactive character control remains a fundamental challenge in animation, simulation, and modern games. Users continuously steer virtual characters using heterogeneous keyboard and mouse signals while expecting immediate, coherent feedback that responds to the local environment. Unlike offline motion generation methods (Tevet et al., 2022; Zhang et al., 2024; Dai et al., 2024; Chen et al., 2023), interactive control operates under strict latency and stability constraints: sparse

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

user events must be interpreted in context, small prediction errors can quickly compound, and the system must adapt on the fly within a closed control loop.

Recent autoregressive systems already support online control: A-MDM adapts an unconditional next-frame diffusion model through sampling, inpainting, or task-specific RL (Shi et al., 2024), CAMDM maps a user-side synthetic trajectory and style label to real-time motion (Chen et al., 2024), and DARTControl optimizes a continuous latent for spatial constraints (Zhao et al., 2024). PARC (Xu et al., 2025a) combines terrain-conditioned generation with physics-based tracking and iterative data augmentation, but is not a keyboard/mouse closed-loop controller. These methods establish interactivity or terrain traversal, but do not learn a discrete transition-action interface jointly inferred from keyboard, mouse, motion history, and local terrain. Classical motion matching (Büttner & Clavet, 2015) remains a strong retrieval baseline, while physics-based RL controllers (Peng et al., 2018; Wang et al., 2020; Juravsky et al., 2024; Tessler et al., 2024) target torque-level execution. Our scope is complementary: low-latency kinematic generation from causal user controls and terrain observations.

This gap arises from a mismatch in both rate and semantics between user inputs and character motion. Input is logged at 30 Hz, whereas our controller emits an eight-frame primitive every 0.4 s (2.5 Hz). A short control window can correspond to turning, braking, or accelerating depending on motion state and terrain. Temporally extended primitives capture steps and transitions more directly than independent frames (Zhao et al., 2024; Zhang & Tang, 2022); the challenge is to map causal inputs onto coherent primitive transitions.

Motivated by these observations, we propose **TINMO**, a framework for **T**errain-aware **IN**teractive **MO**tion generation. We use "motion world model" in the narrow state–action–transition sense: a state is a spatiotemporal latent primitive, an action is an unsupervised discrete transition code, and a conditional denoiser predicts the next primitive. This specializes the latent-action formulation used for video and 3D scenes (Bruce et al., 2024; Gao et al., 2025; Zheng et al., 2024) to human kinematics.

To control this latent space, a lightweight multimodal controller maps keyboard/mouse windows, local terrain features, and motion history to a Latent Action Code and a future root trajectory. The recorded input stream is incorporated at primitive boundaries. Terrain affects generation through these controller outputs rather than as a direct denoiser input.

To enable skill expansion, we further introduce the *Action Library*, a repository of encoded motion behaviors. Each entry stores an action-code sequence together with its local root trajectory; a semantic key invokes both through the frozen generator after history matching and heading alignment. The same frozen transition model thus supports both controller-driven locomotion and semantic invocation of complete stored behaviors. As illustrated in Fig. 1, TINMO offers a low-latency framework for terrain-conditioned interactive character motion. In summary, our contributions are threefold:

- We cast terrain-aware interactive motion generation as a *motion world model* whose state space is a primitive-level latent and whose action space is a discrete code learned without supervision. This latent-dynamics formulation supports low-latency autoregressive rollouts in closed-loop control.
- We develop a lightweight multimodal controller that maps short windows of keyboard and mouse input, together with local terrain features, to discrete Latent Action Codes and root trajectories, bridging 30 Hz observations and 2.5 Hz primitive decisions.
- We propose the Action Library, which stores new behaviors as sequences of existing action codes and local trajectories, enabling semantic invocation of complete behaviors through the frozen generator without retraining.

2 Related Work

Generative World Models for Dynamic Scenes. Generative world models synthesize future states of dynamic systems from partial observations and control signals. Recent advances cover video generation (Bruce et al., 2024; Yuan et al., 2026; Mei et al., 2025; Zhou et al., 2026; Wan et al., 2025; Wu et al., 2024), occupancy prediction (Jin et al., 2026; Zheng et al., 2024; Zhang et al., 2025), and LiDAR synthesis (Liu et al., 2025; Zyrianov et al., 2025; Liang et al., 2025; Xu et al., 2025b). Genie (Bruce et al., 2024) and AdaWorld (Gao et al., 2025) learn latent actions from video. TINMO

transfers the state–action–transition abstraction to motion primitives; our "world model" claim is restricted to this learned kinematic transition model.

Trajectory- and Text-Conditioned Motion Generation. A large body of work synthesizes motion conditioned on paths (Wang et al., 2025; Wan et al., 2024; Bahmani et al., 2024; Huang et al., 2024; Liu et al., 2024), spatial constraints (Xie et al., 2023), or classifier guidance (Karunratanakul et al., 2023). Interactive autoregressive models include A-MDM (Shi et al., 2024), CAMDM (Chen et al., 2024), and DARTControl (Zhao et al., 2024); their released interfaces use task-specific sampling/RL, user-synthesized trajectory and style, or continuous latent optimization, respectively. PARC (Xu et al., 2025a) learns terrain traversal and corrects generated motion with a physics tracker. TINMO instead studies a kinematic controller that infers a discrete transition code and root trajectory from aggregated causal controls and terrain. These are complementary interfaces rather than claims that prior work lacks interactivity or terrain awareness.

Interactive Character Control. Two paradigms dominate prior interactive control. *Kinematic* approaches such as motion matching (Büttner & Clavet, 2015) and phase-functioned networks (Holden et al., 2017) retrieve or blend pre-recorded clips conditioned on control cues. They remain the de-facto industrial baseline due to low latency and high fidelity, but are inherently retrieval-driven, so behavior is limited by the database and scaling to new terrains or skill combinations requires expanding curated motion sets. We therefore include motion matching as a baseline and analyze this trade-off. *Physics-based RL* controllers, including DeepMimic (Peng et al., 2018), UniCon (Wang et al., 2020), MimicKit (Peng, 2025), SuperPADL (Juravsky et al., 2024), and MaskedMimic (Tessler et al., 2024), synthesize reactive behaviors via simulation and policy learning. These methods operate on a different abstraction, using articulated rigid bodies and torque-level control rather than kinematic SMPL-X trajectories. Incorporating new skills typically requires substantial retraining or reward redesign. We therefore treat physics-based control as a *complementary* paradigm and report a physics-based tracking evaluation of MaskedMimic on our benchmark for context (Sec. 4.2). Within the kinematic regime, TINMO operates in a structured latent action space, synthesizes motion via transitions between spatiotemporal primitives controlled by discrete Latent Action Codes, and supports plug-and-play skill expansion through the Action Library without retraining.

3 METHOD

Motion as a World Model. We instantiate TINMO as a *motion world model*, importing the action-aware (state, action, transition) formulation from video and 3D-scene world models (Bruce et al., 2024; Gao et al., 2025; Zheng et al., 2024) and specializing each component to human kinematics. Indexing transitions by their destination primitive t: (i) the *state* $z_t \in \mathcal{Z}$ is a latent embedding of a spatiotemporal motion primitive (Zhang & Tang, 2022) rather than a frame-level pose, capturing temporally extended structure; (ii) the *action* $\hat{a}_t \in \mathcal{C}$ is a discrete code from a finite learned codebook, inferred without action labels from latent transitions (Gao et al., 2025); (iii) the *transition* $\mathcal{P}$ evolves z_{t-1} into z_t conditioned on $\hat{a}_t$, history, and a predicted root trajectory. Environment observations affect the transition through the controller outputs; the denoiser does not observe terrain directly. This view distinguishes TINMO from autoregressive motion models with only continuous generation latents (e.g., DARTControl; Zhao et al., 2024) by exposing a learned discrete transition interface. We use the world-model terminology only for this motion-specific state–action–transition abstraction.

Problem Definition. Let τ_t be the last observed frame before primitive t. Its rolling history is $\mathcal{H}_t = \{h_{\tau_t - H+1}, \dots, h_{\tau_t}\}$, and the initial history seed contains H frames representing initial poses and root features (Zhao et al., 2024). Given a stream of keyboard/mouse signals $\mathcal{S} = \{s_1, s_2, \dots\}$ and local environment observations $\mathcal{O}$, the objective is to autoregressively predict motion through repeated transitions

$$z_t = \mathcal{P}(z_{t-1}, \hat{a}_t \mid \mathcal{H}_t, \mathcal{T}_t), \tag{1}$$

where $\mathcal{T}_t$ is a root trajectory predicted by the controller. The controller maps an aggregated signal window, terrain, and history to both outputs: $(\hat{a}_t, \mathcal{T}_t) = \pi(\mathcal{S}_{\tau_t - \Delta T:\tau_t}, \mathcal{O}_t, \mathcal{H}_t)$. The remainder instantiates the latent representations (Sec. 3.1), controller (Sec. 3.2), and Action Library (Sec. 3.3).

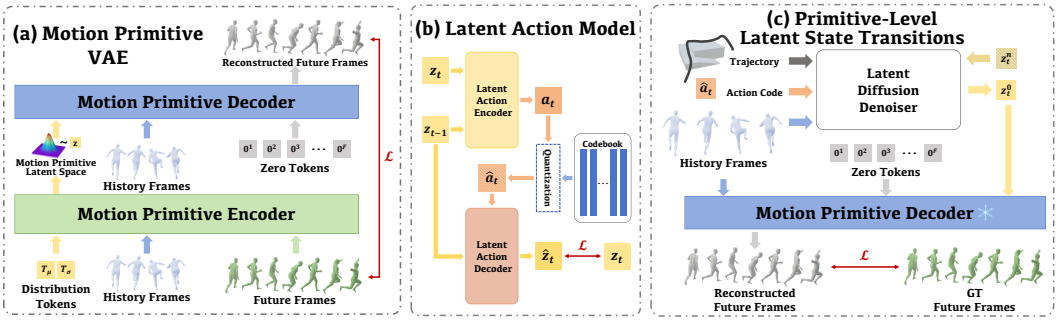

Figure 2: Overview of the latent dynamics architecture. (a) The Motion Primitive VAE encodes translation-free local primitives into $\mathcal{Z}$. (b) The Latent Action Model quantizes transition descriptors from (z_{t-1}, z_t) without action labels. (c) A latent diffusion denoiser predicts future states conditioned on the code, root trajectory, and motion history; the frozen MVAE decoder reconstructs motion.

3.1 Unsupervised Latent Dynamics

Motion Primitive Representations. To facilitate autoregressive modeling of human dynamics and ensure rotation invariance, we decompose continuous motion sequences into a stream of overlapping motion primitives (Zhang & Tang, 2022; Zhao et al., 2024). Each primitive $\mathcal{M}_t = [\mathcal{H}_t, \mathcal{X}_t]$ consists of H frames of historical poses and F frames of future motion. Unlike DARTControl (Zhao et al., 2024), we explicitly decouple translation from pose to simultaneously preserve kinematic realism and enable precise spatial control. Specifically, each motion sequence $\mathcal{X}_t$ is decomposed into a translation-free local pose sequence $\mathcal{P}_t$ and a global root trajectory $\mathcal{T}_t$. The final motion is obtained via a superimposition operator $\oplus$:

$$\mathcal{X}_t = \mathcal{P}_t \oplus \mathcal{T}_t, \tag{2}$$

where $\mathcal{P}_t$ is decoded from the latent state z_t in a canonical local coordinate system with global translation removed, mitigating displacement artifacts. The root trajectory $\mathcal{T}_t = \{\mathbf{p}_1, \ldots, \mathbf{p}_F\}$, where $\mathbf{p}_f \in \mathbb{R}^3$, is predicted by an independent trajectory module conditioned on motion history $\mathcal{H}_t$, control signals $\mathcal{S}$, and terrain observations $\mathcal{O}_t$. The operator $\oplus$ restores the canonical heading and adds the predicted 3D root translation to each local pose.

Building on this formulation, we introduce a motion primitive variational autoencoder (MVAE) that embeds each translation-free primitive into a compact latent space $\mathcal{Z}$ as shown in Fig. 2 (a). In contrast to DARTControl (Zhao et al., 2024), which retains relative root translations in its latent representation, our MVAE encodes only local kinematic structure. The encoder uses learned distribution-query tokens to parameterize the latent Gaussian; the decoder uses zero-valued future queries with positional encodings to reconstruct the future poses. Appendix B specifies these tokens and all component architectures. This separation of semantic motion dynamics from global spatial control provides the foundation for our latent world model and enables action-based interactive synthesis.

Unsupervised Latent Action Modeling. Interactive motion generation benefits from transition representations that abstract beyond low-level pose details. Inspired by latent-action video models (Gao et al., 2025), we use a Latent Action Model (LAM) to learn discrete transition descriptors from unlabeled motion. The codes provide a compact interface for predicting changes between latent primitives. As illustrated in Fig. 2 (b), the action encoder E_{act} maps consecutive latent states (z_{t-1}, z_t) to a continuous representation a_t. A Vector Quantization (VQ) layer maps it to the learned code set $\mathcal{C} = \{e_1, \ldots, e_{N_a}\}$:

$$\hat{a}_t = \text{Quant}(E_{act}(z_{t-1}, z_t)). \tag{3}$$

The 256-entry codebook is learned with exponential-moving-average (EMA) updates and a commitment loss, without action labels. LAM is trained with a predictive objective: given z_{t-1} and $\hat{a}_t$, the decoder predicts the next latent state $\hat{z}_t$, optimized via

$$\mathcal{L}_{LAM} = \|z_t - \hat{z}_t\|^2 + \mathcal{L}_{commit}. \tag{4}$$

After pretraining, these codes condition the state transition model. Unlike prior video-domain formulations (Bruce et al., 2024; Gao et al., 2025), we evaluate this interface for interactive 3D motion primitives and infer its codes from causal controls.

Primitive-Level Latent State Transitions. We parameterize the transition function $\mathcal{P}$ with a latent diffusion denoiser D_θ operating on the MVAE latent space as shown in Fig. 2 (c). Given history

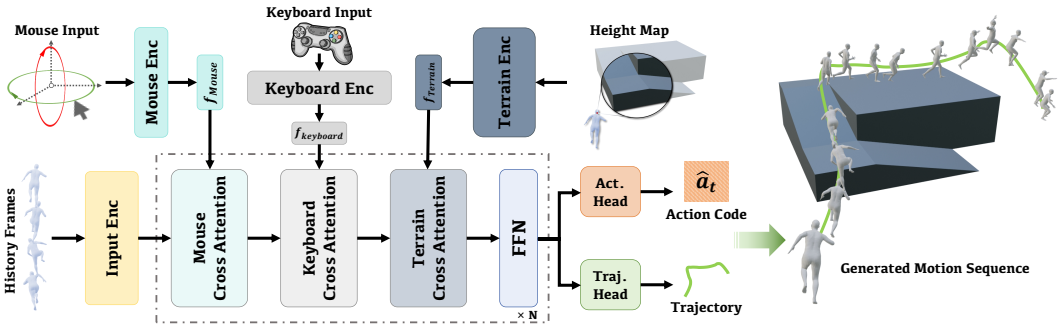

Figure 3: The multimodal controller aggregates a primitive window of keyboard and mouse input, encodes a local terrain heightmap, and fuses these features with motion history. It predicts (i) a discrete transition code $\hat{a}_t$ from the learned LAM codebook and (ii) a continuous root trajectory $\mathcal{T}_t$. Terrain influences the generator through these outputs rather than through direct denoiser conditioning.

frames $\mathcal{H}_t$ and action code $\hat{a}_t$ from the pretrained LAM, the denoiser learns to directly predict the clean latent state z_t^0 from a noisy sample z_t^n:

$$\hat{z}_t = D_\theta(z_t^n, n \mid \hat{a}_t, \mathcal{H}_t, \mathcal{T}_t), \tag{5}$$

where z_t^n is produced by the forward diffusion process and n denotes the diffusion timestep. In practice, we concatenate the action code embedding, the encoded history motion, and a root trajectory condition $\mathcal{T}_t$ as input. This design explicitly injects controllable intent and trajectory guidance into latent dynamics while preserving the compactness of the state space. We then reconstruct the future motion through the frozen MVAE decoder. MVAE and LAM parameters remain fixed while training the denoiser; reconstruction losses propagate through the fixed decoder to the denoiser. The full objective is a weighted sum of latent reconstruction, motion reconstruction, and motion-consistency regularizers:

$$\mathcal{L}_D = \lambda_z \|z_t - \hat{z}_t\|_2^2 + \lambda_x \|\mathcal{P}_t - \hat{\mathcal{P}}_t\|_2^2 + \lambda_\Delta \mathcal{L}_\Delta, \tag{6}$$

where $\hat{\mathcal{P}}_t = \mathrm{Dec}(\hat{z}_t, \mathcal{H}_t)$ reconstructs local pose, and $\mathcal{L}_\Delta$ penalizes temporal deltas of joints and orientation. Global motion is obtained after root-trajectory composition. We also apply conditional dropout on $\hat{a}_t$ for classifier-free guidance to improve robustness and generalization.

To mitigate exposure bias in long-horizon generation, we adopt a rollout training strategy: after a warm-up stage using ground-truth history, we gradually replace the history input with model-predicted motion and, when enabled, sample with a full denoising loop. The specific multi-phase History-GT mixed training schedule is detailed in Appendix B. This aligns training with inference-time autoregressive usage and stabilizes multi-primitive transitions.

3.2 Latent Intent Mapping

Interactive control faces a rate mismatch between asynchronous user signals and structured motion primitives. Keyboard and mouse events are sampled at 30 Hz, whereas the controller emits one eight-frame primitive every 0.4 s (2.5 Hz). Directly regressing latent states from raw signals often violates the learned motion prior and leads to physically implausible behavior. To address this issue, we design an interactive controller (Fig. 3) that maps multimodal user inputs and motion history $\mathcal{H}_t$ to both discrete action codes $\hat{a}_t \in \mathcal{C}$ and continuous future root trajectories $\mathcal{T}_t$. This dual-output formulation explicitly decouples semantic motion selection from spatial grounding, enabling user intent to modulate both the motion manifold and the character's trajectory in a principled and controllable manner.

User control has three modalities. Keyboard input $\mathbf{s}_t^k \in \{0,1\}^{K_{\text{key}}}$ and mouse displacement $\mathbf{s}_t^m \in \mathbb{R}^2$ are logged at 30 Hz, while a terrain heightmap $\mathbf{M}_t \in \mathbb{R}^{G\times G}$ describes local geometry. Signals are collected into a temporal window for each primitive. Learnable positional embeddings encode temporal order, and each recorded input sample is incorporated at the next primitive update. We rotate the local terrain patch into a head-aligned egocentric frame and express mouse displacement in the same root-centered coordinates. Linear encoders process the aggregated key and mouse features, while a shallow CNN embeds the heightmap. The final fusion of these signals into actions and paths is achieved through a sequence of cascade cross-attention blocks. History frames are used as

semantic query to selectively extract relevant intent from the modality features. This query interacts sequentially with the mouse, keyboard, and terrain features, and separate heads output a categorical distribution over $\mathcal{C}$ and a continuous future root trajectory $\mathcal{T}_t$. The two outputs separate transition selection from spatial grounding. During controller training, the pretrained LAM supplies the target action codes and the recorded motion supplies the target root trajectories; the motion generator remains frozen.

3.3 Action Library Construction

Construction. Motivated by our action-code interface and translation–pose decoupling, we introduce the Action Library $\mathcal{L}$ as a curated repository of reusable behaviors, each stored as a sequence over $\mathcal{C}$ with its local root trajectory. For each sequence $s = \{y_f\}_{f=1}^{N}$, we extract $x_f^g = \Pi_{\text{pelvis}}(y_f)$ and remove translation to obtain local poses. After encoding the resulting primitives as $z_0, \ldots, z_K$, we quantize their transitions with LAM:

$$\mathcal{A}(s) = \{\text{Quant}(E_{act}(z_{t-1}, z_t))\}_{t=1}^{K}, \tag{7}$$

and store the corresponding local trajectory

$$\mathcal{T}(s) = \{\bar{x}_f^g\}_{f=1}^{N}, \quad \bar{x}_f^g = x_f^g - x_1^g. \tag{8}$$

We keep the sequence if it passes motion filters on displacement, average speed, angular variation, path length, and primitive count, and add it to the library as $(\mathcal{A}(s), \mathcal{T}(s))$. The specific filtering thresholds for the Action Library are summarized in Table 5 in the Appendix.

Inference. At runtime, a semantic key press switches the controller into *library retrieval* mode, where the selected library item directly provides the action-code sequence and trajectory condition. To reduce drift, we match over a short history window and can re-select at each primitive boundary. Given a window of joint positions and velocities $\{p_{\tau_t-h,j}, \dot{p}_{\tau_t-h,j}\}_{h=0}^{H-1}$ and the first H frames of each library item $\{p_{i,j}^h, \dot{p}_{i,j}^h\}$, we select

$$i^* = \arg\min_i \sum_{h=0}^{H-1} \sum_{j=1}^{J} \left(\|p_{\tau_t-h,j} - p_{i,j}^h\|_2^2 + \lambda \|\dot{p}_{\tau_t-h,j} - \dot{p}_{i,j}^h\|_2^2 \right). \tag{9}$$

We align the local trajectory by rotating it with the previous root velocity direction $\phi = \text{atan2}(\Delta r_{\tau_t}^y, \Delta r_{\tau_t}^x)$:

$$\tilde{\mathcal{T}}(s_{i^*}) = \{R(\phi)\, \bar{x}_f^g + r_{\tau_t}\}_{f=1}^{N}. \tag{10}$$

Finally, the frozen generator is conditioned on $\mathcal{A}(s_{i^*})$ and $\tilde{\mathcal{T}}(s_{i^*})$.

4 Experiments

4.1 Experimental Setup

Task Definition. We evaluate closed-loop path following on held-out ALS sequences. TINMO replays only the recorded causal keyboard/mouse stream; at each primitive boundary its Controller receives this stream, the local heightmap centered on the generated character, and generated motion history, then predicts the action code and root trajectory used by the denoiser. The held-out root path $\mathcal{T}^* = \{\mathbf{p}_1^*, \ldots, \mathbf{p}_T^*\}$ is never an input to TINMO and is used only after rollout to compute trajectory error. Baselines without a keyboard/mouse interface receive $\mathcal{T}^*$ through their native trajectory-control mechanisms, so this is a best-effort common-path benchmark rather than an architecture-only comparison.

Dataset. Our framework is trained and evaluated using two primary datasets: (1) **BABEL** (Punnakkal et al., 2021): A large-scale dataset with fine-grained action labels combined with ALS motion for pre-training the MVAE, Latent Action Model (LAM), and denoiser to learn a robust motion prior. (2) **ALS Dataset**: A specialized interactive motion dataset featuring complex steering and navigation patterns. We use the ALS training split to fine-tune the Controller and Trajectory Predictor, and the test split for the benchmark evaluation. See Appendix C for more details on the ALS dataset collection and terrain representation.

Table 1: Path-following task results. Comparison of trajectory error (TE), foot skating (FS), FID, control latency (Lat.), penetration rate (Pen.), penetration depth (Depth), contact consistency (CC), and foot-height error (Foot) across baselines. **Bold** marks the best, underline the second-best per column, excluding GT. Our method achieves the lowest trajectory error, penetration rate, depth, and foot-height error, with second-best foot skating and latency, demonstrating strong accuracy, realism, and responsiveness for interactive character control.

Paradigm	Method	TE ↓	FS ↓	FID ↓	Lat. ↓	Pen. ↓	Depth ↓	CC ↑	Foot ↓
Retrieval	Motion Matching	0.66	**0.97**	11.74	**2.04**	2.28	135.1	19.3	99.0
Diffusion	OmniControl (official)	1.85	22.87	6.89	1138.71	4.54	119.1	25.8	162.4
	OmniControl (ALS-ft)	1.15	29.80	2.76	1168.73	1.51	94.6	19.2	83.5
	DNO	0.48	1.71	251.53	2030.17	13.03	282.6	7.5	369.5
	MotionLCM (official)	2.94	39.52	14.24	124.54	10.36	241.9	23.5	277.0
	MotionLCM (ALS-ft)	1.44	29.48	**2.54**	127.83	4.21	283.3	13.6	159.2
	DARTControl	4.55	1.72	45.30	2304.36	12.65	243.5	**56.4**	297.8
	A-MDM	2.24	30.55	8.93	64.07	6.59	203.7	10.1	202.1
Physics-RL	MaskedMimic	2.98	6.87	16.47	84.63	7.32	123.7	43.3	156.9
World Model	**TINMO (Ours)**	**0.19**	1.11	4.00	11.82	**1.34**	**86.9**	15.0	**53.9**
Reference	GT	—	—	—	—	1.42	64.5	29.8	51.8

Metrics. We evaluate the benchmark axes below, with metric directions marked explicitly. *Tracking accuracy: Traj. Error* (TE) is the mean L2 distance, in meters, between the generated root trajectory and the reference (the target path in the path-following benchmark; the source-clip pelvis in reconstruction settings). *Joint accuracy* (when reported): *MPJPE* is the mean per-joint position error in meters over all 22 SMPL-X body joints, averaged across frames. *Physical realism: Foot Skating* (FS, %) measures horizontal foot motion near the support surface. We additionally report heightfield body-penetration rate (%), penetration depth (mm), foot-contact consistency (%), and foot-height error (mm); these measure different failure modes and are not substitutes for FS. *Motion fidelity: FID* is the Fréchet distance against the reference motion distribution. *Efficiency: Latency* (Lat.) is reported wall-clock computation time per frame in milliseconds, distinct from input-to-motion response time. App. E.4 specifies the frame denominators and measured operations.

Baselines. We compare against representative methods from distinct families using their released spatial-control interfaces. *Generative diffusion models*: OmniControl (Xie et al., 2023), treating path following as dense spatial constraints on the root joint; DNO (Karunratanakul et al., 2024), an inference-time optimization method using a pretrained diffusion prior without additional training; MotionLCM (Dai et al., 2024), a real-time model based on Latent Consistency Models with trajectory injection via an adapter; and DARTControl (Zhao et al., 2024), an autoregressive diffusion framework combining a motion VAE with DDIM-based latent optimization for trajectory adherence. *Retrieval-based interactive control*: we include classical *Motion Matching* (Büttner & Clavet, 2015) as the de-facto industrial baseline, using Holden's open-source raylib implementation (see Appendix D for details). We additionally run A-MDM (Shi et al., 2024) with its released PPO `PathEnv` policy and report MaskedMimic (Tessler et al., 2024) as a cross-paradigm physics reference. Appendix E.4 documents interface and representation gaps.

4.2 Quantitative Results

Tab. 1 evaluates closed-loop path tracking on 126 non-overlapping windows from four held-out source sequences. We report released checkpoints and ALS-adapted variants, with evaluation details in App. E.4.

Tracking Precision and Efficiency. TINMO attains *Traj. Error* 0.19 m, approximately 60% lower than DNO (0.48 m) and 71% lower than Motion Matching (0.66 m). TINMO predicts its path from recorded controls; the baselines observe target-path geometry through different native interfaces. A-MDM's released PPO `PathEnv` policy observes four future path points and obtains 2.24 m (three-seed average). TINMO takes 11.82 ms per generated frame; Motion Matching has the lowest reported latency, 2.04 ms.

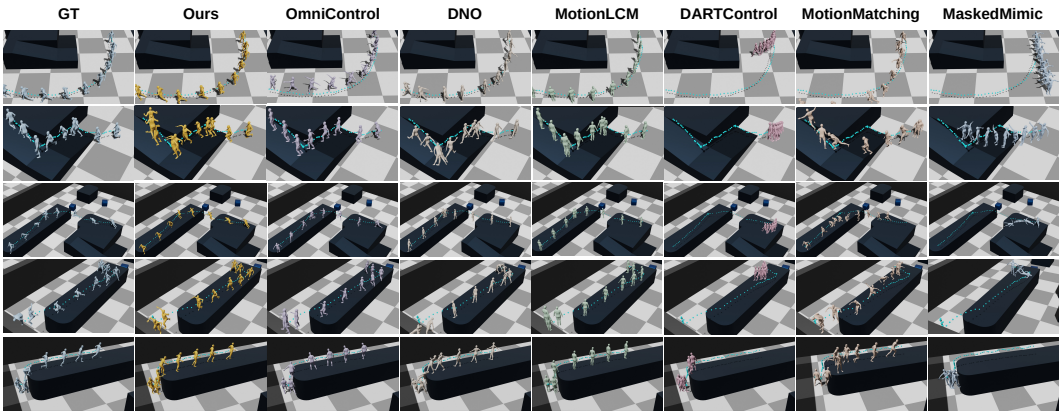

Figure 4: Qualitative comparison of different methods on the path-following task. Rows represent five test scenarios with different path curvatures and terrain elevations. The eight columns show GT, Ours (TINMO), OmniControl, DNO, MotionLCM, DARTControl, Motion Matching, and MaskedMimic.

Responsiveness. Across 1,024 input events in the 126-window split, direction and gait responses are within 300 and 198 ms, respectively. Inputs are sampled at 30 Hz and incorporated into eight-frame predictions (0.4 s at 20 FPS). These event-response measurements and the 2.5 Hz primitive update rate are distinct from compute latency.

Terrain Interaction. The full-split measurements report TINMO penetration rate 1.34%, penetration depth 86.9 mm, and foot-height error 53.9 mm, the lowest among evaluated methods excluding GT (Tab. 1). GT penetration is 1.42%, but its penetration depth and foot-height error are lower (64.5 and 51.8 mm). TINMO's contact consistency (15.0%) is below DARTControl's (56.4%) and MaskedMimic's (43.3%), whose support speeds (0.60/1.22 vs. 2.40 m/s for GT) affect speed-threshold contact scores. These aggregate measurements do not establish collision-free or physically valid interaction. App. B.5.1 specifies the heightfield and contact thresholds.

Motion Fidelity. TINMO attains FID 4.00 and FS 1.11% in the path-following evaluation (App. E.4). The full-model ablation reproduces these values; removing motion primitives increases FID to 25.47, linking motion quality to the temporal structure of the learned states.

Out-of-Distribution Generalization of the Action Library. We evaluate whether the Action Library extension mechanism (Sec. 3.3) degrades when populated with skills outside the ALS training distribution. We curate entries from three categories absent from ALS, *acrobatics*, *upper-body gestures*, and *sports*, using BABEL *val* clips (held out from LAM training), and further test strict OOD generalization with eight clips from the AMASS DanceDB subset, which is never paired with BABEL labels and thus unseen by LAM. Each clip is encoded into action codes via LAM and re-decoded with the frozen denoiser, and compared to the source motion. Table 2 reports per-clip mean *Traj. Error*, full-body *MPJPE*, and pooled-segment *FID*. All four groups have segment-feature FID below 1.0. The eight DanceDB clips attain MPJPE 0.053 m and FID 0.467. These reconstructions demonstrate reuse of the learned action interface for complete behaviors outside ALS, using source-derived codes and local trajectories with a frozen generator. This handcrafted-feature FID is not compared numerically with path-following FID.

4.3 Qualitative Results

Visual comparison. Fig. 4 shows five terrain and path scenarios in eight columns: the reference motion, TINMO, OmniControl, DNO, MotionLCM, DARTControl, Motion Matching, and Masked-Mimic. The rows illustrate curved paths, elevation changes, and vertical motion. The first four supplementary-video segments (0:05–0:27) show synchronized keyboard, mouse, predicted-path, and terrain overlays, connecting user commands to generated motion. Jump and climb examples (0:47.7–0:55.2) also show the remaining motion and contact artifacts. The video encodes 20-FPS motion at 30 FPS; playback at approximately $0.67\times$ restores native motion speed.

User study. We conducted a user study with 15 participants to evaluate perceived motion naturalness and tracking accuracy. Each participant rated 20 randomly selected clips from our method and the baselines on a 5-point Likert scale (1: Poor, 5: Excellent) based on (1) **Motion Naturalness** (absence

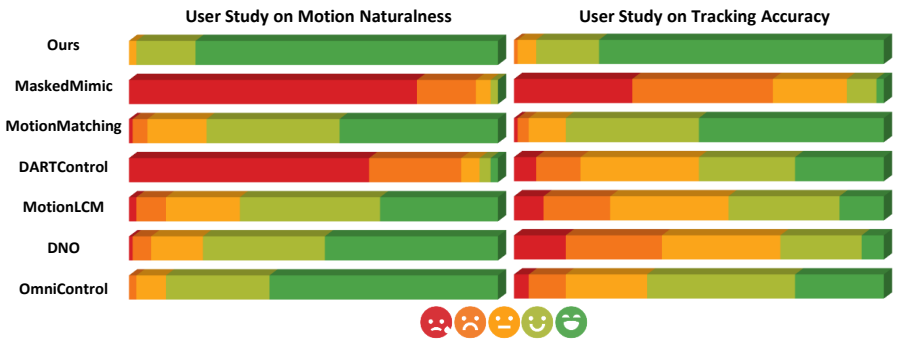

Figure 5: User study. Participants rated motion naturalness and tracking accuracy; our method received the highest mean ratings on both.

Table 2: Action Library reconstruction on held-out skills. TE/MPJPE are in m; FID pools hand-crafted segment features. N counts clips.

Dataset	Category	N	TE	MPJPE	FID
BABEL	acrobatics	19	0.338	0.382	0.934
	upper-body	50	0.148	0.181	0.313
	sports	50	0.199	0.240	0.467
DanceDB	dance	8	0.040	0.053	0.467

Table 3: Ablation studies. Top: contribution of core components. Bottom: terrain window size W sweep.

Variant	MPJPE	Traj. Err.	FS	FID	Lat.
Full model	**0.240**	**0.190**	**1.11**	**4.00**	11.82
w/o decoupling	0.850	0.870	2.84	12.81	11.54
w/o mo. primitives	1.210	0.560	5.37	25.47	**10.23**
w/o terrain input	0.520	0.710	2.26	8.29	11.65
Terrain W=1.0 m	0.252	0.200	2.42	4.64	10.72
Terrain W=3.0 m	0.249	0.197	2.10	4.75	11.00
Terrain W=4.0 m	0.249	0.196	2.10	4.83	10.68

of artifacts such as foot skating or jitter) and (2) **Tracking Accuracy** (how well the character follows the target path). As shown in Fig. 5, our method receives the highest mean ratings on both axes. OmniControl ranks second overall on naturalness, while Motion Matching ranks second overall on tracking accuracy. These ratings characterize the perceived quality of the evaluated clips; the study protocol is detailed in App. E.6.

4.4 Ablation Studies

We conduct ablation studies to analyze the contribution of each core component. Table 3 evaluates component removals and terrain-window sizes under the same TINMO evaluation setting. **1) Importance of Decoupling.** Removing translation–pose decoupling (*w/o decoupling*) increases *Traj. Error* from 0.19 m to 0.87 m, showing that separating root translation from local pose simplifies the Controller's task and improves spatial navigation. **2) Impact of Motion Primitives.** Without motion primitives (*w/o motion primitives*), *FID* rises from 4.00 to 25.47 and *MPJPE* to 1.21 m, indicating that frame-level control leads to joint jitter and unstable transitions, while primitive-level modeling preserves motion coherence. **3) Contribution of Terrain Awareness.** Removing the heightmap (*w/o terrain input*) degrades tracking and joint accuracy (*MPJPE* 0.52 m), showing that local elevation context helps the Controller adapt its root path and pose. Varying the heightmap window $W \in \{1, 2, 3, 4\}$ m (Tab. 3) shows W=2 m is optimal for the reported tracking metrics: smaller windows lack look-ahead, while larger ones dilute spatial resolution and increase foot skating.

5 Conclusion

We presented TINMO, a framework for terrain-aware interactive motion generation that models motion as transitions between spatiotemporal primitives controlled by a lightweight multimodal controller, with plug-and-play skill expansion via an Action Library. Experiments show superior tracking accuracy over the evaluated baselines while supporting interactive responsiveness.

Limitations and Broader Impact. TINMO relies on preprocessed terrain features and a manually defined input set; future work will study learned terrain representations and richer input modalities. As a kinematic model, it does not enforce physical contact, motivating contact-aware refinement. The framework can lower authoring costs for interactive characters in games, simulation, and accessibility-oriented media; it operates on generic skeletons without modeling any specific individual, limiting human-likeness misuse risk.

Reproducibility statement

The model architecture and objectives are specified in Secs. 3.1–3.3; training details, dataset construction, evaluation protocols, and per-component configurations are provided in the appendix. The supplementary video illustrates keyboard/mouse control with synchronized overlays. We will release the evaluation and baseline-adaptation code, per-window outputs, and trained models to make coordinate conversions and metric aggregation auditable.

AI use statement

Generative AI tools assisted manuscript editing and LaTeX preparation. They were not used to generate experimental data.

References

Sherwin Bahmani, Xian Liu, Wang Yifan, Ivan Skorokhodov, Victor Rong, Ziwei Liu, Xihui Liu, Jeong Joon Park, Sergey Tulyakov, Gordon Wetzstein, et al. Tc4d: Trajectory-conditioned text-to-4d generation. In *European Conference on Computer Vision*, pp. 53–72. Springer, 2024.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Michael Büttner and Simon Clavet. Motion matching and the road to next-gen animation. *Game Developers Conference*, 2015.

Rui Chen, Mingyi Shi, Shaoli Huang, Ping Tan, Taku Komura, and Xuelin Chen. Taming diffusion probabilistic models for character control. In *ACM SIGGRAPH 2024 Conference Papers*, pp. 1–10, 2024.

Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, and Gang Yu. Executing your commands via motion diffusion in latent space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18000–18010, 2023.

Wenxun Dai, Ling-Hao Chen, Jingbo Wang, Jinpeng Liu, Bo Dai, and Yansong Tang. Motionlcm: Real-time controllable motion generation via latent consistency model. In *European Conference on Computer Vision*, pp. 390–408. Springer, 2024.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions. *arXiv preprint arXiv:2503.18938*, 2025.

Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

Yiheng Huang, Hui Yang, Chuanchen Luo, Yuxi Wang, Shibiao Xu, Zhaoxiang Zhang, Man Zhang, and Junran Peng. Stablemofusion: Towards robust and efficient diffusion-based motion generation framework. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 224–232, 2024.

Bu Jin, Songen Gu, Xiaotao Hu, Yupeng Zheng, Xiaoyang Guo, Qian Zhang, Xiaoxiao Long, and Wei Yin. Occtens: 3d occupancy world model via temporal next-scale prediction. *IEEE Robotics and Automation Letters*, 2026.

Jordan Juravsky, Yunrong Guo, Sanja Fidler, and Xue Bin Peng. Superpadl: Scaling language-directed physics-based control with progressive supervised distillation. In *ACM SIGGRAPH 2024 Conference Papers*, pp. 1–11, 2024.

Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. Guided motion diffusion for controllable human motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2151–2162, 2023.

Korrawe Karunratanakul, Konpat Preechakul, Emre Aksan, Thabo Beeler, Supasorn Suwajanakorn, and Siyu Tang. Optimizing diffusion noise can serve as universal motion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1334–1345, 2024.

Ao Liang, Youquan Liu, Yu Yang, Dongyue Lu, Linfeng Li, Lingdong Kong, Huaici Zhao, and Wei Tsang Ooi. Lidarcrafter: Dynamic 4d world modeling from lidar sequences. *arXiv preprint arXiv:2508.03692*, 2025.

Pei Liu, Songtao Wang, Lang Zhang, Xingyue Peng, Yuandong Lyu, Jiaxin Deng, Songxin Lu, Weiliang Ma, Xueyang Zhang, Yifei Zhan, et al. Listar: Ray-centric world models for 4d lidar sequences in autonomous driving. *arXiv preprint arXiv:2511.16049*, 2025.

Qingze Tony Liu, Danrui Li, Samuel S Sohn, Sejong Yoon, Mubbasir Kapadia, and Vladimir Pavlovic. Trajdiffuse: A conditional diffusion model for environment-aware trajectory prediction. In *International Conference on Pattern Recognition*, pp. 382–397. Springer, 2024.

Zhiting Mei, Tenny Yin, Micah Baker, Ola Shorinwa, and Anirudha Majumdar. World models that know when they don't know: Controllable video generation with calibrated uncertainty. *arXiv preprint arXiv:2512.05927*, 2025.

Xue Bin Peng. Mimickit: A reinforcement learning framework for motion imitation and control. *arXiv preprint arXiv:2510.13794*, 2025.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

Abhinanda R Punnakkal, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra Quiros-Ramirez, and Michael J Black. Babel: Bodies, action and behavior with english labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 722–731, 2021.

Yi Shi, Jingbo Wang, Xuekun Jiang, Bingkun Lin, Bo Dai, and Xue Bin Peng. Interactive character control with auto-regressive motion diffusion models. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024.

Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. Maskedmimic: Unified physics-based character control through masked motion inpainting. *ACM Transactions on Graphics (TOG)*, 2024.

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

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Weilin Wan, Zhiyang Dou, Taku Komura, Wenping Wang, Dinesh Jayaraman, and Lingjie Liu. Tlcontrol: Trajectory and language control for human motion synthesis. In *European Conference on Computer Vision*, pp. 37–54. Springer, 2024.

Runqi Wang, Caoyuan Ma, Jian Zhao, Hanrui Xu, Dongfang Sun, Haoyang Chen, Lin Xiong, Zheng Wang, and Xuelong Li. Leader and follower: Interactive motion generation under trajectory constraints. *arXiv preprint arXiv:2502.11563*, 2025.

Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. Unicon: Universal neural controller for physics-based character motion. *arXiv preprint arXiv:2011.15119*, 2020.

Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideogpt: Interactive videogpts are scalable world models. *Advances in Neural Information Processing Systems*, 37: 68082–68119, 2024.

Yiming Xie, Varun Jampani, Lei Zhong, Deqing Sun, and Huaizu Jiang. Omnicontrol: Control any joint at any time for human motion generation. *arXiv preprint arXiv:2310.08580*, 2023.

Michael Xu, Yi Shi, KangKang Yin, and Xue Bin Peng. PARC: Physics-based augmentation with reinforcement learning for character controllers. In *ACM SIGGRAPH Conference Papers*, 2025a. doi: 10.1145/3721238.3730616.

Xiang Xu, Ao Liang, Youquan Liu, Linfeng Li, Lingdong Kong, Ziwei Liu, and Qingshan Liu. U4d: Uncertainty-aware 4d world modeling from lidar sequences. *arXiv preprint arXiv:2512.02982*, 2025b.

Jianhao Yuan, Xiaofeng Zhang, Felix Friedrich, Nicolas Beltran-Velez, Melissa Hall, Reyhane Askari-Hemmat, Xiaochuang Han, Nicolas Ballas, Michal Drozdzal, and Adriana Romero-Soriano. Inference-time physics alignment of video generative models with latent world models. *arXiv preprint arXiv:2601.10553*, 2026.

Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

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

Zhang Zhang, Qiang Zhang, Wei Cui, Shuai Shi, Yijie Guo, Gang Han, Wen Zhao, Jingkai Sun, Jiahang Cao, Jiaxu Wang, et al. Occupancy world model for robots. *arXiv preprint arXiv:2505.05512*, 2025.

Kaifeng Zhao, Gen Li, and Siyu Tang. Dartcontrol: A diffusion-based autoregressive motion model for real-time text-driven motion control. *arXiv preprint arXiv:2410.05260*, 2024.

Wenzhao Zheng, Weiliang Chen, Yuanhui Huang, Borui Zhang, Yueqi Duan, and Jiwen Lu. Occworld: Learning a 3d occupancy world model for autonomous driving. In *European conference on computer vision*, pp. 55–72. Springer, 2024.

Yang Zhou, Hao Shao, Letian Wang, Zhuofan Zong, Hongsheng Li, and Steven L Waslander. Drivinggen: A comprehensive benchmark for generative video world models in autonomous driving. *arXiv preprint arXiv:2601.01528*, 2026.

Vlas Zyrianov, Henry Che, Zhijian Liu, and Shenlong Wang. Lidardm: Generative lidar simulation in a generated world. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6055–6062. IEEE, 2025.

A Related Work

Generative World Models for Dynamic Scenes. Generative world models synthesize future states of dynamic systems from partial observations and control signals. Recent advances cover video generation (Bruce et al., 2024; Yuan et al., 2026; Mei et al., 2025; Zhou et al., 2026; Wan et al., 2025; Wu et al., 2024), occupancy prediction (Jin et al., 2026; Zheng et al., 2024; Zhang et al., 2025), and LiDAR synthesis (Liu et al., 2025; Zyrianov et al., 2025; Liang et al., 2025; Xu et al., 2025b). Genie (Bruce et al., 2024) and AdaWorld (Gao et al., 2025) learn latent actions from video. TINMO transfers the state–action–transition abstraction to motion primitives; our "world model" claim is restricted to this learned kinematic transition model.

Trajectory- and Text-Conditioned Motion Generation. A large body of work synthesizes motion conditioned on paths (Wang et al., 2025; Wan et al., 2024; Bahmani et al., 2024; Huang et al., 2024; Liu et al., 2024), spatial constraints (Xie et al., 2023), or classifier guidance (Karunratanakul et al., 2023). Interactive autoregressive models include A-MDM (Shi et al., 2024), CAMDM (Chen et al., 2024), and DARTControl (Zhao et al., 2024); their released interfaces use task-specific sampling/RL, user-synthesized trajectory and style, or continuous latent optimization, respectively. PARC (Xu et al., 2025a) learns terrain traversal and corrects generated motion with a physics tracker. TINMO instead studies a kinematic controller that infers a discrete transition code and root trajectory from aggregated causal controls and terrain. These are complementary interfaces rather than claims that prior work lacks interactivity or terrain awareness.

Interactive Character Control. Two paradigms dominate prior interactive control. *Kinematic* approaches such as motion matching (Büttner & Clavet, 2015) and phase-functioned networks (Holden et al., 2017) retrieve or blend pre-recorded clips conditioned on control cues. They remain the de-facto industrial baseline due to low latency and high fidelity, but are inherently retrieval-driven, so behavior is limited by the database and scaling to new terrains or skill combinations requires expanding curated motion sets. We therefore include motion matching as a baseline and analyze this trade-off. *Physics-based RL* controllers, including DeepMimic (Peng et al., 2018), UniCon (Wang et al., 2020), MimicKit (Peng, 2025), SuperPADL (Juravsky et al., 2024), and MaskedMimic (Tessler et al., 2024), synthesize reactive behaviors via simulation and policy learning. These methods operate on a different abstraction, using articulated rigid bodies and torque-level control rather than kinematic SMPL-X trajectories. Incorporating new skills typically requires substantial retraining or reward redesign. We therefore treat physics-based control as a *complementary* paradigm and report a physics-based tracking evaluation of MaskedMimic on our benchmark for context (Sec. 4.2). Within the kinematic regime, TINMO operates in a structured latent action space, synthesizes motion via transitions between spatiotemporal primitives controlled by discrete Latent Action Codes, and supports plug-and-play skill expansion through the Action Library without retraining.

B Implementation Details

B.1 Component Parameter Counts

We summarize the parameter counts of all TINMO components in Table 4. The full system contains roughly 50.2M parameters and runs at **11.82 ms/generated frame** on a single NVIDIA L40S GPU. The controller outputs eight frames per primitive; compute latency and event response time measure distinct properties.

Table 4: Component-wise parameter counts of TINMO.

Component	Architecture	# Params
MVAE	Transformer VAE (7 layers, d=256)	∼12.4M
LAM	MLP encoder/decoder + codebook of size 256	∼2.2M
Denoiser	Transformer (8 layers, d=512)	∼18.3M
Controller	Transformer (4 blocks, d=512)	∼17.3M
Total		**∼50.2M**

B.2 Motion Primitive Configuration

Unless otherwise specified, we represent motion as overlapping primitives with history length $H{=}2$ frames and future length $F{=}8$ frames (primitive length $H{+}F{=}10$ frames). Consecutive primitives are extracted with stride F (i.e., overlap of H frames). During training, we sample 4 consecutive primitives per sequence chunk.

B.3 Network Architectures

The TINMO framework consists of several key neural components.

MVAE. A Transformer-based VAE with 7 layers, 4 attention heads, and a latent dimension of 256. We apply a dropout of 0.1. The encoder prepends $2L$ learned distribution-query tokens to the embedded $H{+}F$ motion frames, with $L{=}1$. The first L token outputs parameterize μ and the next L parameterize $\log \sigma^2$, clamped to $[-10, 10]$. We sample $z = \mu + \sigma \odot \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$. The decoder attends to the latent, history, and F all-zero future query slots with positional encodings; the outputs at those slots are projected to the F future local-pose features. Distribution tokens are learned parameters; zero tokens are fixed query inputs.

LAM. The Latent Action Model uses an encoder-decoder architecture with 3 layers each and a hidden dimension of 512. It utilizes a codebook of size 256 and produces an action latent of dimension 512.

Denoiser. A Transformer with 8 layers and 4 attention heads, featuring a hidden dimension of 512 and an MLP feed-forward dimension of 1024. At each of the 10 diffusion steps (cosine schedule), the model is conditioned on: (i) an **Action Latent** (512D) from the LAM; (ii) **History Motion** ($H{=}2$ frames, normalized); and (iii) a **Root Trajectory** $\mathcal{T}_t \in \mathbb{R}^{F\times 3}$, containing one 3D translation per future frame. We apply classifier-free guidance with a dropout probability of 0.1 and a guidance scale $s{=}5.0$ at inference.

Controller. A Transformer-based controller with 4 blocks and 8 heads, using a hidden dimension of 512, FFN dimension of 2048, and dropout 0.1.

B.4 Training and Optimization

Unless otherwise specified, we use the Adam optimizer with a learning rate of $1\mathrm{e}{-4}$, EMA decay of 0.999, and mixed precision training across all components.

B.4.1 Component-Specific Settings

The **Denoiser** is trained with a batch size of 1024 per GPU. Its objective is Eq. 6, with λ_z, λ_x, and λ_Δ weighting latent reconstruction, local-pose reconstruction, and temporal consistency, respectively. The pretrained MVAE and LAM are frozen during denoiser training. During controller training, the pretrained motion generator is frozen; the controller learns the action-code and root-trajectory outputs. The **LAM** is trained with a commitment cost of 0.25, while the **Controller** utilizes a batch size of 128.

B.4.2 History-GT Mixed Training Strategy

To improve robustness under autoregressive sampling, we adopt a curriculum that gradually shifts the training distribution from ground-truth histories to model-generated ones. Concretely, we train for 300k steps in three phases: during the first 100k steps, the history is always ground-truth (GT); during the next 100k steps, we linearly anneal the rollout probability P_{rollout} from 0 to 1:

$$P_{\text{rollout}} = \min\left(1, \frac{u - 100\text{k}}{100\text{k}}\right),$$

where u is the optimization step. With probability P_{rollout} the model consumes its own prediction as the next-step history; during the last 100k steps, training is predominantly performed in rollout mode. In practice, the rollout history is produced by sampling the previous primitive and reusing the predicted future as the next history.

B.5 Additional Quantitative Results

We evaluate body penetration and foot contact to complement path tracking and motion-distribution measurements.

B.5.1 Terrain Interaction Evaluation Protocol

The terrain-interaction columns of Table 1 evaluate the complete held-out split: 126 windows totaling 8.6 minutes. Methods with native SMPL-X output (TINMO, Motion Matching, DARTControl, MaskedMimic, and GT) are converted to world space with error below 10^{-5} m; joint-only outputs (DNO, OmniControl, MotionLCM, and A-MDM) are fitted to SMPL-X by inverse kinematics, with about 13 mm mean pelvis error. We bilinearly sample the 512^2 ALS heightfield for 9,383 exterior body vertices and use a 2 cm penetration tolerance. Penetration rate is the fraction of penetrating vertex–frame samples; penetration depth (Depth) averages only the penetrating samples.

Each foot uses the Foot$\cup$ToeBase region (251 vertices). A foot's clearance in each frame is the 5th percentile of the terrain-relative heights of its region vertices, and its support speed is the velocity of the mean of its SMPL-X ankle and toe joints. Contact candidates have clearance ≤ 10 cm and absolute vertical speed ≤ 0.5 m/s; consistency additionally requires absolute clearance ≤ 10 cm and horizontal speed ≤ 0.5 m/s. Foot-height error (Foot) is the mean absolute clearance over all detected contact candidates, and the reported support speed is their mean horizontal support speed; both are pooled over the candidates of all 126 windows, as is contact consistency. The same thresholds apply to all methods in the table. Nonzero GT penetration reflects the 2.5D heightfield and SMPL-X retargeting, so GT serves as a sanity reference. A-MDM and MaskedMimic do not observe terrain: A-MDM follows a planar path and MaskedMimic runs flat-ground physics (App. E.3). Their outputs are placed in ALS world coordinates at the GT start pose, and A-MDM is averaged over three seeds. Because both methods leave the target path (TE 2.24 and 2.98 m), many of their samples lie on terrain they were never asked to traverse, so their terrain columns in Table 1 are a reference rather than evidence of terrain-aware interaction. IK-fitted rows sit about 2.9 cm higher than native SMPL-X rows, which under-reports their penetration.

B.6 Experimental Environment

All experiments were conducted on a Linux server (Ubuntu 22.04, kernel 5.15) equipped with one NVIDIA L40S GPU (48GB). We utilized CUDA 12.1 and Python 3.10.

C ALS Dataset Details

C.1 Data Collection

The ALS (Action Locomotion Suite) dataset is collected using Unreal Engine 5 (UE5). We utilize the UE5 third-person character template integrated with a custom script. This script logs the character's root position, bone orientations, and interactive control signals (keyboard and mouse) at 30 FPS. The collected data is then retargeted to the SMPL-X body model and downsampled to 20 FPS for training and evaluation.

C.2 Dataset Statistics

The ALS dataset consists of 16 sequences, totaling 46,198 frames. This corresponds to approximately 38.5 minutes of motion at 20 FPS. The dataset covers a wide range of interactive behaviors, including walking and running on uneven terrain, jumping across gaps, and climbing obstacles. We split the dataset into a training set (12 sequences) and a test set (4 sequences).

C.3 Terrain Representation

We use a local terrain heightmap patch as the terrain representation. The patch is extracted from a 2.0m window centered at the character and resized to 64×64.

D MOTION MATCHING BASELINE SETUP

We use the raylib/C++ reference implementation of Motion Matching by Holden (`https://github.com/orangeduck/Motion-Matching`, MIT licence) as our retrieval-based baseline. Because the implementation is built around BVH skeletal animation with hard-coded joint names, we adapt it to our SMPL-X data with the following pipeline: (i) we export the ALS training-split sequences from SMPL-X to BVH using the official SMPL-X tools and re-name root, foot, and hand joints to the names expected by `database.cpp`; (ii) we resample the original 30/20 FPS data to the implementation's expected 60 FPS via cubic interpolation on root translation and SLERP on joint rotations; (iii) we tag binary foot-contact labels with a 5 cm height threshold and a 0.05 m/s velocity threshold to populate the trajectory features. Database build time is under one minute; inference latency is dominated by per-frame feature search and remains real-time. Path-following targets are converted to trajectory feature vectors with the same conventions as the original demo.

E ACTION LIBRARY AND EVALUATION SETUP

E.1 ACTION LIBRARY FILTERING THRESHOLDS

To ensure the quality and distinctness of behaviors in the Action Library, we apply a set of motion filters and only add a sequence if it meets all thresholds summarized in Table 5.

Table 5: Filtering thresholds for adding a motion clip to the Action Library.

Criterion	Threshold
Minimum displacement ($D_{\min}$)	0.6 m
Minimum average speed ($V_{\min}$)	0.3 m/s
Minimum total angular variation ($\Theta_{\min}$)	$30.0°$
Minimum path length ($L_{\min}$)	0.8 m
Minimum primitives (K_{lib})	2 (≈ 0.8s)

E.2 ACTION LIBRARY RECONSTRUCTION SETUP

For Table 2 we curate library entries from sources that are absent from the ALS training distribution and report per-clip reconstruction quality. *BABEL val* candidates are obtained by substring-matching curated keyword sets against `proc_label`, `raw_label`, and `act_cat` fields: *acrobatics* = {backflip, cartwheel, handstand, somersault, frontflip, flip}; *upper-body* = {wave, clap, punch, throw, shake hand, arm raise, salute, hug}; *sports* = {kick, swing, shoot, dribble, serve, pitch, catch, dunk}. We retain at most 50 matched clips per category. The strict-OOD probe uses the first 8 stage-II SMPL-X clips from `AMASS/DanceDB`, which is never paired with BABEL labels. For each clip we run 10 primitives of LAM-conditioned rollout starting from the first valid history window: the source clip is encoded into action codes via the LAM, and the frozen denoiser regenerates the future motion from those codes. Traj. Error (root pelvis L2) and full-body MPJPE are averaged per-clip; FID is computed via handcrafted motion features pooled at the segment level (every primitive contributes one feature vector).

E.3 MASKEDMIMIC ON ALS

We use MaskedMimic (Tessler et al., 2024) as a cross-paradigm physics reference. The evaluation uses the official AMASS policy, dense pelvis targets, and flat-ground physics rollout with per-window ground-normalized initialization, without policy retraining. It obtains TE 2.98 m and zero falls in 252 rollouts. Dense position/rotation conditioning of the pelvis, both hands, both ankles, and head gives TE 3.10 m.

E.4 BASELINE CONFIGURATION

The "official" and "ALS-ft" labels distinguish released checkpoints from ALS-fine-tuned variants. OmniControl and MotionLCM use their native origin-centered coordinate and conditioning conventions; MaskedMimic uses the flat-ground, per-window normalization in App. E.3.

82-frame FS and T2M-FID evaluation. The OmniControl and MotionLCM variants, MaskedMimic, and A-MDM are evaluated from saved motion outputs in Y-up coordinates with 22 joints, 20 FPS, and 82 frames per window; longer OmniControl outputs are truncated to the first 82 frames. All 126 held-out windows are included for each method, with three seeds for A-MDM: 1,008 predicted windows and 126 GT windows in total, without selecting favorable subsets or skipping failed windows. FS uses foot joints 10 and 11, with the minimum foot height in each window as the ground reference. A frame pair is counted if either foot is below 0.05 m at both endpoints and both its instantaneous and five-frame-smoothed horizontal speeds exceed 0.5 m/s. The denominator is all 81 adjacent frame pairs, not only contact pairs. Percentages are averaged equally over windows, then over seeds for A-MDM. This is a floor-relative metric, not terrain contact consistency.

For these methods, FID uses the pretrained T2M motion evaluator distributed with MotionLCM. Y-up joints are converted to HumanML3D 263-D features; 81 feature frames are truncated to 80 and encoded into 512-D embeddings. Each method's 126 windows are compared with the same 126 GT windows. A-MDM FID is computed separately for each seed and then averaged, rather than pooling 378 predictions. The sample count is smaller than the embedding dimension.

TINMO receives recorded causal keyboard/mouse input, a local heightmap, and generated motion history, without reference-path coordinates. Motion Matching uses an ALS-training database and native future-path query features. Main-table DNO uses the official HumanML3D prior, 500 optimization steps, 10 ODE steps, and guidance 5.0; DARTControl uses the official BABEL checkpoint, 100 optimization steps, and guidance 5.0. Both receive target-path objectives and control-derived text.

MotionLCM and OmniControl use official HumanML3D checkpoints with dense pelvis/root constraints and text. We transform between Y-up and Z-up by a rotation and express control hints in the native origin-centered frame. The official checkpoints obtain TE 2.94/1.85 m, respectively. OmniControl is additionally fine-tuned for 8k steps at learning rate 10^{-5}; MotionLCM fine-tunes ControlNet with its released recipe while freezing the VAE and denoiser. Both adaptations use only the 12 ALS training sequences in HumanML3D 263-D format and keyboard-derived captions, yielding 1.44 m for MotionLCM and 1.15 m for OmniControl. The latter uses a 196-frame cap. A-MDM uses its released PPO `PathEnv` policy, four future target-path points, and a three-seed average.

Confidence intervals. For A-MDM, 10k window-bootstrap resamples give a 95% trajectory-error interval of $[2.014, 2.481]$ m around the three-seed mean 2.243 m. This interval describes variation within the benchmark's four test sequences.

Timing boundary. Motion Matching, DNO, DARTControl, and TINMO report end-to-end time per generated frame on an L40S, including adapted control calls and coordinate conversion. Other entries report time per effective ALS frame. For OmniControl, MotionLCM, and MaskedMimic, each reported value is the equal-weight mean over 126 windows of measured wall time divided by 82 effective ALS frames, converted to milliseconds. OmniControl reports 1138.71/1168.73 ms for official/ALS-ft checkpoints, and MotionLCM reports 124.54/127.83 ms. Their timers enclose the external process call, including startup and loading overhead; OmniControl generates 196 frames although only 82 enter the denominator. MaskedMimic reports 84.63 ms, including policy inference, physics stepping, and state collection but excluding initialization and subsequent coordinate mapping/resampling.

A-MDM reports 64.07 ms per effective ALS frame from a ten-window timing run, separate from its three-seed quality evaluation. Its synchronized timer encloses the generation loop but excludes model loading, reset, joint mapping, and resampling. The same record gives 43.06 ms per native 30-FPS generated frame; Table 1 uses the 82-frame ALS denominator. These timings describe the evaluated pipelines with their stated frame denominators and included operations; we do not derive cross-method speedup factors.

E.5 Text Prompt Mapping for Interactive Baselines

Many of our comparison baselines are text-conditioned (text-to-motion) methods. To enable a fair evaluation on ALS sequences collected with interactive keyboard/mouse controls, we convert the

recorded control signals into a compact text prompt describing the intended locomotion command. We first map direction key presses to a default locomotion template "walk + direction": `W` (walk forward), `S` (walk backward), `A` (walk while sidestepping left), and `D` (walk while sidestepping right). `Shift` is treated as a gait modifier: when it co-occurs with any locomotion direction key, the locomotion verb is switched from "walk" to "run" (e.g., `Shift+W` → "run forward"). When multiple direction keys are pressed simultaneously, we describe their composition in a single prompt by combining the forward/backward axis with the lateral sidestep axis (e.g., `Shift+W+A` → "run forward while sidestepping left"). If opposing keys on the same axis are pressed together (e.g., `W+S` or `A+D`), we treat that axis as neutral and omit the corresponding directional phrase in the prompt. If both axes are neutral, we use an idle template "" and only keep other active clauses (crouch/jump) if present. We additionally map `Alt` to "crouch" and `Space` to "jump"; these transient actions are appended as additional clauses when active.

E.6 User Study Protocol

Fifteen volunteer participants from our research community viewed rendered motion clips and rated motion naturalness and tracking accuracy on a five-point Likert scale. Each participant rated 20 randomly selected clips from TINMO and the baselines. The study collected anonymous ratings without personally identifying information. Figure 5 reports the rating distributions for the original video stimuli; the later baseline reruns and ALS-fine-tuned variants in Table 1 were not re-rated. The results are descriptive; no statistical significance test is reported.