Interactive World Simulation with Cosmos for Physical AI

NVIDIA[1]

Abstract

We present a shared framework that adapts Cosmos world foundation models for real-time interactive world simulation in Physical AI. By streaming visual observations in response to actions and scene controls, the framework enables users and embodied agents to interact with evolving virtual environments. It supports both predicting the visual consequences of actions and generating observations from physical state maintained by an external simulator. We demonstrate this approach across camera navigation and robot manipulation, controllable video transfer, and multi-view autonomous vehicle (AV) simulation. These examples provide a shared foundation for developing interactive virtual environments for Physical AI training and evaluation. Our code and model checkpoints are available at github.com/nvidia/cosmos and huggingface.co/collections/nvidia/cosmos3-examples. The project website is available at research.nvidia.com/labs/cosmos-lab/cosmos-simulation.

Open-Source Code	
Cosmos	github.com/nvidia/cosmos
Cosmos-Framework	github.com/nvidia/cosmos-framework

Open-Weight Model Checkpoint	
Cosmos3-Nano-Sim-Bimanual	huggingface.co/nvidia/Cosmos3-Nano-Sim-Bimanual
Cosmos3-Nano-Sim-Transfer	huggingface.co/nvidia/Cosmos3-Nano-Sim-Transfer
Cosmos3-Nano-Sim-Auto	huggingface.co/nvidia/Cosmos3-Nano-Sim-Auto
Cosmos3-Nano-Transfer-Auto	huggingface.co/nvidia/Cosmos3-Nano-Transfer-Auto

[1]Contributors and acknowledgments are listed in Appendix A.

© 2026 NVIDIA. All rights reserved.

Contents

1 Introduction **4**

2 Model Architecture **5**
- 2.1 Cosmos 3 Backbone . . . 5
- 2.2 Multimodal Token Arrangement . . . 6
- 2.3 Temporal-causal Attention . . . 9

3 Post-training Recipe **10**
- 3.1 Bidirectional Adaptation . . . 11
- 3.2 Causal Training with Teacher Forcing . . . 12
- 3.3 Causal Consistency Distillation With Teacher Forcing . . . 13
- 3.4 Distribution Matching Distillation with Self Forcing . . . 14

4 Streaming Inference **16**
- 4.1 Streaming State and Memory . . . 17
- 4.2 Runtime Optimization . . . 18

5 Post-training Example: Cosmos3-Nano-Sim-Bimanual **20**
- 5.1 Architecture . . . 21
- 5.2 Data Preparation . . . 22
- 5.3 Post-training . . . 23
- 5.4 Inference Optimization . . . 24
- 5.5 Results . . . 25

6 Post-training Example: Cosmos3-Nano-Sim-Transfer **30**
- 6.1 Architecture . . . 31
- 6.2 Data Preparation . . . 31
- 6.3 Post-training . . . 33
- 6.4 Inference Optimization . . . 34
- 6.5 Results . . . 35

7 Post-training Example: Cosmos3-Nano-Sim-Auto **42**
- 7.1 Architecture . . . 42
- 7.2 Data Preparation . . . 48
- 7.3 Post-training . . . 49
- 7.4 Infrastructure and Inference Optimization . . . 51
- 7.5 Results . . . 53

8 Related Work **60**
- 8.1 Controllable World Models for Physical AI . . . 61
- 8.2 Causal Video Generation and Few-step Distillation . . . 62
- 8.3 Efficient Inference for Streaming Video Generation . . . 62

9 Conclusion **63**

A Contributors and Acknowledgments **64**
- A.1 Contributors . . . 64
- A.2 Acknowledgments . . . 65

B Cosmos3-Nano-Sim-Bimanual: Additional Details **66**
B.1 Data and Caption Construction . . . 66
B.2 Training Configuration . . . 66
B.3 Decoder Studies . . . 67
B.4 Evaluation Metrics . . . 67
B.5 Training Ablations and Development Studies . . . 68

C Cosmos3-Nano-Sim-Transfer: Additional Details **71**
C.1 Data and Sampling Details . . . 71
C.2 Runtime Profiling Details . . . 71
C.3 Supplementary Benchmark Analysis . . . 73
C.4 Isaac Lab Evaluation Details . . . 75
C.5 Stage Comparisons and Failure Cases . . . 77

D Cosmos3-Nano-Transfer/Sim-Auto: Additional Details **85**
D.1 Architecture Details . . . 85
D.2 Data Preparation . . . 90
D.3 Training Configuration . . . 95
D.4 Runtime Optimization . . . 97
D.5 Inference and Benchmark Settings . . . 101
D.6 Supplementary Results and Ablations . . . 103

1. Introduction

Physical AI enables autonomous machines, such as robots and vehicles, to extend perception and reasoning into the physical world (NVIDIA, 2025c). Since their actions influence future observations, these systems must continually adapt to changing contexts: a robot adjusts its manipulation strategy for different object poses, while a vehicle responds to road and traffic conditions. This continuous feedback makes perception, prediction, and action interdependent (NVIDIA, 2026a), making Physical AI consequential for tasks like object handling, spatial navigation, and vehicle operation. Building these capabilities requires exposing systems to diverse environments and interventions, while also testing how they respond when conditions change.

Gathering all such experience solely in the real world is costly and can entail risks to people and equipment (NVIDIA, 2025c). Simulation offers a safe and controllable environment for training, testing, and debugging before deployment. Cosmos world foundation models provide synthetic training data and generate observations for learning in virtual environments (NVIDIA, 2025c, 2026a). Learned visual priors expand the range of appearances and scene conditions in these environments, while structured controls retain grounding in the external physical state. As a result, world simulation bridges the gap between data available for learning and the environments needed to evaluate policy behavior.

Interactive simulation generates observations incrementally as new controls arrive from a user, a policy, or an external simulator (Gao et al., 2026a; Basant et al., 2026). Evaluating this behavior requires testing how changed controls affect subsequent observations and how well scene changes persist over time. To enable live policy or human interaction at real-world timescales, observations must be returned within the required response time. Achieving this in neural world simulation demands coupling visual generation with control fidelity, retention of state changes, and responsive execution. These requirements motivate transforming a world foundation model into an interactive simulator, while showing that it meets downstream needs still requires independent validation.

We introduce a shared post-training framework for converting Cosmos world foundation models into interactive simulators. Built on Cosmos 3 (NVIDIA, 2026a), the framework adapts broad visual knowledge to predicting how scenes evolve under supplied actions or scene controls. Its purpose is to make that knowledge usable throughout an ongoing interaction, so a user, policy, or external simulator can shape the experience as it unfolds. This supports both learning visual dynamics from recorded experience and generating observations guided by an external simulator's physical state.

The framework combines causal training and distillation with streaming execution in a shared recipe for interactive simulation (Zheng et al., 2026c). Post-training teaches the model to follow domain-specific controls and continue from its own generated observations (Huang et al., 2025b). Distillation reduces the computation needed to produce those observations, while streaming execution supports repeated interaction as new inputs become available. Together, these components address visual quality, control fidelity, and timely responses within a common framework. The shared recipe provides a reusable starting point for specialized simulators, with each domain contributing its own controls, data, and evaluation requirements.

We ground this approach in three post-trained Cosmos3-Nano model examples:

1. Cosmos3-Nano-Sim-Bimanual predicts future visual dynamics from a streaming sequence of camera navigation and robot manipulation actions.
2. Cosmos3-Nano-Sim-Transfer supports single-view video transfer based on spatial controls, while the state can be maintained externally by a simulator.
3. Cosmos3-Nano-Sim-Auto supports synchronized multi-view AV simulation with LiDAR.

All three examples follow the shared procedure, each adapting control representations, data preparation, and modeling choices for its domain. They connect robot interaction, simulator-guided video transfer, and AV scene generation through one post-training framework. Quality, control fidelity, and runtime are evaluated under

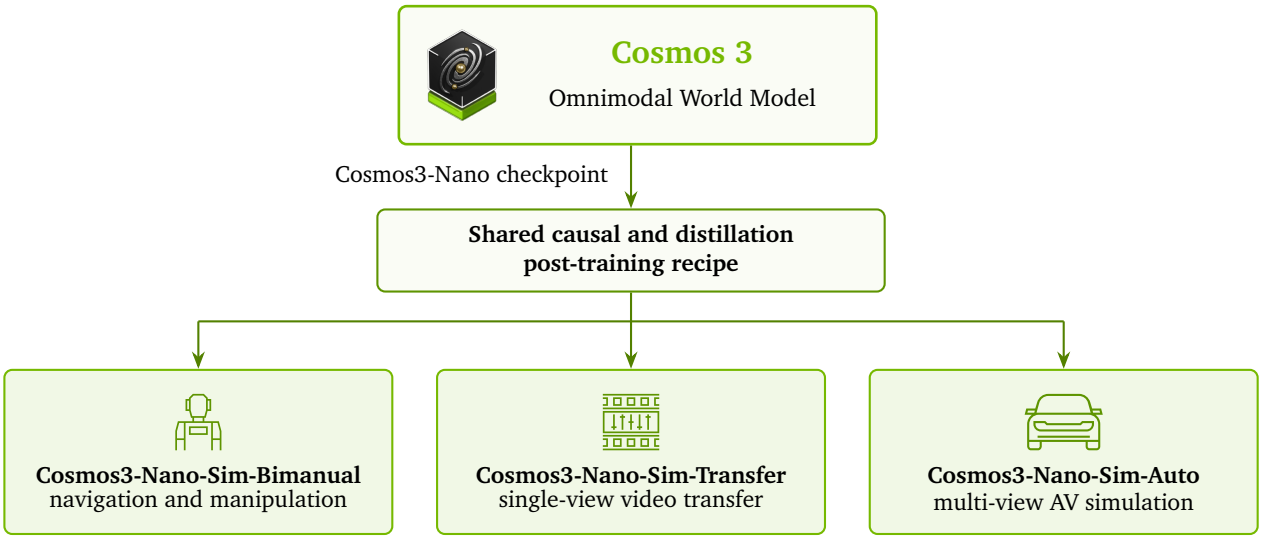


Figure 1: **From Cosmos 3 to interactive world simulation.** A shared post-training recipe adapts Cosmos3-Nano from the Cosmos 3 family (NVIDIA, 2026a) into three example models for camera navigation and robot manipulation, single-view video transfer, and multi-view autonomous vehicle (AV) simulation. Each specialized model incorporates domain-specific controls tailored to its application and trained on datasets reflecting the types of interactions and physical dynamics relevant to that domain.

each example's stated protocols, with policy-driven simulator utility requiring separate closed-loop validation. Fig. 1 summarizes this shared foundation and its three post-trained specializations.

The technical report is organized as follows. Sec. 2 introduces the shared model architecture and its support for simulation controls and causal generation. Sec. 3 presents the post-training recipe for causal training and distillation on the foundation model. Sec. 4 describes streaming inference optimizations for efficient execution. Sec. 5 to 7 describe the three model examples, covering their respective architectures, data preparation, training procedures, and evaluation results. Sec. 8 discusses related work and Sec. 9 concludes the paper.

2. Model Architecture

Our framework builds directly on Cosmos 3, retaining its reasoner and generator towers, unified representation space, and positional embedding formulation (NVIDIA, 2026a). To support interactive simulation, the architecture must generate observations incrementally as conditioning inputs arrive. The primary shared architectural modification is the introduction of temporal-causal attention across video chunks. This temporal-causal attention ensures that, as each new control becomes available, the simulator predicts subsequent observations autoregressively, without access to future target chunks. During generation, each chunk is produced through iterative denoising, conditioned on both the available observation history and any provided control signals, including language, actions, source video, or structured visual controls.

2.1. Cosmos 3 Backbone

All three example models build directly on Cosmos 3 checkpoints, preserving their multimodal backbone, representation space, and architectural foundations. At the core of Cosmos 3 is a Mixture-of-Transformers (MoT) design, which organizes processing into two complementary towers: the reasoner and the generator (Deng et al., 2025; NVIDIA, 2026a). Modality-specific encoders and projections first embed each input type (*e.g.* language, videos, actions) into a shared representation space. Within this architecture, the reasoner tower specializes in autoregressive next-token prediction for language, while the generator tower handles non-language modalities through iterative denoising.

MoT towers and joint attention. In each MoT layer, the reasoner and generator towers maintain separate sets of parameters for layer normalization, query/key/value projections, and feed-forward networks. The simulator architecture routes the text prompt into the reasoner tower as the autoregressive subsequence. Visual, control, and denoising-related tokens (including clean conditions, denoised history, and noisy prediction targets) are routed into the generator tower as the diffusion subsequence. Generator queries attend to the entire diffusion subsequence and, via dual-stream joint attention, can also attend to keys and values from the reasoner, while language tokens in the reasoner retain standard causal self-attention. During causal conversion, the generator tower replaces full temporal visibility with temporal-causal attention masking, governed by token role, physical time, and the availability of conditions (Sec. 2.2 and 2.3).

Visual encoding and generation. Visual features are extracted by a vision transformer (ViT) (Dosovitskiy et al., 2021) and fed to the reasoner tower, while the generator tower receives latent representations from a causal video variational autoencoder (VAE) (NVIDIA, 2026a). A causal RGB tokenizer achieves $4\times$ temporal compression, mapping $1{+}4n$ input frames to $1{+}n$ latent time steps (Wan et al., 2025). The tokenizer's first latent time step represents the initial RGB frame, and each subsequent step adds 4 RGB frames. Learned projections merge latent patches into the unified representation space. Sampling integrates the output projection's rectified flow velocities into the latents (Liu et al., 2023), and the tokenizer decoder reconstructs RGB observations. For autonomous driving, we extend this design to jointly generate synchronized multi-view RGB videos and LiDAR; see Sec. 7.1 for details.

Multimodal positional embedding. Cosmos 3 applies 3D multimodal RoPE (MRoPE) (Bai et al., 2025) to its embeddings, dividing the rotary channels of each attention head among temporal, height, and width coordinates (t^{pos}, h, w) (NVIDIA, 2026a). Within each latent time, both source and target visual tokens share the temporal coordinate t^{pos}, with their spatial coordinates tiling a grid that resets at every frame. Language tokens are encoded with increasing, identical values along all three axes, while action tokens carry temporal positional information and set $h = w = 0$. A large temporal gap (15,000) separates the autoregressive and diffusion subsequences, and both conditioning and generated visual streams initiate from a common temporal origin. These instantiations inherit the foundation model's MRoPE formulation and maintain the same coordinate offsets throughout causal conversion.

Conditioning and control. Cosmos 3 accommodates both language and control streams as direct conditions for video generation (NVIDIA, 2026a). Control tokens are concatenated alongside noisy target video tokens and remain clean throughout the denoising of targets. For video transfer, the video tokenizer can process source video or structured visual controls (*e.g.* depth, segmentation, edges, or blurred videos) to steer generation. In action-conditioned video generation, action tokens express relative camera, end-effector motions, or other transformations. Domain-specific projections adapt action representations of varying dimension into the unified representation (Zheng et al., 2026b).

2.2. Multimodal Token Arrangement

The framework groups visual targets and aligned clean controls into *chunks* for temporal-causal attention and autoregressive video generation. A chunk contains the spatial tokens at one or more consecutive latent video time steps together with their corresponding control tokens. For multi-view generation, it includes the aligned visual groups from the participating views. This prediction unit is distinct from the tokenizer's internal units for encoding or decoding video.

Each prediction uses generation history and the provided control streams. For action-conditioned video generation, controls specify the desired actions and transformations that determine the upcoming predicted visual outcomes. For video transfer tasks, observations for the target interval are guided by either a source video or structured visual controls. Both configurations may include language tokens as conditions, follow the same next-chunk prediction structure, and maintain distinct token roles and attention rules.

Multimodal inputs for chunk prediction. Partition the L target latent time steps into ordered, disjoint, contiguous index sets $\mathcal{B}_1, \ldots, \mathcal{B}_J$. Chunk j spans $N_j = |\mathcal{B}_j| \geq 1$ target latent time steps, with $\sum_{j=1}^{J} N_j = L$. After the initial tokenizer step, these N_j latent time steps correspond to $4N_j$ new RGB frames per view. Its visual part $\mathcal{V}_j = \mathcal{V}_{\mathcal{B}_j}$ contains all spatial tokens at those latent time steps, and all of its target visual tokens are jointly denoised. At diffusion step τ, the noisy tokens for chunk j are denoted $\widetilde{\mathcal{V}}_j^{\tau}$.[2] The aligned controls $\mathcal{U}_j$ belong to the same chunk, and their token types and lengths depend on the post-trained model. The current generator chunk and its logical prediction input are

$$\begin{aligned} \mathcal{G}_j^{\tau} &= [\mathcal{U}_j, \widetilde{\mathcal{V}}_j^{\tau}], \\ \mathcal{S}_j(\tau) &= [\mathcal{T} \mid \mathcal{H}_j, \mathcal{G}_j^{\tau}], \end{aligned} \tag{1}$$

where $\mathcal{T}$ denotes available language tokens and $\mathcal{H}_j$ denotes retained clean generator history. Only $\widetilde{\mathcal{V}}_j^{\tau}$ is noisy and predicted. The controls remain clean and are excluded from the target loss, even though they are processed inside the same chunk.

Autoregression between video chunks routes visual history and targets through the generator, with new chunks produced by denoising. The notation describes which token sets are included in prediction, but not their memory layout. At inference, the clean history $\mathcal{H}_j$ retained before predicting chunk j is represented by its K/V cache $\mathcal{H}_j^{\mathrm{KV}}$. These history tokens do not need to be physically concatenated with the current input.

Within the input, only the current target chunk is noisy. The clean generator history $\mathcal{H}_j$ may include an initial prefix $\mathcal{H}_0$, previously used clean control groups $\mathcal{U}_k$, and denoised vision chunks $\widehat{\mathcal{V}}_k$ for $k < j$. When sampling starts from provided controls, the initial prefix can be empty or may contain initial observations from the target sequence. Cached controls remain as conditioning information, not part of the generative targets.

Once denoising produces $\widehat{\mathcal{V}}_j$, the completed clean chunk is $\widehat{\mathcal{G}}_j = [\mathcal{U}_j, \widehat{\mathcal{V}}_j]$. The runtime constructs its clean layerwise K/V and commits the completed history before predicting the next chunk (Eq. (21)). If language tokens are unchanged and no history eviction or reselection occurs, the next logical prediction input is given by

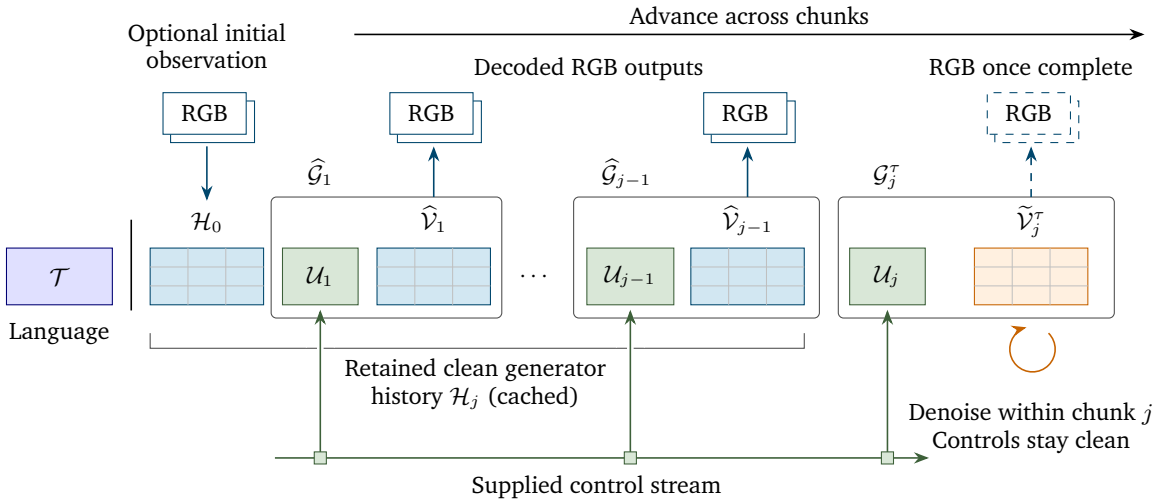
$$\begin{aligned} \mathcal{S}_{j+1}(\tau) &= [\mathcal{T} \mid \mathcal{H}_{j+1}, \mathcal{U}_{j+1}, \widetilde{\mathcal{V}}_{j+1}^{\tau}] \\ &= [\mathcal{T} \mid \underbrace{\mathcal{H}_j, \mathcal{U}_j, \widehat{\mathcal{V}}_j}_{\mathcal{H}_{j+1}}, \mathcal{U}_{j+1}, \widetilde{\mathcal{V}}_{j+1}^{\tau}]. \end{aligned} \tag{2}$$

With a limited-size K/V cache, adding new entries may require evicting older history tokens to keep the retained history within capacity. For the single-view action-conditioned path, clean K/V are refreshed in latent-time order within the completed chunk. Multi-view execution follows the clean-history visibility rules configured in Sec. 7.1.

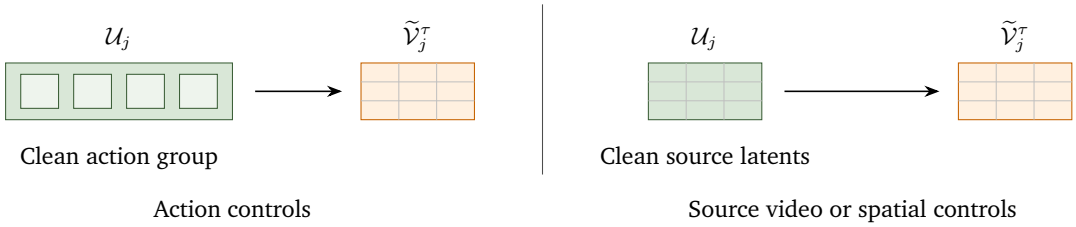
Streaming control inputs. In autoregressive video generation, each target chunk j pairs with a group of clean control tokens $\mathcal{U}_j$ matched to the chunk's physical interval. Each generator chunk $\mathcal{G}_j^{\tau}$ contains both its clean controls $\mathcal{U}_j$ and noisy visual targets $\widetilde{\mathcal{V}}_j^{\tau}$. The next control group belongs to the following chunk and is introduced for that prediction. Within a chunk, controls remain fixed and clean while denoising proceeds, and they continue serving as conditions once committed to history. All required controls must be available before sampling the chunk starts. During deployment, a policy or external simulator must therefore supply the corresponding action or control group before the model jointly denoises its visual part. If these inputs arrive incrementally, larger chunks can require a longer wait to collect the controls for their physical interval. The logical meaning of token roles and temporal association is unaffected by how data is laid out in memory.

Figure 2 illustrates the control-group arrangement for both actions and source video. For action-conditioned

[2] Plain letters denote scalars and indices. Bold lowercase letters are vectors, bold uppercase letters are matrices, and calligraphic uppercase letters are collections. Token sequences are ordered and may include repeated entries. Discrete token IDs use scalar labels.



(a) Autoregressive video generation at inference step j

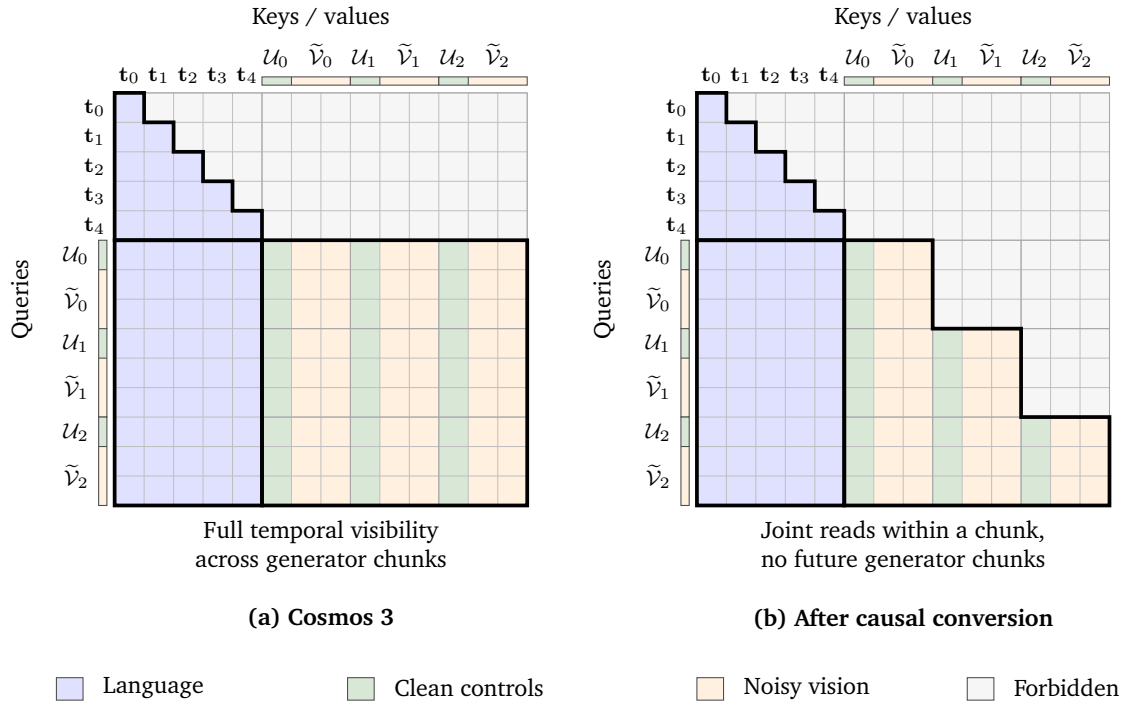


(b) Two example control streams for $\mathcal{U}_j$

Figure 2: **Autoregressive video generation with streaming controls.** (a) Each generator chunk groups clean controls $\mathcal{U}_k$ with the corresponding visual tokens. The history $\mathcal{H}_j$ comprises previously denoised chunks $\widehat{\mathcal{V}}_k$ and the retained controls. The current chunk $\widetilde{\mathcal{V}}_j^\tau$ is progressively denoised with clean controls held constant. (b) The action-based example shows 4 individual clean action tokens per latent interval. For video transfer, aligned source latent frames form the clean control group $\mathcal{U}_j$ for the target chunk's interval. Source tokens entering the cache keep their conditioning role for future predictions.

generation, $\mathcal{U}_j$ contains the transition actions corresponding to the current predicted chunk. For video transfer, clean source tokens $\mathcal{V}^{\text{src}}$ supply observations or structured controls, with alignment to the target's timestamp enforced, even if source and target use the same tokenizer and projection. This scheme can be extended to extra input signals, such as audio. Source tokens may be pre-encoded or provided incrementally, as long as all tokens for the interval are available before target denoising. After each prediction, the corresponding generated chunk and any new control tokens are appended to the history for subsequent predictions. Section 3 details the specific training targets and roles for each group.

Temporal alignment. Each control group is temporally aligned with its associated target chunk: actions carry their transition times, source video matches target timestamps, and spatial controls are matched to their target frames. FPS modulation transforms physical time to rotary temporal coordinates. For video transfer, source video is encoded at the target's frame rate and temporal compression, ensuring matched temporal increments. Since actions are recorded at 1 per RGB frame with a $4\times$ temporal compression, every step between latent vision frames covers 4 transition actions. Actions thus use $1/4$ of the rotary temporal increment used for vision tokens, resulting in the last action sample sharing a temporal coordinate with the corresponding vision latent frame. Action groups for chunks spanning multiple latent frames are collected accordingly. The first observation in the sequence lacks a preceding real action transition.



(a) Cosmos 3 **(b) After causal conversion**

Figure 3: **Temporal-causal attention for action-conditioned video generation.** Rows represent queries and columns stand for keys/values. Each generator chunk contains clean controls $\mathcal{U}_k$ and visual tokens $\widetilde{\mathcal{V}}_k$, with chunk indices starting at $k = 0$. When conditioning on actions, $\mathcal{U}_0$ is a null group and the first vision group is the clean frame $\mathcal{V}_0$. Language uses causal self-attention, while generator queries access reasoner keys/values, but not vice versa. Token counts are schematic only. (a) In Cosmos 3, every generator chunk can read from every other. (b) After causal conversion, reads are allowed only within the current block and to earlier ones. Video-transfer and multi-view layouts can further restrict token-role visibility.

Sequence packing. Cosmos 3 reduces padding by packing recordings with different durations and resolutions into shared token buffers (NVIDIA, 2026a). Each recording always retains its own temporal origin, token roles, and attention domain, regardless of where tokens are packed within shared buffers. To ensure recording isolation, a boolean attention mask is applied for every query and key index. This mask ensures that only tokens belonging to the same recording session, at valid (non-padding) positions, and allowed by cross-tower, temporal, and token-role rules are able to attend to each other. Padding is excluded from both attention and loss computation. Device partitions do not define attention boundaries: tokens from the same recording may exchange information across devices, and tokens from different recordings remain isolated—even if packed on the same device. A retained prefix must preserve its source recording's identity and temporal coordinates.

2.3. Temporal-causal Attention

Cosmos 3 uses full self-attention in the generator tower to denoise entire video clips. Our post-training recipe replaces the generator's bidirectional temporal attention with a block-causal pattern to enable autoregressive video generation. Each attention block corresponds to one generator chunk $\mathcal{G}_j^\tau$, containing aligned clean controls and the visual targets for $N_j \geq 1$ latent time steps. The temporal mask operates on chunks, and the visual targets within the current chunk are jointly denoised. The action-conditioned layout permits mutual attention between the current clean actions and visual targets. Video-transfer layouts can impose additional restrictions on control queries while retaining the controls inside the same logical chunk. Teacher forcing additionally separates the clean-reference and noisy-target branches as described in Sec. 3.2.

Generator visibility. The generator's attention mask specifies which tokens each query is allowed to access while generating video. Within the current noisy target chunk $\widetilde{\mathcal{V}}_j^\tau$, queries may attend to all tokens in that chunk, enabling joint denoising. Queries can also attend to the retained clean history $\mathcal{H}_j$, which contains previously denoised frames and other finalized context. No attention is allowed to tokens from future target chunks. Supplied source video that coincides with the current interval remains accessible, but captions and cached conditioning must restrict visibility to prevent leaking information from future targets. Fig. 3 visually outlines which reads are allowed and which are blocked. In some cases, more specialized token-role masks are enforced, such as for cross-view attention (Sec. 7.1). Details on constructing these masks during training, particularly for teacher forcing, are given in Sec. 3.2.

Next-chunk factorization. Let $\mathbf{z}_{1:L}$ represent the L target latents for generated vision frames, omitting the initial supplied observations. Divide the frame indices into J prediction chunk sets $\mathcal{B}_1, \ldots, \mathcal{B}_J$. Define $\mathcal{I}_j$ as the conditioning accepted before generation begins for chunk j. Only this conditioning is exposed to its queries via the attention mask. The conditioning $\mathcal{I}_j$ comprises the accepted control groups, any supplied initial observations, initialization metadata, physical timestamps, and available language. With a predetermined sequence of such conditions, autoregressive generation factorizes as:

$$p_\theta(\mathbf{z}_{1:L} \mid \mathcal{I}_{1:J}) = \prod_{j=1}^{J} p_\theta\big(\mathbf{z}_{\mathcal{B}_j} \mid \mathbf{z}_{\mathcal{B}_{<j}}, \mathcal{I}_j\big)\,. \tag{3}$$

Here, $\mathbf{z}_{\mathcal{B}_{<j}}$ indicates all previously generated targets. The initial observed frame establishes the clean history $\mathcal{H}_0$ and counts as a conditioning input, not a generated output. If new inputs arrive during denoising, they only impact subsequent chunks unless the current prediction is interrupted or redone. Once a chunk prediction completes, its observations are appended to the clean history for the prediction of chunks that follow (Sec. 4.1). Temporal causality is thus enforced between chunks, but the frames within a chunk interact freely in joint denoising. The chunk size sets the granularity of this interaction: smaller chunks permit the model to respond to new interactive controls more often, while larger chunks capture longer visual intervals during joint modeling and may enable higher visual quality.

This generative factorization assumes conditioning inputs are predetermined and presented from outside the system, not arising from closed-loop feedback. Under closed-loop simulation, a policy would select new controls based on the model's previous outputs, so the conditioning sequence would depend on the simulated rollout. Running with a prerecorded sequence of controls covers conditional generation but does not exercise feedback from policy-driven interaction.

3. Post-training Recipe

Our post-training recipe converts the base Cosmos 3 architecture into a generator capable of continuing from its own predictions. This post-training process unfolds in four sequential stages, summarized in Fig. 4:

- **Stage 0:** Bidirectional adaptation trains on reference clips with full temporal visibility to improve domain and control fidelity over long contexts. The adapted teacher initializes causal conversion and remains the frozen bidirectional reference for stage 3.
- **Stage 1:** Teacher Forcing (**TF**) learns causal chunk prediction from clean reference history. Its multi-step checkpoint initializes the stage-2 student and supplies its frozen causal teacher.
- **Stage 2:** Teacher Forcing with discrete-time Consistency Models (**TF-dCM**) retains clean reference history while training a few-step student to reduce denoising evaluations and preserve conditional behavior.
- **Stage 3:** Self Forcing with Distribution Matching Distillation (**SF-DMD**) trains on autoregressive student rollouts to address exposure bias and refine perceptual quality. The stage-0 bidirectional teacher and a trainable auxiliary score model evaluate the generated clips.

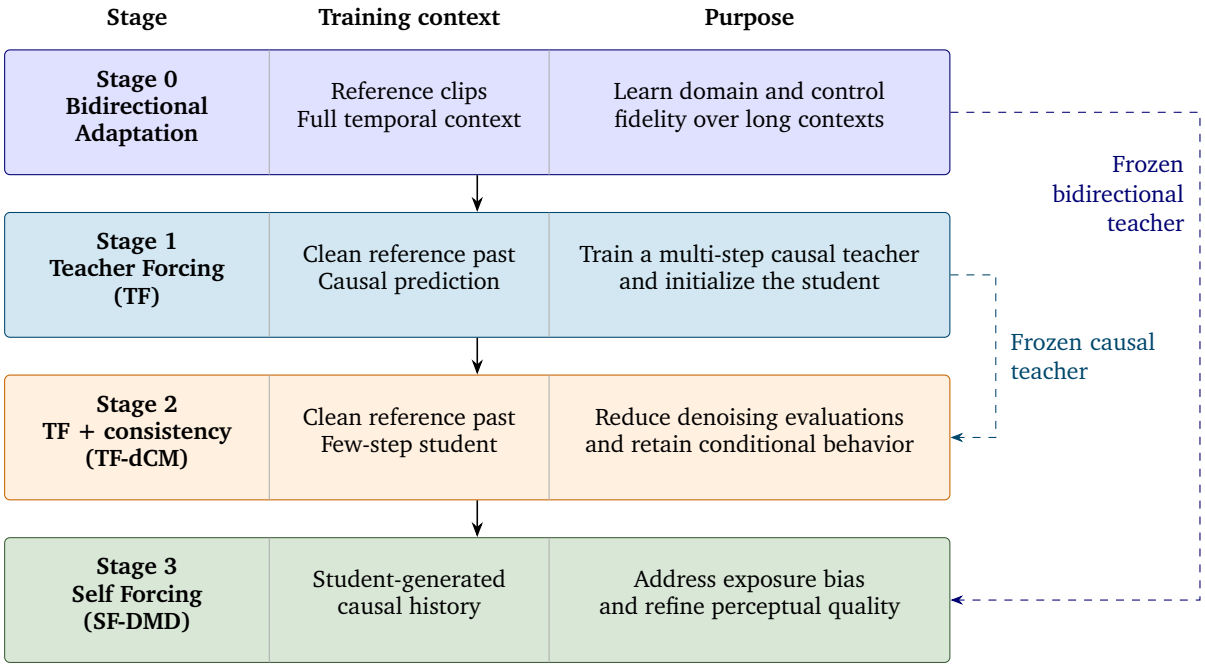


Figure 4: **Shared post-training stages for interactive world simulation.** Stages follow Causal-rCM (Zheng et al., 2026c) from top to bottom, with context and purpose alongside. Solid arrows initialize the next stage. Dashed paths supply the frozen stage-1 causal teacher for TF-dCM and the frozen stage-0 bidirectional teacher for SF-DMD. SF-DMD also uses a trainable auxiliary score model to evaluate student rollouts.

The causal distillation curriculum employs the same recipe as Causal-rCM (Zheng et al., 2026c). Conditioning tokens, such as actions, source video tokens, structured visual controls, and initial observations, are treated as clean conditions and are excluded from the loss.

3.1. Bidirectional Adaptation

Stage 0 adapts the bidirectional Cosmos 3 generator to the observation domain and conditioning interface used by each example. Action-conditioned records pair transition actions with the observations they affect, while video-transfer records align source controls and target observations over the same physical interval. The teacher retains full temporal attention while learning these domain-specific relationships. Long-context adaptation extends the temporal span of contiguous condition/observation records presented with their aligned captions. It exposes the teacher to longer visual histories before the subsequent stages restrict temporal visibility and train autoregressive generation. The adapted weights initialize causal conversion and supply the frozen bidirectional teacher for SF-DMD. Sec. 5 to 7 describe the example-specific data, sequence lengths, and adaptation schedules.

For one latent sequence, let $\mathbf{z}$ stack the clean latent targets over the complete training clip, excluding supplied target observations. The conditioning $\mathcal{C}$ contains language, aligned controls, and any supplied observations, all held clean during denoising. We train the bidirectional predictor with rectified flow (Liu et al., 2023):

$$\begin{aligned} \mathbf{x}_\tau &= (1-\tau)\mathbf{z} + \tau\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ L_{\text{bidir}} &= \mathbb{E}_{\mathbf{z},\mathcal{C},\boldsymbol{\epsilon},\tau}\left[w_{\text{bidir}}(\tau) \left\| \mathbf{v}_\theta^{\text{bidir}}(\mathbf{x}_\tau, \tau; \mathcal{C}) - (\boldsymbol{\epsilon} - \mathbf{z}) \right\|_2^2 \right]. \end{aligned} \tag{4}$$

The expectation jointly samples each clip and its conditions, independent Gaussian noise, and a diffusion time $\tau \in [0, 1]$ shared across the clip's generated targets. For the loss definitions below, let C, T, H, W denote the channel count, number of latent frames, and latent height and width of the full latent sequence for a clip, including any conditioning frames. We set $w_{\text{bidir}}(\tau) = 1/(CTHW)$, normalizing the summed squared velocity

error over generated frames by the full sequence size. We write the loss weights as functions of diffusion time for generality; in this work, they are independent of τ. Full temporal attention lets each target location read all noisy target times and the supplied clean conditions. Only generated target coordinates are noised and contribute to the loss.

Bidirectional adaptation and TF use nonuniform noise-time sampling followed by a resolution-dependent shift, as specified in App. B.2.

3.2. Causal Training with Teacher Forcing

The number of latent time steps in each chunk defines both the prediction unit and the temporal context available for modeling. Each example selects a nominal prediction chunk size for its causal training stages. Chunks also contain the aligned clean controls, which are held fixed while their visual targets are denoised. Example-specific choices for chunk size are summarized in Sec. 5 to 7. The RGB tokenizer processes $1 + 4n$ input frames and produces $1 + n$ latent time steps (Wan et al., 2025). The initial latent time step is a singleton at startup, followed by chunks of at least 1 latent time step representing 4 new RGB frames per step. An observed initial frame provides clean conditioning and is excluded from the generated targets. When no initial frame is supplied, the model also generates that startup singleton.

A sampled training example specifies its target latents, supplied conditions, and chunk partition. Using the latent-time partition $\mathcal{B}_j$ from Sec. 2.2, let $\mathbf{z}^j$ flatten the generated visual coordinates at those times, excluding controls and supplied observations. The noisy target $\mathbf{x}_\tau^j$ and noise $\boldsymbol{\epsilon}^j$ are the corresponding chunk restrictions of the rectified flow variables in Eq. (4). Chunks without generated coordinates contribute no loss. Under teacher forcing, the retained clean generator history $\mathcal{H}_j$ contains reference targets, previously supplied controls, and any initial target observations (Sec. 2.2). The additional conditioning $\mathcal{C}_j = (\mathcal{T}, \mathcal{U}_j)$ comprises the available language and the clean controls grouped with the current visual targets. A single diffusion time τ is sampled per training example and shared across its noisy target chunks. Physical-time chunk indices and diffusion times remain distinct even when several targets are evaluated together.

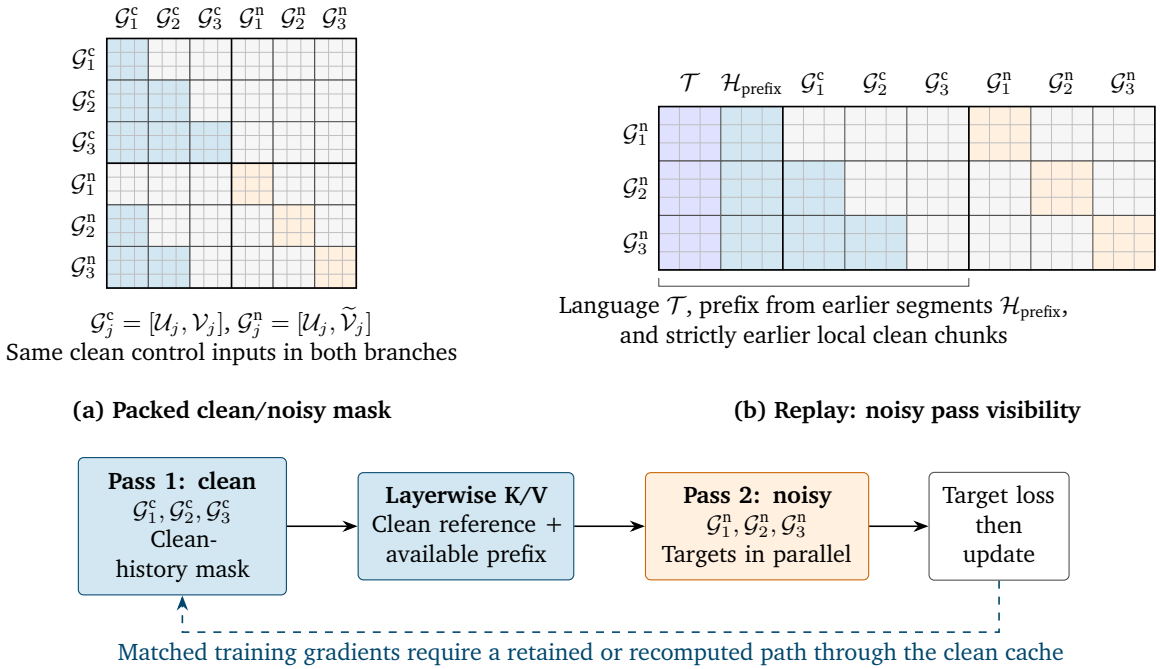
Causal denoising objective. Let $\mathbf{v}_\theta^{\mathrm{TF}}$ denote the velocity prediction for the noisy target chunk, conditioned on the clean reference history $\mathcal{H}_j$ and supplied inputs $\mathcal{C}_j$. The denoising loss is

$$L_{\mathrm{TF}} = \mathbb{E}_{\mathbf{z},\mathcal{C},\boldsymbol{\epsilon},\tau} \sum_j w_j^{\mathrm{TF}}(\tau) \left\| \mathbf{v}_\theta^{\mathrm{TF}}(\mathbf{x}_\tau^j, \tau; \mathcal{H}_j, \mathcal{C}_j) - (\boldsymbol{\epsilon}^j - \mathbf{z}^j) \right\|_2^2 . \tag{5}$$

The expectation samples each example jointly with its conditions, independent standard Gaussian target noise, and a diffusion time using the mode sampling and shift in Sec. 3.1. TF uses the same full-sequence normalization as bidirectional training, applying it to each chunk's squared error before summing over chunks. Clean reference history and supplied conditions, including any initial image, provide context and are excluded from the supervised coordinates.

Clean and noisy teacher forcing masks. Teacher forcing separates clean reference inputs from noisy prediction targets, as illustrated in Fig. 5(a). Clean-reference queries attend only to causally available clean inputs. Noisy queries read clean history from strictly earlier chunks and attend jointly to controls and noisy visual tokens within their own prediction chunk. Controls remain clean throughout denoising. The masks prevent noisy queries from accessing the current clean visual target or future visual tokens, including indirectly through conditioning states or cached K/V.

Replay-based teacher forcing. Replay-based teacher forcing implements this visibility pattern in two passes, avoiding the complexity of a single attention mask that jointly handles clean and noisy tokens. The first pass processes clean references with the selected clean-history mask and stores their layerwise K/V. The second pass evaluates noisy targets in parallel using joint attention within each prediction chunk and cached clean



(a) Packed clean/noisy mask **(b) Replay: noisy pass visibility**

(c) Two-pass training with clean K/V reuse

Figure 5: **Packed and replay-based teacher forcing.** (a) The action-conditioned schematic shows 1 latent time step per chunk, with the same clean controls in both branches. Color identifies the clean-reference or noisy-target branch, while controls remain clean in both. The mask permits clean-to-clean attention, strictly earlier clean history for noisy queries, and joint reads within each noisy chunk. For larger single-view chunks, clean queries can read only their own latent interval and earlier clean intervals. (b) The replay noisy pass reads language $\mathcal{T}$, an available prefix $\mathcal{H}_{\text{prefix}}$ from earlier segments, and strictly earlier clean chunks. (c) The clean pass supplies layerwise K/V for parallel noisy-target evaluation, with shared parameters and a single optimizer update. The dashed arrow marks the retained or recomputed gradient path through the cache.

history from strictly earlier chunks. Action controls are reevaluated with their branch's visual tokens, while the transfer path routes the clean control K/V directly to its noisy target queries. Matching a packed forward also requires retaining or recomputing gradients through the clean cache. Fig. 5 illustrates the action-conditioned layout with 1 latent time step per schematic chunk.

3.3. Causal Consistency Distillation With Teacher Forcing

Stage 2 distills the causal flow model into a few-step generator to reduce denoising evaluations per chunk. With the clean reference history and controls fixed, the student learns a consistency map (Song et al., 2023) that predicts the same clean endpoint from different states along a teacher denoising trajectory:

$$f_\theta(\mathbf{x}_\tau^j, \tau; \mathcal{H}_j, \mathcal{C}_j) = \mathbf{x}_\tau^j - \tau \mathbf{v}_\theta^{\text{TF}}(\mathbf{x}_\tau^j, \tau; \mathcal{H}_j, \mathcal{C}_j), \tag{6}$$

$$f_\theta(\mathbf{x}, 0; \mathcal{H}_j, \mathcal{C}_j) = \mathbf{x}. \tag{7}$$

The velocity-parameterized network is retrained by the consistency objective, so f_θ estimates a trajectory endpoint rather than the posterior mean implied by an optimal RF regression model. The boundary condition holds by construction for finite network outputs.

We sample noise-time pairs (τ, s) with $0 \le s < \tau \le 1$, sharing each pair across target chunks. The sampling rule and resolution-dependent shift are specified in App. B.2. The initial $\mathbf{x}_\tau^j$ is drawn from the rectified flow corruption in Eq. (4), restricted to chunk j. The frozen causal teacher ψ_{AR} numerically advances this state

toward lower noise:

$$\widehat{\mathbf{x}}_s^j = \mathrm{ODESolve}\big(\mathbf{v}_{\psi_{\mathrm{AR}}}^{\mathrm{CFG}}, \mathbf{x}_\tau^j, \tau \to s; \mathcal{H}_j, \mathcal{C}_j\big)\,. \tag{8}$$

We implement $\mathrm{ODESolve}$ with a single Euler step, keeping history and controls fixed. The field $\mathbf{v}_{\psi_{\mathrm{AR}}}^{\mathrm{CFG}}$ includes the configured classifier-free guidance (CFG) (Ho and Salimans, 2021). For conditional and reference-branch velocities $\mathbf{v}_{\mathrm{cond}}$ and $\mathbf{v}_{\mathrm{base}}$, our guidance convention is $\mathbf{v}^{\mathrm{CFG}} = \mathbf{v}_{\mathrm{base}} + \gamma(\mathbf{v}_{\mathrm{cond}} - \mathbf{v}_{\mathrm{base}})$, with $\gamma = 1$ giving the conditional field. Guidance scales and reference-branch conditioning are described in the model-specific training appendices.

The student matches endpoint predictions at the two noise levels:

$$L_{\text{TF-dCM}} = \mathbb{E}_{\mathbf{z},\mathcal{C},\boldsymbol{\epsilon},\tau,s} \sum_j w_j^{\mathrm{dCM}}(\tau)\, \big\| f_\theta(\mathbf{x}_\tau^j, \tau; \mathcal{H}_j, \mathcal{C}_j) - \mathrm{sg}\big[f_\theta(\widehat{\mathbf{x}}_s^j, s; \mathcal{H}_j, \mathcal{C}_j)\big] \big\|_2^2\,. \tag{9}$$

The expectation samples training examples, target noise, and the diffusion-time pairs defined above. We set $w_j^{\mathrm{dCM}}(\tau) = 1/(CHW)$, averaging the squared error over channels and spatial locations and summing over generated latent frames. We then average these losses across latent sequences. Both evaluations use the current student parameters, with gradients stopped through the entire target evaluation, including its clean-history cache. The teacher and student branches share the same reference history and controls, with targets constructed online.

TF-dCM evaluates the consistency map using the two-pass replay layout in Sec. 3.2. A clean-reference pass constructs layerwise K/V, followed by a noisy-target pass that reads strictly earlier clean prefixes to predict the endpoint at each noise level. Each network and teacher CFG branch constructs its own clean K/V using its own parameters and conditioning. The student reuses its clean-reference cache for its two endpoint evaluations within the same update. Gradients through that cache are retained or recomputed for the online prediction, while the complete target branch is detached. The student learns the guided teacher trajectories and uses only a conditional forward pass per denoising step at inference.

3.4. Distribution Matching Distillation with Self Forcing

Stage 2 learns few-step denoising with clean reference history, whereas autoregressive inference conditions on previously generated chunks. Starting from the stage-2 student, stage 3 applies Self Forcing Distribution Matching Distillation (SF-DMD) to optimize rollouts conditioned on the student's own generated history (Huang et al., 2025b; Yin et al., 2024). The retained cache $\mathcal{H}_j^{\mathrm{KV}}$ represents the history $\mathcal{H}_j$ before chunk j, including previous control tokens and generated visual chunks. Within chunk j, K_j denoising evaluations follow the selected schedule $1 = \tau_{j,K_j} > \ldots > \tau_{j,0} = 0$, alternating endpoint prediction and Gaussian re-noising:

$$\widetilde{\mathbf{x}}_{\tau_{j,n-1}}^j = (1 - \tau_{j,n-1}) f_\theta(\widetilde{\mathbf{x}}_{\tau_{j,n}}^j, \tau_{j,n}; \mathcal{H}_j^{\mathrm{KV}}, \mathcal{C}_j) + \tau_{j,n-1}\boldsymbol{\epsilon}_{j,n}, \tag{10}$$

$$\widetilde{\mathbf{x}}_{\tau_{j,K_j}}^j \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{11}$$

The transition is applied for $n = K_j, \ldots, 1$, with independent standard Gaussian draws $\boldsymbol{\epsilon}_{j,n}$. The selected schedule depends on the exit budget K_j. The recurrence specifies forward sample values, with completed endpoint $\widehat{\mathbf{z}}^j = \widetilde{\mathbf{x}}_0^j$ corresponding to visual chunk $\widehat{\mathcal{V}}_j$. Before advancing, the runtime completes the configured clean-history refresh and retention operations from Sec. 4.1, with gradients detached. The single-view action path refreshes latent frames in temporal order within the completed chunk.

Training differentiates only the final denoising evaluation of each chunk:

$$\widehat{\mathbf{z}}^j = f_\theta\Big(\mathrm{sg}[\widetilde{\mathbf{x}}_{\tau_{j,1}}^j], \tau_{j,1}; \mathrm{sg}[\mathcal{H}_j^{\mathrm{KV}}], \mathcal{C}_j\Big)\,. \tag{12}$$

This expression has the same forward value as the rollout endpoint, but stops gradients through earlier denoising

evaluations and generated history (Huang et al., 2025b).

Teacher and auxiliary score model. The frozen stage-0 teacher ψ_{bidir} supplies a reference score estimate, while a trainable auxiliary model ϕ estimates the score of the current student distribution (Yin et al., 2024; Zheng et al., 2026c). Both score models use bidirectional temporal attention over the rollout during training.

Let $\widehat{\mathbf{z}}_i$ stack the nonempty set of generated target coordinates of rollout i, excluding supplied observations and controls. Here $\mathcal{C}_i$ denotes the full supplied conditioning sequence, initial observations, and target layout, rather than the local chunk conditioning $\mathcal{C}_j$. Let $\boldsymbol{\xi}_i$ collect the student rollout's sampling noise and exit-policy choices. For fixed $\mathcal{C}_i$, let p_θ^0 be the conditional law of these outputs under draws of $\boldsymbol{\xi}_i$. For either the student law or a fixed reference law p_{ref}^0, define p_a^τ as the law of $(1-\tau)\mathbf{Z} + \tau\boldsymbol{\epsilon}$ with $\mathbf{Z} \sim p_a^0$ and independent standard Gaussian $\boldsymbol{\epsilon}$. The ideal distribution-matching objective is the noisy reverse KL (Yin et al., 2024):

$$L_{\text{DMD,raw}} = \mathbb{E}_{\mathcal{C}_i,\tau}[D_{\text{KL}}(p_\theta^\tau(\cdot \mid \mathcal{C}_i) \| \, p_{\text{ref}}^\tau(\cdot \mid \mathcal{C}_i))]\,. \tag{13}$$

The expectation samples conditioning examples and a configured score time with $0 < \tau < 1$, with finite expected KL assumed. The rollout law includes the selected exit and history-retention policies, which need not coincide with those used for evaluation.

To estimate the student score, the auxiliary model is trained on re-noised, detached student outputs:

$$\mathbf{y}_{i,\tau} = (1-\tau)\,\text{sg}[\widehat{\mathbf{z}}_i] + \tau\boldsymbol{\epsilon}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{14}$$

$$L_{\text{aux}}(\phi) = \mathbb{E}_{\mathcal{C}_i,\boldsymbol{\xi}_i,\tau,\boldsymbol{\epsilon}_i}\left[\lambda_{\text{RF}}(\tau)\,\text{mean}\left[|\mathbf{v}_\phi(\mathbf{y}_{i,\tau},\tau;\mathcal{C}_i) - (\boldsymbol{\epsilon}_i - \text{sg}[\widehat{\mathbf{z}}_i])|^2\right]\right]. \tag{15}$$

The expectation samples conditioning examples, student rollouts, score times, and corruption noise. Score times are sampled uniformly before applying the resolution-dependent noise shift in App. B.2. The RF time weight is $\lambda_{\text{RF}}(\tau) = 1$. Here and below, mean averages over all generated coordinates of the scored stream, including its temporal extent, with absolute values and squares applied elementwise. For $\tau > 0$, the squared velocity error equals squared clean-endpoint error divided by τ^2, using $f_a(\mathbf{y},\tau) = \mathbf{y} - \tau\mathbf{v}_a(\mathbf{y},\tau)$ for each score model a.

For a clean law p_a^0, let $\mathbf{m}_{a,\tau}(\mathbf{y};\mathcal{C}) = \mathbb{E}_a[\mathbf{Z} \mid \mathbf{Y}_\tau = \mathbf{y}, \mathcal{C}]$ be its posterior endpoint mean. Gaussian corruption relates this mean to the exact score (Yin et al., 2024):

$$\mathbf{s}_{a,\tau}(\mathbf{y} \mid \mathcal{C}) = \nabla_{\mathbf{y}} \log p_a^\tau(\mathbf{y} \mid \mathcal{C}) = \frac{(1-\tau)\mathbf{m}_{a,\tau}(\mathbf{y};\mathcal{C}) - \mathbf{y}}{\tau^2}, \qquad 0 < \tau < 1. \tag{16}$$

An exact RF regression model satisfies $f_a = \mathbf{m}_{a,\tau}$. The teacher and auxiliary networks approximate these posterior means, whereas the consistency student's endpoint map has a different training objective.

With fixed reference, conditioning and sampling laws, exact scores, and the regularity needed to interchange differentiation and expectation, the ideal pathwise gradient is

$$\nabla_\theta L_{\text{DMD,raw}} = \mathbb{E}_{\mathcal{C}_i,\boldsymbol{\xi}_i,\tau,\boldsymbol{\epsilon}_i}\left[\left(\frac{\partial\widehat{\mathbf{z}}_i}{\partial\theta}\right)^{\mathsf{T}} \frac{(1-\tau)^2}{\tau^2}\left(\mathbf{m}_{\theta,\tau}(\mathbf{y}_{i,\tau};\mathcal{C}_i) - \mathbf{m}_{\text{ref},\tau}(\mathbf{y}_{i,\tau};\mathcal{C}_i)\right)\right]. \tag{17}$$

The expectation uses the same draws as the auxiliary loss. The derivative follows the full rollout with sampled noise and exit choices held fixed, before the truncation in Eq. (12). One factor of $1-\tau$ comes from the RF corruption Jacobian and the other from the score-to-endpoint conversion.

The practical update replaces exact posterior means by score-model predictions and uses adaptive normaliza-

tion (Yin et al., 2024; Zheng et al., 2026c):

$$\begin{aligned}\rho_{i,\tau} &= \mathrm{mean}[|\mathrm{sg}[\widehat{\mathbf{z}}_i] - f_{\psi_{\text{bidir}}}(\mathbf{y}_{i,\tau}, \tau; \mathcal{C}_i)|]\,, \\ \mathbf{g}_{i,\tau} &= \frac{f_\phi(\mathbf{y}_{i,\tau}, \tau; \mathcal{C}_i) - f_{\psi_{\text{bidir}}}(\mathbf{y}_{i,\tau}, \tau; \mathcal{C}_i)}{\rho_{i,\tau} + \delta}, \qquad \delta > 0.\end{aligned} \tag{18}$$

We use the additive stabilizer $\delta = 10^{-6}$ in all three models. The generator fits a detached endpoint offset:

$$L_{\text{DMD}}(\theta) = \mathbb{E}_{\mathcal{C}_i, \boldsymbol{\xi}_i, \tau, \boldsymbol{\epsilon}_i}\left[\frac{1}{2a_i}\,\|\widehat{\mathbf{z}}_i - \mathrm{sg}[\widehat{\mathbf{z}}_i - \mathbf{g}_{i,\tau}]\|_2^2\right]. \tag{19}$$

The expectation uses the same conditioning, rollout, time, and noise sampling as the auxiliary loss. Here $a_i > 0$ specifies the loss reduction separately from the adaptive normalizer $\rho_{i,\tau}$. Auxiliary updates optimize L_{aux} over ϕ using detached student outputs. Generator updates optimize L_{DMD} over θ through Eq. (12), with both score models and the complete pseudo-target held fixed.

For the single-view RGB models, $a_i = CHW$, giving a sum of per-frame means over the rollout. For multimodal generation, such as joint RGB–LiDAR generation in Cosmos3-Nano-Sim-Auto, we apply this objective separately to each modality and combine their losses, as detailed in App. D.3.2. Holding the entire pseudo-target fixed gives derivative $\mathbf{g}_{i,\tau}/a_i$ with respect to the generated endpoint. Adaptive normalization replaces the ideal diffusion-time factor, and final-call differentiation replaces the full rollout Jacobian by its truncated counterpart. Together with approximate score models, these choices define a practical surrogate motivated by reverse-KL matching, rather than an exact gradient of $L_{\text{DMD,raw}}$. Teacher CFG, when used, also replaces the reference prediction by a guided prediction, which need not correspond to the Gaussian corruption of one fixed clean reference law. Teacher guidance and the alternating update schedule are specified for each example in Sec. 5 to 7.

Replayed backpropagation. Even with gradient truncation, differentiating the full rollout together retains final-step activations for every chunk. Replayed backpropagation reduces this peak activation memory. A rollout generated without gradients stores each chunk's final noisy input, diffusion time, detached cached prefix, and detached DMD target. We then recompute and backpropagate the final prediction one chunk at a time, accumulating gradients before a single optimizer update. With model parameters, saved inputs, pseudo-targets, conditioning, and stochastic choices fixed, replay preserves the selected truncated gradient estimator at the cost of recomputation. Each chunk retains the same rollout-level loss reduction, so gradient accumulation differentiates the same sum of losses. This equivalence does not restore gradients through earlier denoising evaluations or generated history.

Exit policy. The exit policy selects the number of student denoising evaluations for each chunk. The `max` rule uses the configured per-chunk maximum, matching inference with the same sampling budget. The SF-DMD recipes for the three reported final models use `max` with 4 denoising evaluations per chunk. Under final-step gradient truncation, this concentrates direct supervision at the schedule's last, lowest-noise input. The `cycle` rule cycles through the available counts across updates, subject to each chunk's maximum. Changing the exit depth allows different denoising evaluations to receive direct DMD supervision. Cycling reduces the average rollout computation and broadens gradient coverage, but spreads updates across exit depths. For fixed θ, p_θ^0 under a cycle denotes the mixture of rollout laws induced by the cycle's frequencies, which the auxiliary model is trained to estimate. That mixture generally differs from the fixed-budget law used by `max`, even when their history policies agree.

4. Streaming Inference

Few-step distillation reduces the number of generator evaluations per chunk, but each prediction still requires condition preparation, history access and updates, and decoding. Streaming inference seeks to optimize

per-chunk work by reusing valid state from earlier computations. We describe the generation loop and its memory requirements, the runtime methods that this state enables, and the timing boundaries used to evaluate performance. The examples in Sec. 5 to 7 apply these methods to their conditioning interfaces.

4.1. Streaming State and Memory

Streaming inference organizes execution into discrete chunks, each dictated by the temporal-causal attention described in Sec. 2. The initialization step establishes any available clean history from provided target observations. As the model progresses from one chunk to the next, it continually reuses language representations, clean generator keys and values (K/V), and the causal decoder state between predictions.

Next-chunk prediction. Autoregressive inference samples one chunk at a time, using generated observations in the history rather than the clean reference targets used by TF training (Sec. 3.2). For chunk j, the supplied conditions $\mathcal{U}_j$ encode the available actions, source video, or spatial controls. The sampler processes $\mathcal{G}_j^\tau = [\mathcal{U}_j, \widetilde{\mathcal{V}}_j^\tau]$, jointly denoising its visual targets with language and retained clean history as context. During sampling, the conditions are fixed and history remains unchanged across the denoising steps. New conditions can influence the following unsampled chunk. Sampling requires all controls for the current chunk to be available. Larger chunks can amortize execution overhead, but also extend the physical interval covered before the next condition update is accepted. Once sampling is complete, the resulting latents $\widehat{\mathcal{V}}_j$ feed into two branches: the K/V cache update prepares for the next prediction, and causal decoding reconstructs observations (Fig. 6). Decoder state maintains temporal information across chunks. Both branches can execute concurrently on separate GPUs.

Append clean K/V to history. History $\mathcal{H}_j$ tracks clean conditions and completed observations. After denoising, the runtime refreshes clean K/V from the completed chunk, including its controls and generated latents:

$$\Delta\mathcal{H}_j^{\mathrm{KV}} = \mathrm{RefreshKV}_\theta(\widehat{\mathcal{G}}_j; \mathcal{H}_j^{\mathrm{KV}}, \mathcal{T}_j^{\mathrm{KV}}), \tag{20}$$

The generator appends the resulting layerwise K/V and applies the configured history-retention rule:

$$\mathcal{H}_{j+1}^{\mathrm{KV}} = \mathrm{Retain}_{j+1}\left(\mathcal{H}_j^{\mathrm{KV}} \oplus \Delta\mathcal{H}_j^{\mathrm{KV}}\right). \tag{21}$$

Here, $\widehat{\mathcal{G}}_j = [\mathcal{U}_j, \widehat{\mathcal{V}}_j]$ is the completed clean chunk, and $\Delta\mathcal{H}_j^{\mathrm{KV}}$ contains its new clean K/V. The operator $\mathrm{RefreshKV}_\theta$ denotes the complete history-refresh procedure for the chunk and can contain several internal forward passes. The cache $\mathcal{T}_j^{\mathrm{KV}}$ represents the available language $\mathcal{T}$. The operation $\oplus$ appends entries with their associated token roles and timestamps. The operator Retain_{j+1} selects the entries kept for the next prediction, and is the identity when full history is retained. The resulting $\mathcal{H}_{j+1}^{\mathrm{KV}}$ represents the retained history $\mathcal{H}_{j+1}$ for the subsequent prediction.

Chunk progression waits until the history refresh is complete. The runtime processes the completed latent frames in temporal order, each with its aligned clean actions and allowed earlier history. Therefore, later clean frames in the chunk can read refreshed earlier frames, matching the frame-causal clean branch of TF (Sec. 3.2). Joint denoising within the prediction chunk and ordered clean-history construction serve different roles. Multi-view backends can instead construct clean K/V with their configured chunk-causal mask.

For K_j denoising evaluations and R_j clean-history evaluations, the continuing chunk costs $K_j + R_j$ generator evaluations. The reference single-view path can refresh all N_j latent frames separately. A backend that batches the required refresh into one forward pass has $R_j = 1$.

Lifetime of context in the K/V caches. State in this loop persists for different durations depending on its type. Language representations can be reused as long as their inputs and attention dependencies remain unchanged. Generator history is stable within each target block and only updates when new entries are committed, enabling all denoising steps for a block to share the same clean K/V (Zheng et al., 2026c). Supplied conditions retain

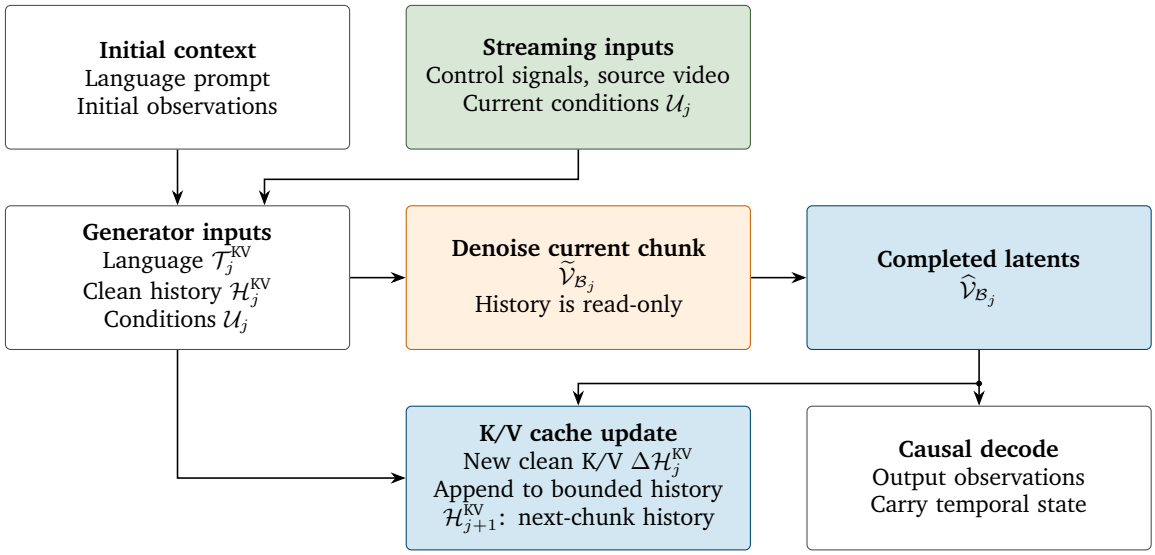


Figure 6: **State and dependencies in one streaming generation step.** Initial context supplies language and any available observed history, while streaming inputs provide the current conditions $\mathcal{U}_j$. Denoising reads these inputs without modifying retained K/V. Completed latents feed the cache update and stateful decoding, which can overlap on separate GPUs. The generator refreshes clean K/V for the completed chunk and appends the retained entries before the next prediction (Eqs. (20) and (21)). For single-view action conditioning, this refresh processes latent frames in temporal order within the chunk.

their representations if they are independent of noisy targets. During denoising, the read path leverages attention buffers until history updates, avoiding extra concatenation, allocation, or padding in each iteration.

Bounded history with initial sinks. Reusing history avoids redundant computation, but retaining all past observations would still increase storage and attention cost as the rollout grows. To bound history, the cache maintains a fixed initial sink region together with a rolling window of recent clean entries, as in StreamingLLM (Xiao et al., 2024). Sink entries keep initial visual context, while the rolling window retains recent motion. When new history is committed, the cache replaces older non-sink entries. Storage locations keep logical timestamps and token roles, so eviction does not renumber observations. Valid-length or role metadata can exclude unused capacity from attention. After the rolling window fills, both the stored K/V and active attention context stay bounded. This limits access to older observations but ensures a fixed history capacity for the runtime methods described below.

History retention is a separate choice from whether history is recorded or generated. A model may retain the full history during causal training, but employ a fixed-size window with an attention sink during evaluation (Sec. 5.3 and 5.5). SF-DMD exposes the student to its own generated observations, but this alone does not remove a difference between training and evaluation retention policies. The selected history policy therefore remains part of the evaluated operating point.

4.2. Runtime Optimization

We optimize autoregressive inference by reducing the cost of each generator call and scheduling work across the available GPUs. The objective is to minimize control-to-observation latency while sustaining the required throughput. Communication, synchronization, and duplicated work can offset the savings from parallelism.

Optimization metrics and execution cost. Let t_{in} be the time the runtime receives an input control assigned to a future chunk, and t_{obs} the time its first affected generated observation is ready after decoding. The control-to-observation latency is defined as

$$T_{\text{control}\to\text{obs}} = t_{\text{obs}} - t_{\text{in}}, \tag{22}$$

This interval includes waiting for the rest of the chunk's required controls, condition preparation, generation, decoding, and any synchronization or bookkeeping that delays the corresponding output. Conditions remain fixed during denoising, so a newly received control can affect only a subsequent unsampled chunk (Sec. 4.1). Let $t_{\text{controls},j}$ be the time all required inputs for chunk j are available and $t_{\text{history},j}$ the time its clean-history cache is ready. The start of denoising satisfies

$$t_{\text{start},j} \geq \max\left(t_{\text{controls},j}, t_{\text{history},j}\right). \tag{23}$$

Collecting inputs over a longer chunk interval can therefore increase response latency even when the model sustains the required output throughput. Controls supplied as a complete plan or prerecorded stream need not incur this input-collection delay. The timing boundary here is decoded-output readiness, with display or transport delay counted separately when relevant to deployment.

Let F count output timestamps produced in elapsed generation time T, with physical duration D. We use

$$\text{FPS}_{\text{sync}} = \frac{F}{T}, \qquad \text{RTF} = \frac{T}{D}. \tag{24}$$

The measurement states the execution boundary for T. At a fixed rate r, a sequence including its initial frame spans $D = (F-1)/r$, while F new steady-state outputs span $D = F/r$. Real-time throughput requires $\text{RTF} \leq 1$ for the specified workload and timing boundary. Throughput measures generation capacity, while control-to-observation latency measures responsiveness.

To identify optimization targets, the sequential per-chunk execution budget is

$$T_{\text{chunk}} = T_{\text{condition}} + T_{\text{denoise}} + T_{\text{cache}} + T_{\text{decode}} + T_{\text{bookkeeping}}, \tag{25}$$

where T_{cache} is the duration of the clean K/V update required before the next prediction. With overlap, measured wall time follows the schedule's dependency critical path, including transfer and synchronization costs. The methods below reduce repeated execution overhead, distribute transformer work, and overlap independent reconstruction and history updates.

Minimizing compilation and launch overhead. Every denoising pass reuses the same transformer layers. Reusing the compilation artifacts and reducing the launch overhead improves total efficiency. Once the rolling attention window reaches its configured capacity, the generation component operates with a fixed history context size even as new entries are committed (Sec. 4.1). Padding the language tokens to a fixed capacity further stabilizes the cache shapes, while valid-length metadata avoids attending to unused rows. Fixed shapes, preallocated metadata, and persistent storage allow static transformer compilation (PyTorch Contributors, 2026a) and CUDA-graph replay (Nguyen et al., 2021).

Maskless attention implementation. Each generator call involves attention over the language, cached history, and the current chunk, following the temporal-causal structure in Sec. 2. Constructing large pairwise masks adds computation overhead. Maskless attention partitions the allowed attention targets into groups and computes each group separately, reducing the computational overhead. Furthermore, log-sum-exp statistics merge these outputs to ensure that every valid key is counted just once (Ye et al., 2025).

Context parallelism. Dividing the transformer computation across multiple GPUs allows for parallel workloads but introduces communication at each attention stage. The design is based on the Ulysses-style context-parallel approach from Cosmos 3 (NVIDIA, 2026a; Jacobs et al., 2023). Tokens are distributed across the GPUs. During each attention layer, the first all-to-all exchange groups tokens by attention head for local computation. The second exchange restores the original token order.

The system can also use replicated input and output to change the communication pattern. Each GPU keeps a full copy of the current chunk's hidden state while only splitting the cached K/V among attention heads. Every GPU computes local projections and then combines results using all-reduce. This replaces the corresponding all-to-all exchanges with an all-reduce while repeating projection and feed-forward computation. The resulting communication volume and runtime depend on the tensor layout, collective implementation, and GPU count. Replicated computation trades against communication, so this layout does not guarantee lower latency than the original layout.

Distribute camera work. Multi-view execution additionally distributes camera loading, eligible encoding, decoding, and saving across the inference replica. Decoded outputs are collected for continuation, which retains communication and host-storage costs. This reduces redundant media processing around the transformer without changing the model's attention visibility. As camera count grows, distributing sensor processing becomes useful when loading, encoding, or decoding limits throughput. Joint RGB and LiDAR generation uses modality-specific reconstruction and returns temporally aligned sensor observations.

Compiled encoding and decoding. Compilation also applies to repeated condition-encoding and observation-decoding operations. Stable input shapes allow these operations to reuse compiled execution while preserving any temporal state carried between chunks. These operations have compilation requirements independent of whether the generator history window is full.

Decoder performance–quality tradeoff. The pipeline supports compiled Wan 2.2 VAE (Wan et al., 2025) and TAEHV Super (Boer Bohan, 2025) as selectable RGB decoders. TAEHV Super uses a lighter reconstruction network while retaining the Wan encoder and generator latent representation. Users can select either decoder according to their application's reconstruction quality and performance requirements.

Asynchronous decoding. Once a chunk's denoising is complete, the latents are sent to a decoder that runs on a separate GPU. The decoder transforms these latents into RGB observations, producing the outputs seen by downstream tasks or users. At the same time, the generator remains active on its original GPU, using the just-completed latents and conditions to compute and append clean K/V to history. Performing decoding and history updates in parallel allows the system to overlap the two computation branches. The overlap can shorten the critical path relative to sequential execution when its savings exceed the added transfer and synchronization costs. The next prediction waits for its clean-history update, while downstream consumers wait for the required decoded observations. Measured throughput and control-to-observation latency must account for these distinct dependencies under the selected schedule.

5. Post-training Example: Cosmos3-Nano-Sim-Bimanual

We demonstrate three models that use the shared causal and distillation recipe, each initialized from Cosmos3-Nano (NVIDIA, 2026a). Cosmos3-Nano-Sim-Bimanual predicts visual dynamics for camera navigation and manipulation action streams. Cosmos3-Nano-Sim-Transfer generates single-view observations from spatial control streams. Cosmos3-Nano-Sim-Auto extends video transfer to multiple views with synchronized cameras and LiDAR streams for AV simulation. All three examples use the same training recipe described in Sec. 3 with different interfaces, supervision, and temporal layouts. For each example model, we explain the architectural choices, data preparation, post-training steps, inference optimization, and results.

Cosmos3-Nano-Sim-Bimanual uses a shared action interface to predict observations for camera navigation

and robot manipulation. Camera motion changes which parts of a scene are visible, while manipulation also changes the configuration of the robot and nearby objects. Both require a generator that follows supplied actions while preserving a coherent scene as its own predictions become history. We specialize Cosmos3-Nano with a common action representation and a mixture of camera, robot, and human videos, then apply the shared causal and distillation curriculum. We evaluate the 4-step student's visual quality, action-conditioned observation agreement, and streaming cost across these domains.

5.1. Architecture

Cosmos3-Nano-Sim-Bimanual predicts RGB observations from an initial image, language description, and a sequence of time-indexed actions. A single generator covers camera navigation and robot manipulation through the shared action space inherited from Cosmos 3 (NVIDIA, 2026a). Domain-specific preprocessing and learned projections accommodate differences in coordinates and effectors, while the visual generator models their consequences. The generator predicts each observation using the projected command and shared scene history.

Action representation. A single 59-dimensional action vector unifies all control commands. This vector includes a 9D ego pose, a 24D right wrist and fingertip encoding along with right gripper width, and a 24D left wrist and fingertip encoding along with left gripper width. Each pose representation consists of 3 translation values and a 6D rotation code (Zhou et al., 2019). Each 24D hand structure combines a 9D wrist pose with positions of 5 fingertips, each specified by a 3D coordinate relative to the wrist.

The action pipeline first maps each source's native fields into this 59D layout, converting coordinates, units, and gripper conventions as required. A validity mask records which fields are observed, so missing measurements can be identified during preprocessing rather than interpreted as physical commands. The domain-specific normalization applies an inverse hyperbolic sine transform before a learned projection maps the vector into the model's feature space. The embedded action at time t is

$$\mathbf{h}_t^{\text{act}} = \text{Proj}_{d_t}(\mathbf{m}_t \odot \text{Norm}_{d_t}(\mathbf{a}_t)), \tag{26}$$

where $\mathbf{a}_t$ is the unified action vector, d_t specifies its domain, $\mathbf{m}_t$ is the validity mask, and Norm_{d_t} encodes coordinate conversion and normalization. The domain selects the appropriate projection after the common layout has been constructed.

Temporal alignment. Pose actions use anchors that update every 16 RGB frames. For an observed pose $\mathbf{T}_t$ in anchor interval j, the action encodes $\mathbf{T}_{b(j)}^{-1}\mathbf{T}_t$, where $b(j)$ is the observation immediately preceding that interval. The corresponding global and frame-relative alternatives encode $\mathbf{T}_0^{-1}\mathbf{T}_t$ and $\mathbf{T}_{t-1}^{-1}\mathbf{T}_t$, respectively. The interval anchor expresses motion in a common local frame without allowing its displacement to grow over the entire trajectory. Its 16-frame span defines the pose reference and is independent of the generator's prediction chunk size.

Groups of 4 consecutive per-frame actions are paired with each latent visual interval (Fig. 7). The causal model predicts chunks of 2 latent frames, corresponding to 8 new RGB frames after startup. Thus an anchor interval spans 4 latent intervals, or 2 prediction chunks, rather than a single model step. Actions remain supplied conditions, and the prediction loss applies only to RGB targets. The initial observation precedes physical motion and is paired with null actions. This startup condition is semantically different from a later valid zero-motion command, which asks the model to preserve the commanded pose.

Mixed-action sequences. The interface also permits contiguous navigation and manipulation spans in one action sequence. Each transition retains its domain tag and validity mask, and the corresponding preprocessing and projection map it into the shared generator's input space. When the domain changes, generation continues from the retained visual history without loading another model or restarting the rollout. This implements a persistent session across control modes.

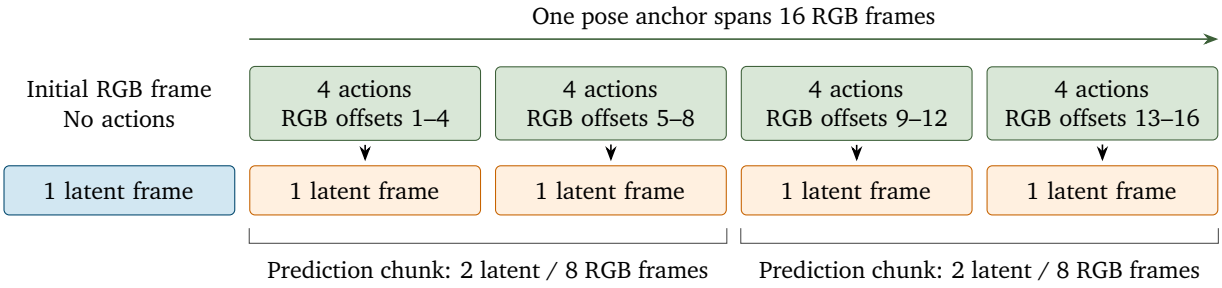


Figure 7: **Cosmos3-Nano-Sim-Bimanual temporal alignment.** The leftmost clean latent encodes the initial RGB frame, before any physical action. After startup, 4 per-frame actions align with each latent visual interval, and 2 latent frames form an 8-RGB-frame prediction chunk. The 16-RGB-frame pose-anchor interval spans the 4 subsequent latent frames and excludes the initial observation.

Table 1: **Cosmos3-Nano-Sim-Bimanual data sources and conditioning roles.** All records pair RGB observations with aligned actions and language.

Source	Action information	Language condition
Camera videos, mostly drone and egocentric	Camera poses, supplied or estimated	Environment description independent of camera motion
AgiBot (Bu et al., 2025) and YAM (I2RT Robotics, n.d.)	Embodiment-specific robot fields	Rewritten initial-state description
Mecka (Mecka AI, n.d.)	Human wrist/fingertip and available ego fields	Rewritten initial-state description
Behavior-1K (Li et al., 2023)	Simulated embodiment actions	Supplied task description

5.2. Data Preparation

Sources and sampling. Training combines camera trajectories, recorded robot demonstrations, and human egocentric interactions (Tab. 1). Camera videos expose the generator to scene changes caused by viewpoint motion, while robot and human videos show effectors interacting with objects. Egocentric human interaction data broadens the interaction coverage and contributes to robot generalization. We match RGB observations, actions, and captions by their source video and physical interval. Each training window aligns the observed motion with the commands supplied for prediction.

Filtering and alignment. For in-the-wild videos without raw pose labels, we use ViPE (Huang et al., 2025a) to estimate camera trajectories. We filter out trajectories with low confidence or high jitter by checking pose continuity and smoothness with an empirical threshold. For robot and human egocentric sequences, we extract long continuous episodes and normalize embodiment-specific fields. This normalization covers coordinates, moving-base states, and gripper conventions, transforming them into standard actions and validity masks. Unavailable fields are masked or excluded from the corresponding task. These checks directly assess whether each supplied action matches the observed transition.

Captioning. Language describes the scene in which an action takes place, while the action stream specifies the controlled motion. Captions should supply appearance, spatial relationships, and relevant context without giving a second, potentially conflicting account of the commanded trajectory. A teacher model first generates a structured scene breakdown and a detailed general caption, then rewrites the caption for the control domain. The prompts prioritize accuracy and coverage of visible content, instructing the model to generalize or omit uncertain details. The scene breakdowns and rewritten captions supervise separate in-house captioners for camera navigation, robotics, and human egocentric video. These captioners annotate the corresponding collections at scale.



Initial frame ($t = 0$)

Source caption (excerpt)

"…a white pair with rounded fronts, white soles, tan interiors, and busy multicolored floral or illustrated uppers sits near the right cabinet wall. …"

"The right gripper moves inward from frame-right, closes on the patterned pair near their upper openings, lifts both together from the tile, …"

Final caption (excerpt)

"…a white pair with rounded fronts, white soles, tan interiors, and busy multicolored floral or illustrated uppers sits near the right cabinet wall. …"

Figure 8: **Initial-state caption rewriting on an AgiBot-GEAR example.** The initial frame shows the patterned slippers on the floor beside the right cabinet wall. The excerpts compare the first caption segment with its final rewrite. Pink shading marks the removed grasp and lift, while the initial location is retained. Ellipses omit other caption text. The image is shown for comparison only: the rewriter receives the first-segment text, without the video or later segments.

For camera navigation, the caption describes the environment independently of the camera trajectory. It may describe scene content outside the initial view, but excludes camera motion and apparent scene motion caused by viewpoint changes. Object appearance, spatial relationships, and independently moving subjects remain useful conditions. This provides a description of the scene without duplicating the camera trajectory supplied through the action stream.

For robot and human egocentric training clips, captions must avoid revealing the consequences of future actions. Removing a sentence about the agent's motion is insufficient if the remaining description gives an object's eventual location or state. We therefore rewrite the first temporal segment as an initial-state description, retaining the setting, appearance, configuration, and existing contacts while removing controlled actions and their outcomes. An object already held at the start may be described, while its later displacement or placement is excluded. The rewrite receives neither later segments nor the video and must not infer an initial state from a removed outcome. Because the source description was generated from the full video, the rewrite is an initial-state approximation rather than a frame-level visibility annotation. Fig. 8 illustrates this rewrite on an AgiBot-GEAR benchmark example: it retains the slippers' initial location but removes their subsequent grasp and lift.

The generator receives the final caption without the intermediate structured scene breakdown. All four stages use the same final captions for camera, robot, and human videos, preserving the language condition through causal conversion and distillation. Robot benchmark captions use the training initial-state rewrite (Sec. 5.5). Behavior-1K uses separately supplied task descriptions, which are available before rollout and express the intended activity without summarizing future observations. We retain this source-native condition because the caption rewrite is calibrated for recorded robot and human egocentric video.

5.3. Post-training

Training setup. All four stages use camera, robot, and human interaction data with $2:2:1$ source weights. Each GPU processes one clip using the action-processing pipeline for its source, and gradients from all GPUs are combined before one optimizer step. The first three stages sample 256p and 480p clips at a $2:1$ ratio, whereas SF-DMD uses only 480p clips. Cosmos3-Nano-Sim-Bimanual training clips contain at most 901 RGB frames. These settings expose the same model to both viewpoint motion and object interaction throughout the curriculum, while the final stage concentrates on the higher resolution tier.

The causal stages predict 2 latent frames per chunk and retain the full available sequence history without

Table 2: **Cosmos3-Nano-Sim-Bimanual post-training settings.** Iterations identify selected stage checkpoints rather than a shared cumulative counter. RGB frame limits include the initial frame. The first three stages mix 256p/480p at $2:1$, and SF-DMD uses 480p only. SF-DMD learning rates are student / auxiliary score model. Teacher CFG lists the distillation guidance scale, with scale 1 denoting no CFG.

Stage	Checkpoint iteration	RGB frame limit	Learning rate	Resolution	Teacher CFG
Teacher	35,000	901	10^{-5}	256p / 480p	—
TF	8,000	901	10^{-5}	256p / 480p	—
TF-dCM	4,000	901	2×10^{-6}	256p / 480p	3
SF-DMD	1,380	901	10^{-6} / 2×10^{-7}	480p	1

attention sinks. This chunk size matches the final student's prediction unit, but training history is longer than the bounded K/V window used in the reported streaming benchmarks. Consequently, the curriculum and final inference configuration differ in both the source of visual history and how much of that history remains visible. Tab. 2 summarizes the selected stage checkpoints.

Training curriculum. Stage 0 initializes the bidirectional teacher from the public Cosmos3-Nano model and adapts it to the action-conditioned data before imposing causal attention (Sec. 3.1). The action projection and action-modality embedding use $5\times$ the base learning rate, allowing these action-specific parameters to adapt at a different rate from the inherited generator. The teacher provides the domain-adapted starting point for causal conversion. Its selected checkpoint is iteration 35,000.

Stage 1 applies causal teacher forcing with the shared replay procedure (Sec. 3.2 and Eq. (5)). Each prediction uses the supplied actions and earlier recorded RGB history, and the rectified flow loss supervises RGB targets rather than actions. This establishes the causal prediction problem while keeping the visual context clean. Using recorded history keeps this stage's context clean, while the subsequent SF-DMD stage addresses stability under generated history.

Stage 2 uses TF-dCM to distill the causal model to 4 denoising steps while retaining recorded history (Sec. 3.3 and Eq. (9)). The teacher uses classifier-free guidance (CFG) with scale 3. This stage reduces the denoising work within each chunk without changing the source of its preceding RGB context. It therefore addresses sampling cost separately from the mismatch between recorded training history and generated inference history.

Stage 3 applies SF-DMD so that the 4-step student learns while continuing from its own generated chunks (Sec. 3.4). Completed predictions become the visual history for subsequent chunks, exposing training to the consequences of earlier prediction errors. We alternate 1 student update with 5 auxiliary score-model updates, using a teacher with CFG disabled and only 480p training clips. The selected checkpoint at iteration 1,380 uses the combined phase-update counter, comprising 230 student updates and 1,150 auxiliary score-model updates.

5.4. Inference Optimization

Execution configuration. Cosmos3-Nano-Sim-Bimanual combines compiled denoising, CUDA-graph replay, attention bounded by a fixed K/V history window, and TAEHV Super decoding using the mechanisms in Sec. 4.2. The measured configuration runs on 1 GB300 GPU with 4 denoising steps, a history window of 64 latent frames, and chunks of 2 latent frames decoded into 8 RGB frames at 640×352 resolution. Bounding the visible history controls the attention workload as the session grows, while the lightweight decoder reduces the cost of turning completed latents into observations. These choices complement few-step sampling because clean-history refresh and decoding remain necessary for continued generation.

Measured performance. Tab. 3 compares two decoders while holding the recorded-action workload and compiled execution preset fixed. TAEHV Super achieves 48.1 FPS with median and 95th-percentile chunk times of 166 and 169 ms, compared with 35.0 FPS and 229/231 ms for Wan 2.2. The median reduction is 63

Table 3: **Cosmos3-Nano-Sim-Bimanual decoder runtime on 1 GB300.** Chunks are 8 decoded frames at 30 FPS (267 ms of physical time) from 2 latent frames with 4 denoising steps and a K/V window of 64 latent frames. Chunk wall time covers the recorded generation call, and capacity is 8 frames divided by that time. All rows use the same recorded-action workload. Counts pool steady-state chunks after 10 warm-up chunks across the listed runs. Nsight runs are excluded.

Configuration	Chunk wall time (ms)		Capacity	Chunks	Runs
	p50	p95	(FPS)		
Compiled preset, TAEHV Super	166	169	48.1	720	3
Compiled preset, Wan 2.2 decoder	229	231	35.0	240	1

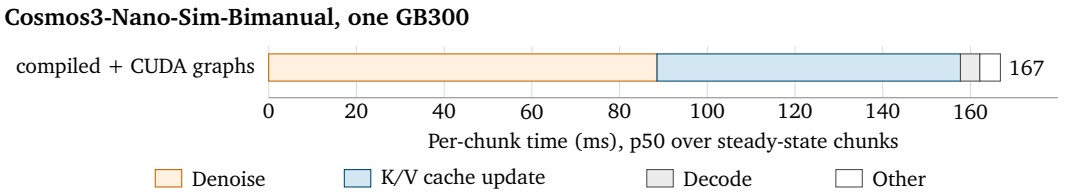


Figure 9: **Cosmos3-Nano-Sim-Bimanual compiled inference on 1 GB300.** The bar shows denoising, clean-history K/V updates, decoding, and residual time for the compiled configuration with CUDA graphs and TAEHV Super on the recorded-action workload. Stage medians come from 1 instrumented run with a median chunk time of 167 ms, separate from the pooled 166 ms result in Tab. 3. Host timers surround device synchronizations. The stacked components summarize timing attribution rather than an execution timeline.

ms, or 27.5%, and the measured capacity increases by approximately $1.37\times$. At a 30-FPS playback rate, an 8-frame chunk represents 267 ms of physical time, so both rows have steady generation capacity above the playback rate on this workload. These measurements include the recorded generation call through decoded output and exclude the first 10 warm-up chunks of each run.

The separate instrumented run in Fig. 9 attributes approximately 89 ms to denoising and 69 ms to clean-history K/V updates, compared with 4.4 ms for TAEHV Super decoding. The substantial history-update cost explains why reducing the number of denoising evaluations alone cannot determine the end-to-end speedup. This profile has a 167-ms median chunk time and is separate from the pooled 166-ms result. Component medians attribute time to the major stages, while the synchronized chunk timer provides the execution-level measurement.

The decoder comparison shows faster reconstruction, with quality measured on a separate Camera Easy workload. A separate 80-sample Camera Easy test reports a PAI-Bench-G score of 71.8 with Wan and 70.9 with TAEHV Super. The quality and runtime results therefore characterize complementary workloads rather than a matched operating point for the final student. Sustained throughput measures steady generation, while response latency additionally includes action collection, chunk scheduling, startup, and display before the first affected observation appears.

5.5. Results

Evaluation setup. We evaluate the final SF-DMD student on camera navigation and robot manipulation using an initial image, a scene caption, and supplied actions. The student uses 4 denoising steps and a 64-latent-frame K/V window with 8 latent frames retained as sinks. The benchmarks replay recorded or synthesized actions and assess the generated observations as predicted history accumulates. They therefore test action-conditioned video prediction without a policy choosing subsequent actions from the generated observations. We compare each baseline as a complete configured generator under its native inference settings.

Camera navigation benchmarks. The navigation suite contains 300 PAI-Bench-G (Zhou et al., 2026a) image-to-video cases with synthesized camera trajectories, together with 80 simple and 80 hard Sana-WM-Bench (Zhu



Figure 10: **Cosmos3-Nano-Sim-Bimanual camera navigation from a shared initial image.** The first column shows the shared initial image. Each row then shows frames at 2.5, 5, 7.5, and 10 s from one rollout conditioned on a different camera trajectory (follow behind, crane up follow, truck left, and truck right). Insets show the supplied camera trajectory and the camera pose at the displayed frame.

et al., 2026a) cases. The supplied PAI-Bench-G trajectories span 1 minute. The Sana-WM-Bench rollouts contain 961 frames at 16 FPS and span 60 seconds. Each case supplies an initial image, a scene caption independent of camera motion, and the requested trajectory. We compare with LingBot-World 2.0 (Gao et al., 2026b) and Sana-WM (Zhu et al., 2026a), which use dedicated camera-conditioning pathways. The shared interface is tested on sustained viewpoint control across all three navigation populations.

Video quality and camera metrics. Both benchmark tracks use PAI-Bench-G to assess video quality and its change over a rollout. The visual quality column reports the PAI-Bench-G overall score. Quality drift is the absolute start/end difference

$$\Delta M(V) = |M(V_{\text{start}}) - M(V_{\text{end}})|\,, \tag{27}$$

where M is the PAI-Bench-G overall score on equal-duration segments. We interpret quality drift together with the quality score, since a uniformly poor sequence can also have low drift.

Camera alignment compares ViPE-estimated poses (Huang et al., 2025a) from generated videos with supplied trajectories after Umeyama alignment (Umeyama, 1991). Absolute translation error (ATE) measures agreement along the aligned trajectory. Relative translation error (RTE) and relative rotation error (RRE) measure agreement between pose pairs. These measurements complement appearance scores by checking whether viewpoint motion follows the supplied control.

Camera navigation results. The student's quality scores exceed the strongest listed baseline by 8.55 points on PAI-Bench-G, 11.23 on Sana-WM-Bench simple, and 10.83 on Sana-WM-Bench hard (Tab. 4). On both Sana-WM-Bench sets, these gains coincide with lower quality drift and closer trajectory agreement, with an RTE tie on the hard set. Thus the quality advantage on these cases is accompanied by stronger camera following rather than obtained by ignoring the requested viewpoint motion. The same checkpoint supports these camera-conditioned rollouts alongside the robot results below.

PAI-Bench-G exposes a more qualified tradeoff. The student has lower RTE and RRE than the listed baselines, but its ATE is 4.7072 compared with LingBot-World 2.0's 4.1698, and its quality drift is 0.0210 compared with 0.0209. These results combine stronger appearance and local motion agreement with a modest ATE and

Table 4: **Camera navigation benchmark results.** The benchmark contains 300 PAI-Bench-G (Zhou et al., 2026a) image-to-video cases and 80 cases each from the simple and hard Sana-WM-Bench sets (Zhu et al., 2026a). The reported Cosmos3-Nano-Sim-Bimanual checkpoint is the final SF-DMD student, evaluated with 4 denoising steps and a K/V window of 64 latent frames with sink 8. The Video quality columns report the PAI-Bench-G overall score and its start/end quality drift. Camera alignment compares ViPE-estimated poses with supplied trajectories after Umeyama alignment. Bold and underlined values mark the best and second-best distinct values within each benchmark group, with ties sharing the same emphasis.

Model	Video quality		Camera alignment		
	Quality ↑	Drift ↓	ATE ↓	RTE ↓	RRE ↓
PAI-Bench-G I2V					
Cosmos3-Nano-Sim-Bimanual	**73.9099**	<u>0.0210</u>	<u>4.7072</u>	**0.0545**	**0.1002**
LingBot-World 2.0 (Gao et al., 2026b)	<u>65.3577</u>	**0.0209**	**4.1698**	<u>0.0596</u>	0.1781
Sana-WM (Zhu et al., 2026a)	65.2090	0.0299	6.4823	0.1284	<u>0.1601</u>
Sana-WM-Bench simple					
Cosmos3-Nano-Sim-Bimanual	**77.2887**	**0.0118**	**1.0040**	**0.0124**	**0.0551**
LingBot-World 2.0 (Gao et al., 2026b)	65.9748	0.0371	1.4254	<u>0.0185</u>	0.2591
Sana-WM (Zhu et al., 2026a)	<u>66.0599</u>	<u>0.0223</u>	<u>1.2159</u>	0.0192	<u>0.2356</u>
Sana-WM-Bench hard					
Cosmos3-Nano-Sim-Bimanual	**76.3971**	**0.0161**	**1.3085**	**0.0161**	**0.1774**
LingBot-World 2.0 (Gao et al., 2026b)	64.5462	0.0372	1.3423	**0.0161**	0.3264
Sana-WM (Zhu et al., 2026a)	<u>65.5699</u>	<u>0.0251</u>	<u>1.3165</u>	<u>0.0178</u>	<u>0.1826</u>

endpoint-drift tradeoff. The tradeoff may reflect generated geometry, trajectory following, pose estimation, or a combination of these factors.

Robot manipulation benchmarks. The robot benchmarks use validation videos from AgiBot-World and AgiBot-GEAR (Bu et al., 2025). AgiBot-World contributes 232 windows from 186 source episodes, and AgiBot-GEAR contributes 100 windows from 30 episodes. Each window contains 901 observations at 30 FPS, spanning 30 seconds of transitions under supplied robot actions. The initial image, caption, and actions define the observation-prediction problem. The DreamDojo (Gao et al., 2026a) 2B and 14B bidirectional teachers provide external comparisons with different backbones, training data, action adapters, and inference settings.

Robot manipulation metrics. Video quality and drift use the same PAI-Bench-G definitions as the navigation track. Reference-frame agreement measures whether the generated observations match recorded observations under the supplied actions. Peak signal-to-noise ratio (PSNR, dB) measures pixel agreement, structural similarity (SSIM) measures structural image agreement (Wang et al., 2004), and learned perceptual image patch similarity (LPIPS) measures perceptual feature distance (Zhang et al., 2018). These image metrics quantify action-conditioned observation agreement.

Robot manipulation results. The student improves all five reported metrics on both AgiBot benchmarks relative to the listed DreamDojo teachers (Tab. 5). Against DreamDojo 14B, PSNR increases by 2.06 dB on World and 1.90 dB on GEAR, while LPIPS decreases by approximately 3.7% and 6.2%, respectively. The agreement across quality, pixel, structural, and perceptual metrics supports more faithful observation prediction under the supplied actions, rather than an appearance gain alone. The lower quality drift also indicates a smaller difference between the beginning and end of these rollouts.

These findings extend the navigation results to observations that depend on robot and object motion. A shared checkpoint can cover both control domains without selecting a separate visual generator for each, which is the evaluated benefit of the common interface. The evaluation therefore demonstrates action-conditioned

Table 5: **Robot manipulation benchmark results.** The benchmark contains 232 AgiBot-World (Bu et al., 2025) windows from 186 source episodes and 100 AgiBot-GEAR windows from 30 source episodes. The reported Cosmos3-Nano-Sim-Bimanual checkpoint is the final SF-DMD student, evaluated with 4 denoising steps and a K/V window of 64 latent frames with sink 8. The Video quality group reports the PAI-Bench-G overall score and its start/end quality-drift diagnostic. PSNR, SSIM, and LPIPS measure reference-frame agreement under supplied actions. Bold marks the best and underlining the second-best value within each benchmark group.

Model	Video quality		Reference-frame agreement		
	Quality ↑	Drift ↓	PSNR (dB) ↑	SSIM ↑	LPIPS ↓
AgiBot-World					
Cosmos3-Nano-Sim-Bimanual	**76.0884**	**0.0118**	**14.4138**	**0.6839**	**0.3454**
DreamDojo 2B (Gao et al., 2026a)	74.5327	0.0149	12.1016	0.6405	0.3766
DreamDojo 14B (Gao et al., 2026a)	<u>74.9390</u>	<u>0.0133</u>	<u>12.3561</u>	<u>0.6635</u>	<u>0.3586</u>
AgiBot-GEAR					
Cosmos3-Nano-Sim-Bimanual	**74.4912**	**0.0188**	**14.0158**	**0.6172**	**0.4057**
DreamDojo 2B (Gao et al., 2026a)	72.4586	0.0232	11.5709	0.5437	0.4638
DreamDojo 14B (Gao et al., 2026a)	<u>73.0033</u>	<u>0.0218</u>	<u>12.1113</u>	<u>0.5818</u>	<u>0.4326</u>



Figure 11: **Robot manipulation comparison.** Rows show the reference video, DreamDojo 2B, DreamDojo 14B, and Cosmos3-Nano-Sim-Bimanual from the same initial image and robot actions. Columns show frames at 0, 10, 20, 25.9, and 28 s, progressing from the initial view through base motion, tissue grasp, motion over the bin, and the final sampled view.

observation prediction across both control domains with one visual generator.

Recipe-stage results. To examine where the final student's behavior emerges, Tab. 6 compares all four stages on the same 30-case AgiBot-GEAR subset with matched initial images, actions, and source captions. This subset is chosen from the full 100-window GEAR evaluation. The teacher and causal TF use 35 denoising steps, while TF-dCM and SF-DMD use 4, retaining each stage's native inference configuration. Causal conversion reduces the quality score from 73.385 to 72.367 and PSNR from 13.452 to 12.482 dB. TF-dCM with 4 denoising steps retains a similar quality score of 72.375, but PSNR falls further to 11.847 dB. Together, the metrics show how causal conversion and distillation affect both appearance and reference-frame agreement.

SF-DMD recovers much of this loss while retaining 4-step generation. Relative to TF-dCM, quality increases



Figure 12: **Camera navigation comparison.** The top row shows the supplied camera trajectory and the current camera pose. The remaining rows show LingBot-World 2.0, Sana-WM, and Cosmos3-Nano-Sim-Bimanual from the same initial image and trajectory. Columns show positions at 0.125, 10, 20, 40, and 59 s along the looping trajectory, including a later revisit to the building and car.

Table 6: **Cosmos3-Nano-Sim-Bimanual stage ablation on an AgiBot-GEAR subset.** All stages use the same 30 cases, initial images, actions, and source captions, separate from the full GEAR benchmark. Steps give each stage's denoising evaluations per prediction. Quality uses the PAI-Bench-G overall score, and PSNR, SSIM, and LPIPS measure reference-frame agreement. Bold and underlined values mark the best and second-best means within each metric.

Stage	Steps	Quality ↑	Reference-frame agreement		
			PSNR (dB) ↑	SSIM ↑	LPIPS ↓
Bidirectional	35	<u>73.385</u>	<u>13.452</u>	**0.591**	<u>0.454</u>
Causal TF	35	72.367	12.482	0.502	0.516
TF-dCM	4	72.375	11.847	0.525	0.512
SF-DMD	4	**74.251**	**13.479**	<u>0.589</u>	**0.447**

by 1.876 points, PSNR by 1.632 dB, and SSIM by 0.064, while LPIPS decreases by 0.065. The final means approach the bidirectional teacher's reference agreement: PSNR is 13.479 versus 13.452 dB, SSIM is 0.589 versus 0.591, and LPIPS is 0.447 versus 0.454. The quality score is also higher, at 74.251 versus 73.385. These results are consistent with the benefit of training on generated history after distillation, as in the shared recipe. Because this transition also adds training, switches to 480p-only clips, and changes other stage settings, we interpret the gain as the effect of the complete SF-DMD stage. The teacher and student have similar means on these 30 cases, and paired uncertainty estimates are not available for the small differences. Reducing 35 denoising steps to 4 lowers denoising work, while history refresh and decoding remain in the inference path.

Qualitative results. Fig. 10 shows rollouts from one initial image under four camera trajectories. The rollouts diverge according to the supplied trajectory: the camera stays behind the puppy, rises and looks down at the path and the ball, or moves laterally so that the puppy moves across the frame and the bench or railing enters the view. These selected rollouts visualize trajectory-dependent viewpoint control across the four requested motions. The robot comparison in Fig. 11 follows a sequence from base motion to grasping a tissue and moving it toward a bin. The third sampled column shows the tissue at the gripper in the reference and our rollout, and

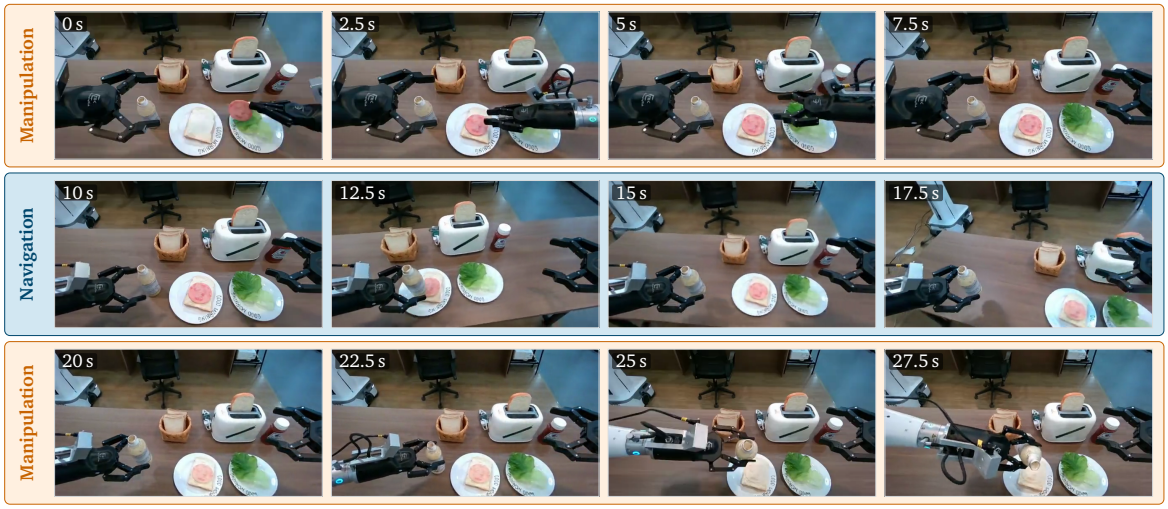


Figure 13: **Mixed manipulation and navigation rollout.** Frames from a 30-second rollout are shown from top to bottom. Here, the action sequence begins with 10 seconds of manipulation, followed by 10 seconds of camera navigation and another 10 seconds of manipulation. During navigation, the camera slowly turns right and then left before returning to its original pose. The robot arms remain stationary while the camera moves, after which manipulation resumes using the accumulated visual history.

the fourth shows the arm over the bin. The baselines change the floor and surrounding scene after the base motion, while our rollout retains more of the reference layout through the subsequent manipulation. Together, the frames illustrate visual continuity across viewpoint and object changes and complement the quantitative results above.

The looping camera example in Fig. 12 complements the forward trajectory metrics by revisiting previously visible content. At the last sampled view, our rollout retains the yellow car and wooden building, while LingBot-World 2.0 shows a changed scene and Sana-WM contains substantial artifacts. This example illustrates scene continuity during a revisit within the active history window.

Building on the manipulation and navigation capabilities demonstrated above, Fig. 13 combines both control modes in a single continuous rollout, switching from robot manipulation to camera navigation and back to robot manipulation without resetting its visual history.

Evaluation scope. The benchmarks evaluate action-conditioned observation prediction from recorded or synthesized action sequences. The camera metrics quantify trajectory following, while the reference-frame metrics quantify agreement with a recorded continuation. Contact mechanics, task success, and closed-loop policy behavior require dedicated evaluation. Because the baselines retain their native configurations and the training stages change several settings together, the comparisons characterize complete configured systems and stage recipes. The qualitative rollouts provide selected examples, and the benchmark tables provide the population-level measurements reported here.

6. Post-training Example: Cosmos3-Nano-Sim-Transfer

Closed-loop simulation provides diverse scenarios for training robot policies through repeated interaction. The simulator updates the physical state in response to policy actions, and rendered observations guide the next action. Simplified simulator scenes lack the visual richness and diversity of real environments.

Cosmos3-Nano-Sim-Transfer converts simple simulator outputs into realistic RGB observations using language and spatial controls. Depth maps, segmentation maps, edges, or blurred control videos encode scene structure, while language specifies appearance. The generator synthesizes detailed textures, materials, lighting, and

backgrounds. This combination couples simulated physical interactions with realistic visual variation for training robust, generalizable robot policies.

Autoregressive generation incorporates updated simulator controls and generated RGB history to render successive chunks as the environment changes. Sampling with 4 denoising steps reduces the denoising work per chunk. We combine these mechanisms for real-time closed-loop simulation and online robot policy learning.

6.1. Architecture

Cosmos3-Nano-Sim-Transfer uses the shared architecture (Sec. 2) to generate RGB observations from language and spatial controls supplied by an external simulator. The simulator evolves the physical state, while the generator determines the appearance of the observations. Each sample supplies 1 control modality: depth, segmentation, edges, or blur. Depth constrains geometry, segmentation constrains region identity, edges constrain contours, and blur retains coarse appearance and layout.

The reasoner (text) tower encodes the tokenized prompt into contextual language representations using causal self-attention. The attention mask allows both control and RGB queries to attend to language keys and values at every MoT layer throughout training and inference. During autoregressive inference, cached language keys and values preserve this conditioning across denoising steps and output chunks.

The generator draws on language, spatial controls, and RGB history to render each new observation. The prompt describes the scene and its desired appearance, while the controls specify the spatial structure that the observation should follow. Clean RGB history gives the model a visual reference for carrying the scene's appearance forward as the controls change. Control videos and RGB targets pass through the same causal Wan2.2 vision tokenizer and input projection. The tokenizer compresses time by 4 and each spatial dimension by 16 (NVIDIA, 2026a), so the transformer operates on compact video latents.

To render each RGB chunk, the model uses its output projection to predict a rectified flow velocity $\mathbf{v}_\theta$ (Liu et al., 2023). At normalized diffusion time τ, the sampler uses this velocity to form the clean prediction $\widehat{\mathbf{x}}_0 = \mathbf{x}_\tau - \tau\mathbf{v}_\theta$, where $\mathbf{x}_\tau$ stacks the noisy RGB latents. Control tokens guide this prediction and are excluded from the generation loss.

Fig. 14 shows how language, controls, and RGB interact through attention. The controls for a chunk describe the simulator state that the generator should render. Control queries can attend to controls through the current chunk, but their access to clean RGB history is restricted to strictly earlier chunks. This lets the model use the current simulator controls without reading the clean RGB target through its control representations during training. In the two-pass teacher-forcing procedure (Sec. 3.2 and Fig. 5), the clean pass processes latent frames causally, and the noisy pass reuses the clean control keys and values computed by that pass. The noisy queries in Fig. 14(b) also read cached language keys and values, strictly earlier clean RGB history, and noisy tokens within their own chunk.

To keep the renderer responsive to changes in the simulator, we use 1 latent frame per chunk throughout causal training and generation. Following the tokenizer's temporal compression, the first chunk corresponds to 1 RGB frame, and each subsequent chunk corresponds to 4. Using this minimum chunk size reduces buffering delay between updated controls and rendered observations during inference. In closed-loop operation, policy actions advance the simulator, which supplies new controls for the renderer. The rendered observations then inform the policy's next decision. For offline benchmarking, the complete control sequence is available in advance, allowing us to resize and encode it before autoregressive generation.

6.2. Data Preparation

Cosmos3-Nano-Sim-Transfer training pairs RGB videos with spatial controls and temporally aligned captions. All four training stages apply aspect-ratio and source-FPS filtering and validate captions before sampling aligned

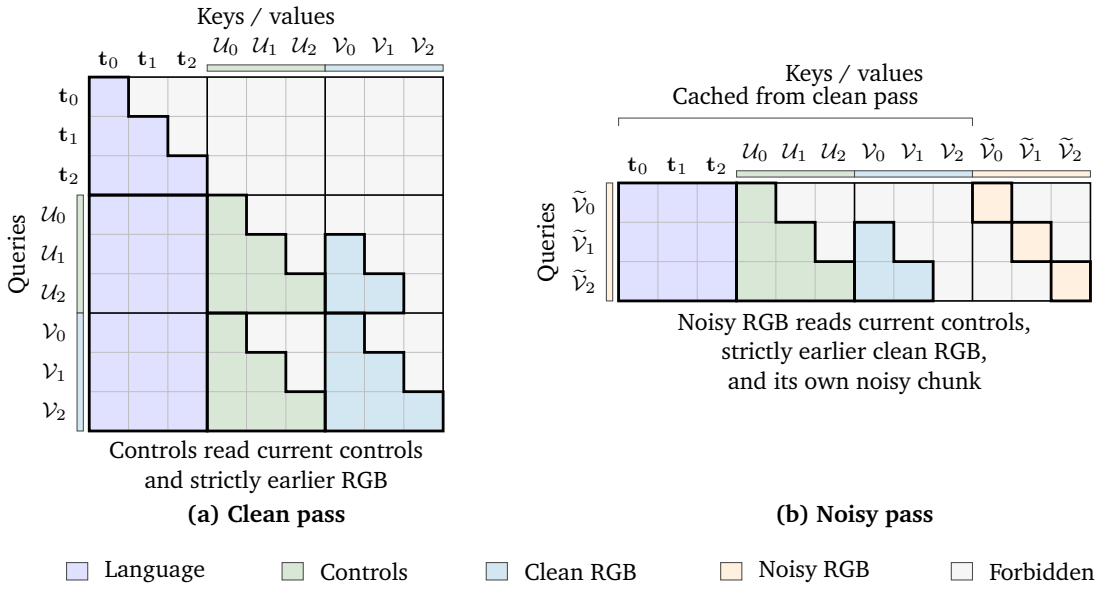


Figure 14: **Attention in Cosmos3-Nano-Sim-Transfer.** Rows are queries and columns are keys/values, using the visual conventions of Fig. 3. $\mathcal{U}_i$, $\mathcal{V}_i$, and $\widetilde{\mathcal{V}}_i$ denote spatial controls, clean RGB, and noisy RGB for latent-frame chunk i. (a) The clean pass preserves causal language attention and gives control and RGB queries access to all language tokens. Control queries read controls through the current chunk and strictly earlier clean RGB, while clean RGB queries also read their own clean chunk. (b) The noisy pass reuses the clean pass's language, control, and RGB keys and values. Each noisy RGB query reads controls through its current chunk, strictly earlier clean RGB, and noisy RGB within its own chunk. Neither controls nor noisy RGB can read the current clean RGB target. Each control or RGB chunk contains 1 latent frame and occupies one row or column, with spatial tokens aggregated into a single block. Three chunks are shown, with the optional conditioning prefix omitted.

RGB and control clips. Our source dataset contains 2,153,548 RGB video entries. The Stage 3 index retains 1,996,602 videos with complete eligible caption windows.

Spatial and temporal alignment. RGB and controls use identical temporal indices and target geometry. We replace reflection padding with direct resizing for RGB and control videos to prevent mirrored artifacts from extending beyond the padded regions in generated outputs. To limit stretching when source and target aspect ratios differ, we reject samples requiring more than 3% aspect-ratio distortion. We use bicubic antialiasing for RGB, bilinear interpolation for depth, and nearest-neighbor interpolation for segmentation. Edge and blur controls are computed from the resized RGB.

Depth controls are precomputed with Video Depth Anything and normalized over each video (Chen et al., 2025). Segmentation controls use colorized SAM2 masks (Ravi et al., 2025). Edge controls use Canny detection with randomized thresholds, while blur controls sample bilateral filtering, Gaussian filtering, or their composition. The index checks RGB/control geometry and frame-rate compatibility and restricts caption windows to their common available frame range. These checks establish metadata and caption eligibility.

Captioning. We generate training captions with a Qwen3-VL-8B (Bai et al., 2025) captioner adapted using LoRA (Hu et al., 2022). To associate each description with a specific interval in the video, the captioner processes nonoverlapping windows of up to 200 source frames. Each eligible annotation records the window's start and end frames alongside a structured JSON description, including a nonempty `temporal_caption` field that describes how the scene unfolds over time. The loader selects the annotation together with the corresponding RGB and control window, keeping the language and visual inputs tied to the same interval.

Temporal sampling and chunking. Across all four training stages, we use identical frame indices for RGB and controls so that each control frame stays aligned with its RGB target. Each training clip contains 1 initial RGB frame followed by complete groups of 4 frames. The causal tokenizer encodes the first RGB frame as 1 latent frame and each subsequent group of 4 as another.

In Stages 0–2, we randomly select a caption window and sample every frame, every second frame, or every third frame within that window. We then trim the sampled clip to satisfy this tokenizer alignment. The clip-length ranges depend on the training stage and resolution. Stage 0 uses 5–201 RGB frames at the 480 resolution tier and 5–149 at the 720 tier. Stages 1 and 2 use 9–201 RGB frames at the 480 tier and divide these clips into chunks of 1 latent frame, matching the chunk size used for autoregressive generation. In Stage 3, all video clips within a training batch have the same number of frames.

6.3. Post-training

Training setup. All four training stages use the same transfer-video collection, pairing each RGB target with 1 spatial-control stream. We sample depth, segmentation, edges, and blur with equal weights so that training covers each way of specifying the desired scene structure. Stage 0 mixes the 480 and 720 resolution tiers with weights 2:1, while Stages 1–3 use only the 480 tier. Throughout the causal stages, the model predicts RGB in chunks of 1 latent frame, matching the chunk size used during autoregressive generation. Tab. 7 summarizes the selected checkpoints, clip-length ranges, learning rates, and teacher guidance for each stage.

Training examples can also include a short recorded RGB prefix that establishes the scene's initial appearance. We sample prefix lengths of 0, 1, or 2 latent frames with probabilities 0.7, 0.2, and 0.1, exposing the model to generation both with and without initial RGB observations. For short clips, we reduce the sampled prefix as needed to leave at least 1 latent frame for the model to generate.

We vary how the accompanying language is presented while keeping the selected RGB/control window fixed. The loader uses the full structured JSON with probability 0.93 and only the dense temporal caption with probability 0.07, giving the model experience with both input formats. In Stages 0–2, we additionally drop the text on 10% of samples so that the model learns to generate from visual conditions alone. Stage 3 retains language conditioning on every sample, with text dropout set to 0. Spatial controls remain present in every stage, with control dropout also set to 0.

The stages share a common optimization setup: fused AdamW with $\epsilon = 10^{-8}$, BF16 computation, FP32 master parameters, and no gradient accumulation. We adjust the momentum and weight decay for each training objective. Stage 0 uses $(\beta_1, \beta_2) = (0.9, 0.99)$ with weight decay 0.05. Stage 1 uses $(0.9, 0.999)$, while Stage 2 and both Stage 3 optimizers use $(0, 0.999)$. All three causal stages use weight decay 0.01. We maintain power-function exponential moving average (EMA) weights with relative width 0.1 in all stages.

Stage 0 packs variable-length clips into a budget of 74,000 tokens per GPU. The causal stages instead use 1 clip per GPU. Across the recorded runs and restarts, this gives global batches of 248 or 512 clips for Stage 1, 512 for Stage 2, and 248 for Stage 3.

Training curriculum. The curriculum first adapts a bidirectional teacher to the transfer data, then converts it into a causal generator and distills it to use fewer denoising steps. The final stage trains this generator on its own outputs, preparing it to continue from the history it will encounter during inference. Stages 2 and 3 both use teacher guidance 3 during distillation.

Stage 0: bidirectional adaptation. Before introducing causal attention, we adapt the bidirectional teacher following Sec. 3.1. We fine-tune it on directly resized RGB videos and aligned spatial controls so that the teacher uses the same image geometry as the later stages. This preprocessing also avoids the mirrored content introduced by reflection padding. The teacher learns to denoise recorded RGB clips by predicting rectified flow velocity from noisy RGB latents, language, and clean spatial controls. At this stage, bidirectional temporal

attention allows each RGB target to use noisy RGB tokens and supplied controls from both earlier and later frames within the clip. The loss applies only to noisy RGB targets, excluding the spatial controls and any clean RGB prefix. We use the adapted teacher at iteration 3,806 to initialize Stage 1.

Stage 1: causal teacher forcing (TF). With the adapted teacher as its starting point, Stage 1 learns to predict each RGB chunk from the available controls and earlier recorded RGB history. We use the two-pass teacher-forcing procedure in Sec. 3.2 and Fig. 5. The clean pass processes recorded RGB tokens and supplied controls to build a causal cache of attention keys and values (K/V). The noisy pass then reuses this cache to predict each chunk from its noisy RGB tokens, strictly earlier clean RGB history, controls through the current chunk, and language conditioning. The rectified flow velocity loss in Eq. (5) supervises only RGB targets, with gradients flowing through the clean K/V cache as well as the noisy pass. The resulting iteration-6,000 EMA checkpoint provides both the initial student and the frozen causal teacher for Stage 2.

Stage 2: consistency distillation (TF-dCM). Once causal prediction is established, Stage 2 reduces the number of denoising steps needed to generate each chunk through TF-dCM (Sec. 3.3 and Eq. (9)). Training continues to use recorded RGB history and the same causal control attention as Stage 1. The frozen causal teacher connects two noisy versions of a chunk at different diffusion times along its denoising trajectory. The student predicts the clean chunk from both versions and learns to make these predictions agree, using identical history, controls, and language in both evaluations. The current student's prediction at the lower-noise input supplies the training target and is computed with gradients stopped. This consistency objective prepares the student to recover a clean chunk with fewer denoising evaluations. The iteration-2,250 EMA checkpoint initializes Stage 3.

Stage 3: self-forcing (SF-DMD). Stage 3 brings training closer to autoregressive inference by letting the student learn from its own generated history, following SF-DMD in Sec. 3.4. After an optional recorded RGB prefix, the student generates successive chunks of 1 latent frame using the aligned controls and the RGB history it has produced so far. Each completed chunk and its supplied controls refresh the clean K/V cache, providing the context for the next prediction. The student therefore trains on the consequences of its earlier outputs throughout the rollout.

To train from these generated clips, we initialize two bidirectional score models from the Stage 0 teacher: a frozen reference teacher and a trainable fake-score network. We add noise to the generated clips and ask both models to denoise them. The teacher supplies the reference prediction, while the fake-score network learns to denoise the student's current outputs. The difference between their predictions defines the distribution-matching target that guides the student (Eq. (19)). We alternate 1 student update with both score models fixed and 5 fake-score updates with the student fixed. Checkpoint iteration counts include both types of updates.

We use replayed backpropagation to train on a complete rollout without retaining all of its intermediate activations. Training first constructs the rollout, then recomputes each chunk's final denoising prediction from its saved noisy input, timestep, and detached cache prefix. Gradients pass through these recomputed predictions but do not propagate through earlier denoising steps or historical K/V caches. This trains the model to use generated history while treating that history as a fixed input for each gradient computation.

6.4. Inference Optimization

Inference. Cosmos3-Nano-Sim-Transfer follows the autoregressive inference loop in Sec. 4.1. For each chunk, the sampler denoises 1 RGB latent frame using aligned spatial controls, cached language keys and values, and retained clean control and RGB history. The controls and retained history remain fixed across denoising steps. After sampling, the runtime refreshes clean control and RGB keys and values for the next prediction. The streaming Wan2.2 VAE (Wan et al., 2025) decodes the completed latent chunk into RGB observations, carrying temporal state across chunks so that output is available before the entire video is generated.

The history policy determines which control and RGB K/V entries remain available to subsequent chunks.

Table 7: **Cosmos3-Nano-Sim-Transfer post-training settings.** Iterations identify the selected EMA checkpoints. RGB ranges are configured sampling limits. Stage 0 multiplies its base learning rate by $0.4(1 - i/100{,}000)$ at iteration i, starting at 4×10^{-5} without warmup. The other learning rates are constant. The SF learning rates are listed as student / fake score.

Stage	Checkpoint iteration	RGB frames	Base learning rate	Teacher CFG
Teacher	3,806	5–201 (480p), 5–149 (720p)	10^{-4}	—
TF	6,000	9–201	10^{-5}	—
TF-dCM	2,250	9–201	2×10^{-6}	3
SF-DMD	1,500	9–201	10^{-6} / 2×10^{-7}	3

Full-history inference retains all entries throughout the rollout. Bounded inference retains initial sink entries and a rolling window of recent history, replacing older entries outside this window while preserving token roles and temporal positions. This bounds history storage and attention context during extended generation.

Compilation with preserved numerical behavior. We combine `torch.compile` with block CUDA Graphs to reduce execution overhead in the single-view SF-DMD student, following the shared runtime design in Sec. 4.2. The optimized path keeps normalization outside compilation and preserves the reduction order for squared activations. It also preserves intermediate BF16 rounding in FP32–BF16–FP32 conversions and the original K/V cache order. These choices retain the eager path's numerical behavior at operations that can affect autoregressive outputs. Sec. 6.5 evaluates the runtime and output quality of these inference choices.

6.5. Results

Evaluation setup. We evaluate Cosmos3-Nano-Sim-Transfer from three complementary perspectives: the quality and control fidelity of its videos, its behavior as an observation renderer in a robot policy loop, and the runtime cost of sustained generation. We begin with TransferBench (NVIDIA, 2025b), which contains 600 source videos: 200 each from AgiBot (Bu et al., 2025), Ego-Exo4D (Grauman et al., 2024), and OpenDV (Yang et al., 2024a). For every source video, we generate an output conditioned on each of depth, segmentation, edges, and blur. This produces 2,400 outputs per model, each with 121 aligned RGB frames. Outputs are 736×544 for AgiBot and 832×480 for the other domains, preserving source rates of approximately 23.98–60 FPS.

To show how the final model compares with both its reference and the stages that lead to it, we evaluate five checkpoints on the same full benchmark. The iteration-1,500 SF-DMD EMA student uses 4 SDE steps, full-history K/V without attention sinks, and student/control guidance 1. For this offline benchmark, the complete control sequence is prepared and encoded before autoregressive generation. We compare it with the Cosmos 3 GA iteration-6,000 EMA reference and the selected curriculum checkpoints: the bidirectional teacher at iteration 3,806, TF at iteration 6,000, and TF-dCM at iteration 2,250. The bidirectional teacher and TF use 50-step UniPC with guidance 3, while TF-dCM and SF-DMD use 4-step SDE sampling with guidance 1. Generation uses upsampled captions, while language scoring uses the original benchmark captions. The comparison therefore evaluates complete generation pipelines with their respective samplers and guidance settings.

Video quality and temporal metrics. We assess both the appearance of the generated videos and whether that quality changes over time. DOVER measures technical video quality across all 2,400 outputs (Wu et al., 2023). For a complementary assessment, Gemini 3.1 Pro Preview receives each generated video and its original benchmark caption and assigns integer grades from 0 to 10. Its Quality grade evaluates visible artifacts, while its Prompt grade evaluates whether the elements described in the caption appear in the video.

To measure changes between the beginning and end of a video, we report quality drift as the absolute difference

between the mean perceptual scores of its first and last 15% of frames. We evaluate each endpoint with PAI-Bench-G, the video-generation component of PAI-Bench (Zhou et al., 2026a). The endpoint score averages six perceptual-quality dimensions: aesthetic quality, background consistency, imaging quality, motion smoothness, overall consistency, and subject consistency. We divide imaging-quality scores by 100 before averaging the dimensions, then average the absolute endpoint differences across videos. Lower drift indicates a smaller change in these perceptual scores between the beginning and end of the video.

Control, language, and geometry metrics. We next measure how closely the generated scene follows the supplied spatial structure, using a separate metric for each control modality and its corresponding 600 outputs. Segmentation IoU measures agreement between regions, and edge F1 measures agreement between contours. Blur SSIM measures structural similarity (Wang et al., 2004), while depth SI-RMSE measures scale-aligned depth error. The Prompt grade described above complements these spatial measurements by assessing agreement with the language condition.

To examine temporal geometry, we measure Sampson error using SuperPoint features matched with LightGlue between all frame pairs separated by approximately 1 second (Sampson, 1982; Lindenberger et al., 2023). For each pair, the evaluator estimates a fundamental matrix and retains the half of inlier correspondences with the lowest Sampson errors. It then scales the square root of their mean Sampson error to a 960×540 reference diagonal. The video score averages the valid pair scores and requires at least 90% pair coverage. This metric checks the geometric consistency of correspondences across time, adding a diagnostic that appearance scores alone do not provide. All five checkpoints have 2,400 valid scores for each video-level metric and 600 for each control metric, with no invalid values in the final records.

Benchmark results. Relative to the Cosmos 3 GA reference, Cosmos3-Nano-Sim-Transfer improves all four spatial-control metrics in Tab. 8. Segmentation IoU increases from 0.6433 to 0.7378 and blur SSIM from 0.8559 to 0.8867. Edge F1 rises from 0.2140 to 0.3407, a 59.2% increase, while depth SI-RMSE falls from 0.8423 to 0.6194, a 26.5% reduction. These results show that the final student follows the supplied regions, contours, coarse image structure, and depth more closely across the four evaluated control modalities.

Alongside its stronger spatial control, Cosmos3-Nano-Sim-Transfer slightly improves the VLM Quality grade from 4.9296 to 4.9508. DOVER and Prompt decrease slightly by 1.8% and 2.2%, respectively, while quality drift and Sampson error increase relative to the GA reference.

The intermediate checkpoints show how the final student relates to the preceding training stages. SF-DMD improves all nine reported metrics relative to TF-dCM. The Quality grade increases from 4.2588 to 4.9508, and drift decreases from 0.011730 to 0.006504. The bidirectional teacher retains stronger scores across all nine metrics than the final causal student. The final self-forcing stage thus improves on the selected consistency-distilled checkpoint, while a gap to the bidirectional teacher remains. As in the reference comparison, these results retain each checkpoint's deployment sampler and guidance settings.

We also examine uncertainty in the smaller differences between TF and TF-dCM using a paired bootstrap over source videos. We draw 10,000 resamples stratified by source domain and keep each source's four control modalities together. For TF-dCM relative to TF, the edge-F1 difference is $+0.0078$ with a 95% interval of $[0.0061, 0.0095]$, and the DOVER difference is -0.1708 with an interval of $[-0.2369, -0.1067]$. The Quality difference is $+0.0388$, with an interval of $[-0.0396, 0.1188]$. The edge-F1 and DOVER intervals preserve the direction of their observed changes, whereas the Quality interval includes zero. These per-metric intervals quantify uncertainty from sampling source videos while holding the recorded generations fixed.

Closed-loop evaluation in Isaac Lab. Cosmos3-Nano-Sim-Transfer increases apple pick-and-place completion for a Unitree G1 humanoid from 68.8% with native Isaac RTX rendering (NVIDIA, 2026f) to 84.4% on the calibrated baseline (Tab. 9). Both configurations use Isaac Lab 3.0.0b2 (Mittal et al., 2025) and Isaac Sim 6.0.0.1 (NVIDIA, 2026e) for physics. We evaluate 32 matched starts with the same 15,000-step GR00T N1.7

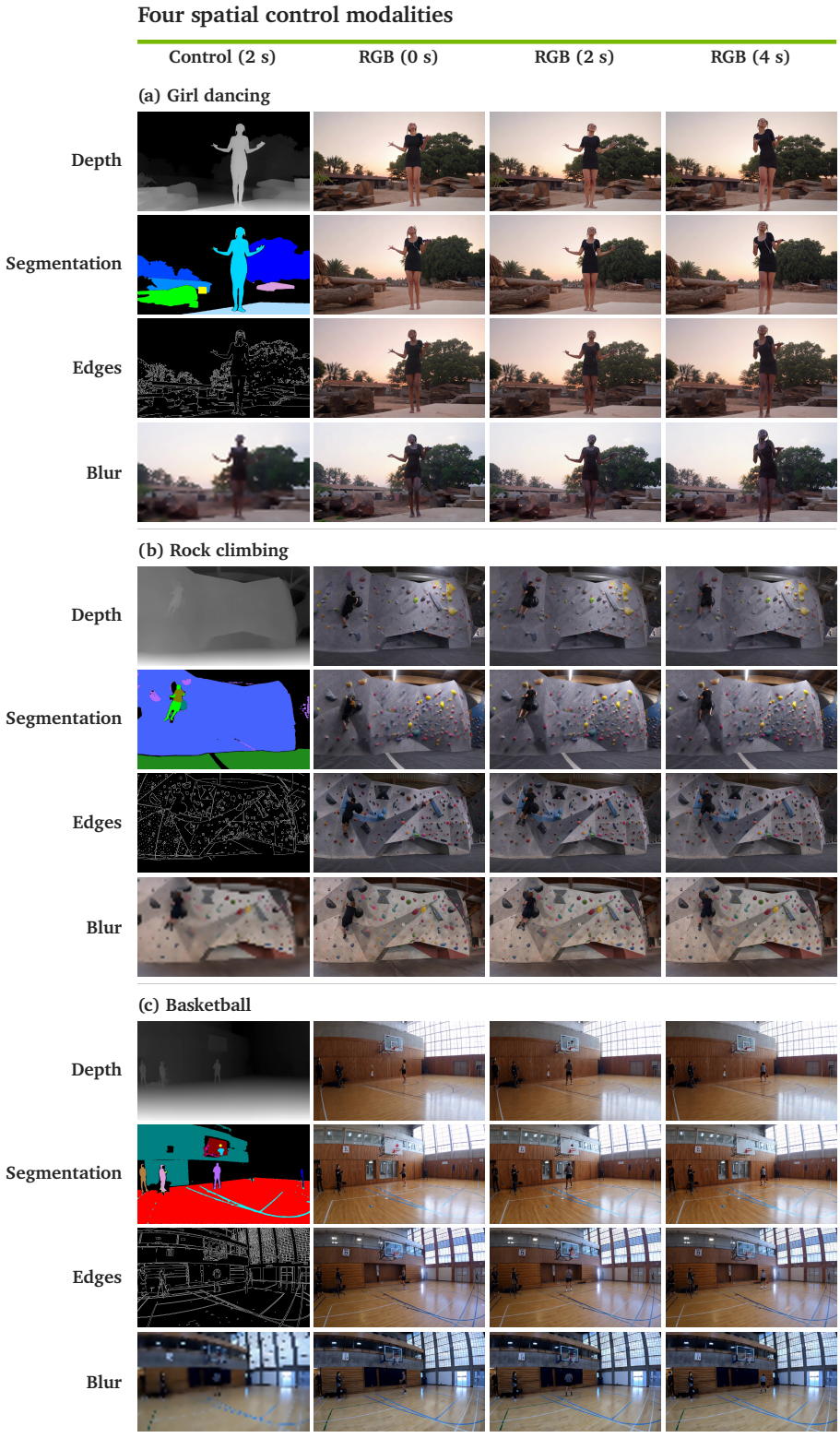


Figure 15: **RGB generation from four spatial control modalities.** From top to bottom: girl dancing, rock climbing, and basketball, each with depth, segmentation, edges, and blur. The first column shows each supplied control at 2 seconds, while the remaining columns show generated RGB at 0, 2, and 4 seconds. Each output contains 121 frames at 832×480 and 30 FPS. Generation uses 4 denoising evaluations per latent-frame chunk and full K/V history without attention sinks.

Table 8: **Full TransferBench video quality, spatial control, and condition agreement.** Each checkpoint generates 2,400 videos from 600 TransferBench sources (NVIDIA, 2025b) and four control modalities. DOVER uses all outputs, while each control metric uses its corresponding 600 outputs. Quality and Prompt retain the recorded VLM grading scales. Drift compares the first and last 15% of frames, and Sampson error uses normalized epipolar residuals at approximately 1-second intervals. Stage comparisons retain each checkpoint's deployment sampler and guidance. Bold and underlined entries mark the best and second-best values within each metric.

(a) Video quality

Model	DOVER ↑	Quality ↑	Drift ↓
Bidirectional teacher (3,806)	**10.3955**	**5.0042**	<u>0.006370</u>
TF (6,000)	10.2112	4.2200	0.008265
TF-dCM (2,250)	10.0403	4.2588	0.011730
Cosmos3-Nano-Sim-Transfer	10.1759	<u>4.9508</u>	0.006504
Cosmos 3 GA reference	<u>10.3596</u>	4.9296	**0.006349**

(b) Spatial control

Model	Segm. IoU ↑	Blur SSIM ↑	Edge F1 ↑	Depth SI-RMSE ↓
Bidirectional teacher (3,806)	**0.7465**	**0.9074**	**0.4692**	**0.5279**
TF (6,000)	0.6948	0.8591	0.2973	0.7594
TF-dCM (2,250)	0.7070	0.8569	0.3051	0.7273
Cosmos3-Nano-Sim-Transfer	<u>0.7378</u>	<u>0.8867</u>	<u>0.3407</u>	<u>0.6194</u>
Cosmos 3 GA reference	0.6433	0.8559	0.2140	0.8423

(c) Language and temporal geometry

Model	Prompt ↑	Sampson ↓
Bidirectional teacher (3,806)	<u>7.1842</u>	**0.080947**
TF (6,000)	6.7625	0.087848
TF-dCM (2,250)	6.7288	0.098861
Cosmos3-Nano-Sim-Transfer	7.0254	0.096544
Cosmos 3 GA reference	**7.1846**	<u>0.081045</u>

policy (NVIDIA, 2026d) fine-tuned on 402 real robot demonstrations. Cosmos3-Nano-Sim-Transfer converts the simulator's depth controls into RGB observations that guide the policy's next actions. At each decision, the policy receives a 640×480 head-camera RGB image, the current 43-dimensional joint state, and the task instruction, then predicts 16 actions. The simulator runs policy actions at 30 Hz and physics at 240 Hz.

As the actions advance the simulated scene, depth is sampled every 4 actions, corresponding to 7.5 Hz in simulated time. After a 1-frame bootstrap, the renderer uses these controls to generate 4 RGB observations per update. The final image becomes the input to the next policy decision, closing the loop between action, simulation, and rendering. This schedule gives a policy decision rate of 1.875 Hz in simulated time.

The renderer uses the iteration-1,500 SF-DMD EMA student with 4 SDE steps, guidance 1, a 30-latent-frame K/V window, and 3 attention-sink frames. A scene prompt fixes the requested appearance before the rollout, while depth from the simulator describes how the geometry changes as the robot acts. We normalize depth over 0.15–0.90 m, resize it from 640×480 to 736×552, and center-crop it to 736×544. We then resize generated RGB back to the policy's camera resolution. This streaming preprocessing differs from the direct resizing used in the offline benchmark. The model's FPS condition is 12.5, distinct from the simulator's observation cadence.

Each episode runs until the task terminates or reaches 600 actions, corresponding to 20 simulated seconds. Success is determined from simulator state using placement, contact, support, settling, and hand-release condi-

Table 9: **Cosmos3-Nano-Sim-Transfer improves task completion on the calibrated Isaac Lab baseline.** Entries are completed episodes out of 32 matched starts per condition. Both configurations use Isaac Lab for physics. The same GR00T policy receives either native Isaac RTX RGB or Cosmos3-Nano-Sim-Transfer RGB. Panel (a) fixes the policy at 15,000 training steps. Panel (b) fixes the baseline appearance and reports success-rate differences (Cosmos3-Nano-Sim-Transfer minus RTX) in percentage points (pp). The scene prompt was calibrated at 15,000 steps on these starts, so the table reports calibration-set outcomes.

(a) Appearance changes

Appearance	Isaac RTX	Cosmos3-Nano-Sim-Transfer
Calibrated baseline	22/32	27/32
Green apple	20/32	24/32
Red plate	22/32	19/32
Green apple, red plate, white cloth	16/32	17/32

(b) Policy checkpoints

Policy training steps	Isaac RTX	Cosmos3-Nano-Sim-Transfer	Difference (pp)
5,000	17/32	20/32	$+9.4$
10,000	16/32	12/32	-12.5
15,000	22/32	27/32	$+15.6$
20,000	24/32	21/32	-9.4
30,000	18/32	18/32	0.0
40,000	17/32	19/32	$+6.3$

tions. For every appearance condition, we evaluate 32 matched initial states in 2 batches of 16 environments. Each matched comparison holds the policy checkpoint, initial states, and simulator physics fixed, changing the policy's RGB input between native Isaac RTX and Cosmos3-Nano-Sim-Transfer.

With the 15,000-step policy, Cosmos3-Nano-Sim-Transfer achieves higher completion than native Isaac RTX in 3 of the 4 tested appearance conditions. The calibrated baseline improves from 22/32 to 27/32 completed episodes, a gain of 15.6 percentage points. The green-apple condition improves from 20/32 to 24/32, and the combined green-apple, red-plate, and white-cloth condition improves from 16/32 to 17/32. The red-plate condition yields 19/32 completions with Cosmos3-Nano-Sim-Transfer and 22/32 with RTX. Across the six evaluated policy checkpoints, the highest observed completion is 27/32 (84.4%) with Cosmos3-Nano-Sim-Transfer at 15,000 steps, compared with 24/32 (75.0%) with RTX at 20,000 steps.

Qualitative results. We first illustrate control-to-RGB generation with depth, segmentation, edges, and blur (Fig. 15). Each of the three examples uses the same source video and prompt across its four control modalities. The control images and generated frames show how each modality supplies scene structure for the model to render in RGB. We then examine rendering over longer simulated trajectories. Fig. 16 shows three 600-frame VisDex rollouts under depth and segmentation, with frames sampled from the beginning, middle, and end of each generated sequence.

Long-rollout runtime. We profile the iteration-1,500 SF-DMD EMA student on all 160 VisDex s42 episodes under four control modalities, generating 640 videos of 600 frames at 640×640. Inference uses BF16 eager execution on GB200 without compilation or CUDA Graphs, with 4 SDE steps, guidance 1, a 30-latent-frame K/V window, and 3 sink frames. For steady-state measurements, we exclude each worker's first video and the first 117 frames of each subsequent SF-DMD video, retaining 144 videos and 17,424 decoded-output chunk intervals per control. Steady throughput ranges from 9.90 to 9.96 RGB frames per second per GPU (Tab. 10). Chunk timings measure completed RGB output, including callback waits for copies to host memory. On the matched depth and segmentation workload, the Cosmos 3 GA reference takes 217.17 s on average to complete a 600-frame

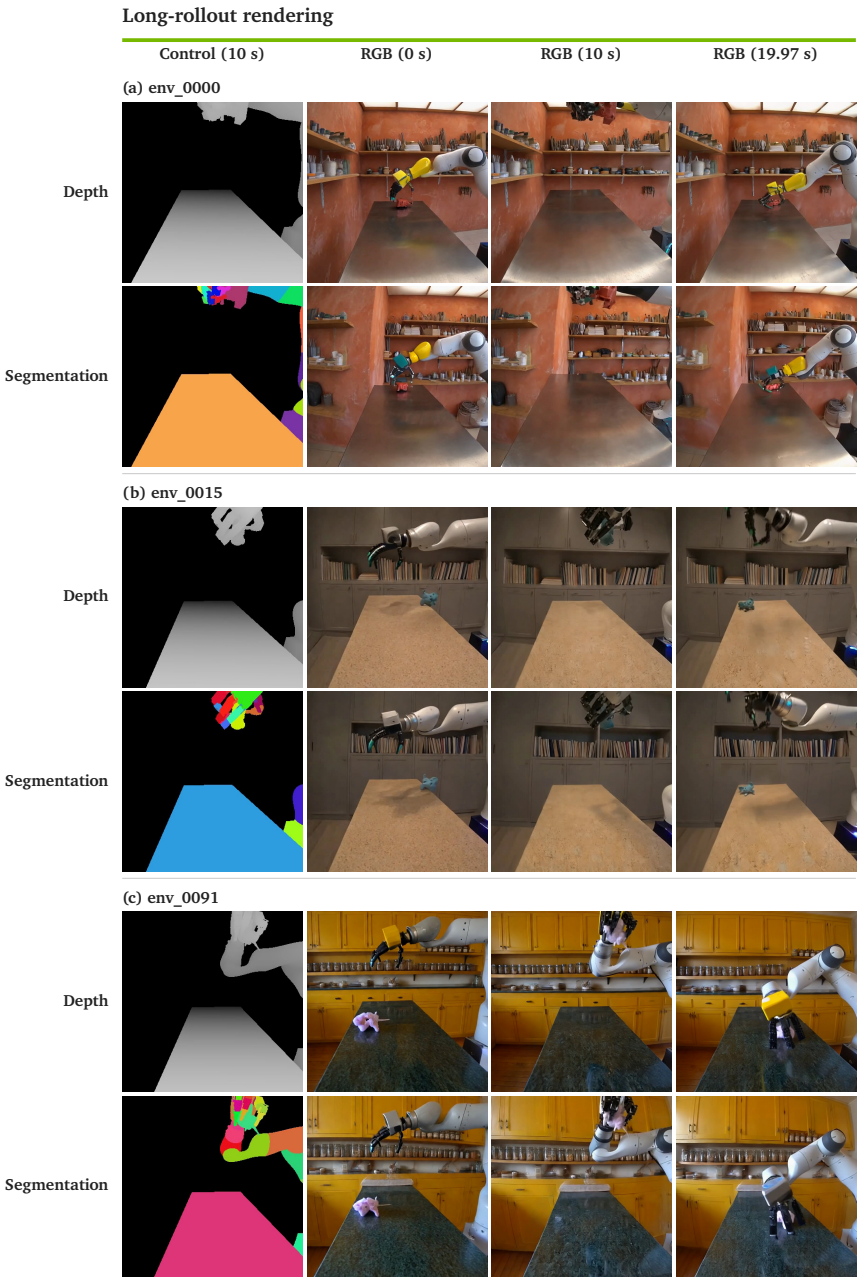


Figure 16: **Long-rollout RGB generation from depth and segmentation.** From top to bottom: VisDex episodes `env_0000`, `env_0015`, and `env_0091`, each sharing its text prompt across both control modalities. The first column shows the supplied control at 10 seconds, while the remaining columns show generated RGB at 0, 10, and 19.97 seconds. The selected SF-DMD student uses 4 denoising evaluations per latent-frame chunk, a 30-latent-frame K/V window, and 3 sink frames. Each output contains 600 frames at 640×640 and 30 FPS.

video. Its approximately 120-frame continuation blocks average 43.30 s after warmup, precluding real-time feedback in this configuration (Tab. 11). Cosmos3-Nano-Sim-Transfer generates observations autoregressively, returning a decoded latent-frame chunk, normally 4 RGB frames, in 0.403 s on average after warmup and providing sub-second observation updates for closed-loop simulation without waiting for a complete video.

Compilation and cache optimization. A separate 597-frame depth-control study evaluates the compilation and cache choices in Sec. 6.4. The iteration-1,500 SF-DMD checkpoint generates 640×640 videos on 1 NVIDIA

Table 10: **Final-student runtime on 600-frame VisDex rollouts.** All measurements use 1 stream per GB200 GPU and 144 warmed videos per control. Chunk times measure completed decoded output, normally 4 RGB frames per chunk. First-chunk times use warmed workers and exclude model loading.

Control	FPS/GPU	Chunk p50 / p95 (ms)	First chunk p50 / p95 (s)
Depth	9.928	401 / 418	6.93 / 7.23
Segmentation	9.896	403 / 420	6.55 / 6.64
Edge	9.920	401 / 418	6.33 / 6.42
Blur	9.957	399 / 421	6.26 / 6.37

Table 11: **Matched 600-frame VisDex runtime.** Both models use the same 160-episode depth and segmentation workload on GB200 with BF16 eager inference. Runtime excludes each worker's first video and retains 144 videos per control. Steady FPS and chunk latency additionally exclude the first 121 frames per remaining GA video or 117 frames per SF-DMD video. Runtime values pool depth and segmentation. Generation latency is the mean per steady continuation block, approximately 120 RGB frames, for Cosmos 3 GA and per latent-frame chunk, normally 4 RGB frames, for Cosmos3-Nano-Sim-Transfer. The means cover 1,152 GA blocks and 34,848 Cosmos3-Nano-Sim-Transfer chunks. Timing measures completed decoded output, including callback waits for copies to host memory. Model loading, source-video decoding, and file saves are excluded.

Metric	Cosmos 3 GA reference	Cosmos3-Nano-Sim-Transfer
Steady FPS/GPU ↑	2.766	9.912
Generation latency	43.30 s/block	0.403 s/chunk
Steady peak allocated memory (GiB) ↓	36.15	39.58

GB200 per worker. Each video has a nominal rate of 30 FPS, using 4 denoising evaluations per latent frame, guidance 1, and no RGB prefix. All configurations use BF16 model weights and K/V caches. The first latent frame decodes to 1 RGB frame, and each subsequent latent frame decodes to 4 RGB frames. We compare full history without sinks with a bounded window of 51 latent positions: 1 sink, 49 recent past positions, and the current position. The bounded window contains 102 paired control/RGB cache slots. The complete control video is prepared before generation.

Compilation and block CUDA Graphs increase inference throughput by up to $1.75\times$ while preserving generation quality (Tab. 12). Measured throughput increases from 6.8802 to 11.4068 FPS with full history ($1.66\times$), and from 9.0403 to 15.8516 FPS with bounded history ($1.75\times$). The corresponding p95 decoded chunk intervals decrease from 761.7341 to 443.8231 ms and from 509.8658 to 268.7143 ms.

Quality evaluation uses 16 depth-control VisDex sources per configuration, with 16/16 successful generations and no failures. Within each history policy, all 16 eager/compiled video pairs have identical decoded frames, covering 32 pairs and 19,104 frames in total. DOVER (Wu et al., 2023) and the recorded VLM prompt and quality grades remain unchanged between eager and compiled execution within each history policy (Tab. 12).

Component p50/p95 timings and decoded-output intervals are measured separately from offline preparation and startup. The timing protocol defines the measurement boundaries and overlapping measurements.

Sub-second RGB updates support the renderer's role in real-time closed-loop simulation: the simulator advances physical state and supplies new controls, and Cosmos3-Nano-Sim-Transfer renders observations for the policy's next decision.

Table 12: **BF16 inference optimization for Cosmos3-Nano-Sim-Transfer on 1 GB200.** The SF-DMD iteration-1,500 checkpoint generates 597-frame, 640×640 depth-conditioned videos. Compiled denotes `torch.compile` with block CUDA Graphs. Panel (a) measures decoded CPU output over latent indices 51–149 after warmup, excluding startup, offline condition preparation, and file encoding. Each configuration contributes 594/594 valid/attempted chunk intervals from 6 measured videos. Memory peaks are the maxima of PyTorch allocated/reserved storage across those videos. Panel (b) reports means over 16 depth-control sources per configuration, with the recorded DOVER and VLM scales.

(a) Throughput, decoded chunk interval, and GPU storage

History / execution	FPS↑	Chunk p50 / p95 (ms)	Peak allocated / reserved (GiB)
Full / eager	6.8802	560.8695 / 761.7341	55.0640 / 60.0859
Full / compiled	11.4068	348.1926 / 443.8231	74.5866 / 79.7207
Sink + recent / eager	9.0403	435.8790 / 509.8658	44.0980 / 47.7598
Sink + recent / compiled	15.8516	240.4241 / 268.7143	52.7295 / 57.8184

(b) Video quality and language following

History / execution	DOVER↑	VLM prompt /10↑	VLM quality /10↑
Full / eager	8.6524	5.3125	2.4375
Full / compiled	8.6524	5.3125	2.4375
Sink + recent / eager	9.0190	5.3750	2.5000
Sink + recent / compiled	9.0190	5.3750	2.5000

7. Post-training Example: Cosmos3-Nano-Sim-Auto

Autonomous-driving simulation requires camera and LiDAR observations that describe the same evolving scene consistently across viewpoints, modalities, and time, while following the supplied road geometry and actor motion. We extend the single-view Cosmos3-Nano to this setting through camera-specific conditioning and joint RGB–LiDAR modeling (Fig. 17). Per-view prompts preserve the foundation model's association between a caption and its video, while attention across cameras and modalities supports scene consistency. The resulting Cosmos3-Nano-Transfer-Auto denoises complete sensor clips bidirectionally for offline generation. Its causal, distilled counterpart, Cosmos3-Nano-Sim-Auto, generates synchronized sensor streams incrementally for streaming simulation.

Camera generation is conditioned on per-view captions and camera-aligned World Scenario Map (WSM) controls, while LiDAR generation uses HD-map controls. Transfer-T2V (Text-to-Video) uses no initial RGB observations, whereas Transfer-I2V (Image-to-Video) additionally uses initial camera images. An initial measured LiDAR sweep is optional in either mode.

Training begins with Stage 0, which adapts Cosmos3-Nano into the bidirectional **Cosmos3-Nano-Transfer-Auto** model. Stage 0 comprises three phases: modality-separate training, short-sequence joint training, and long-sequence joint training with dropout. We then apply the shared three-stage causal recipe: teacher forcing (Stage 1), teacher-forcing discrete consistency distillation (Stage 2), and self-forcing distribution matching distillation (Stage 3). The resulting **Cosmos3-Nano-Sim-Auto** generates synchronized sensor streams in small latent chunks with cached history. The training curriculum is detailed in Sec. 7.3.

7.1. Architecture

Cameras and LiDAR capture complementary aspects of the same scene at different spatial resolutions and sampling rates. We bring these streams into a shared generator through modality-specific representations (Sec. 7.1.1), aligned temporal coordinates, and learned sensor identities (Sec. 7.1.2). Structured attention connects each stream to its language prompts, controls, and other sensors (Sec. 7.1.3). For streaming generation,

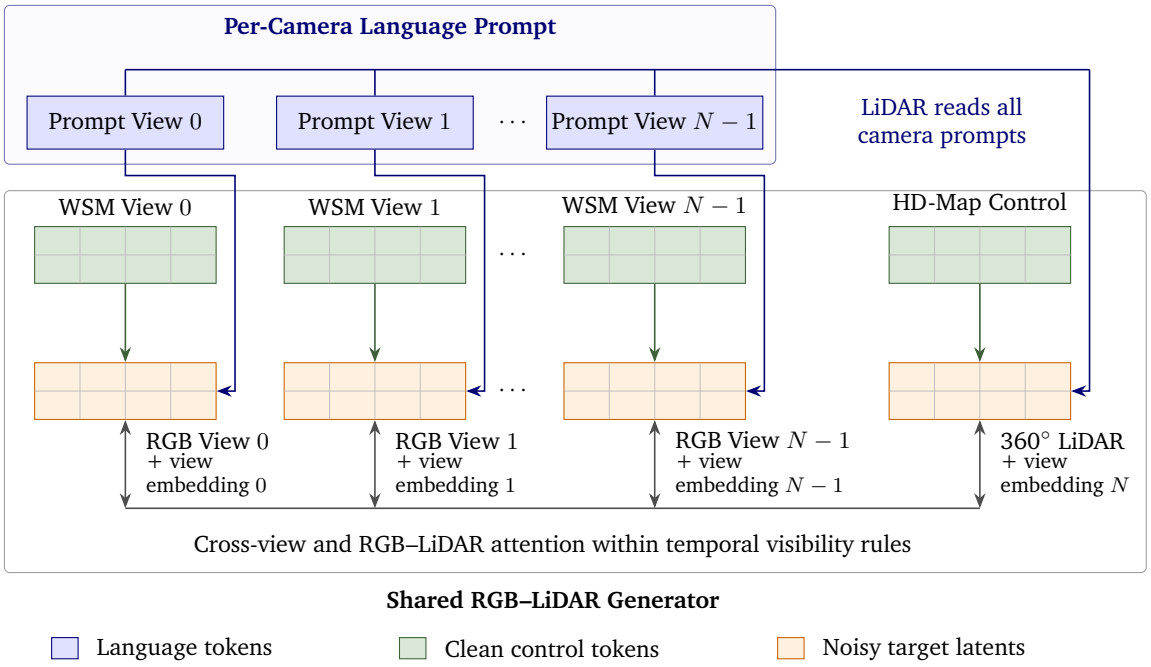


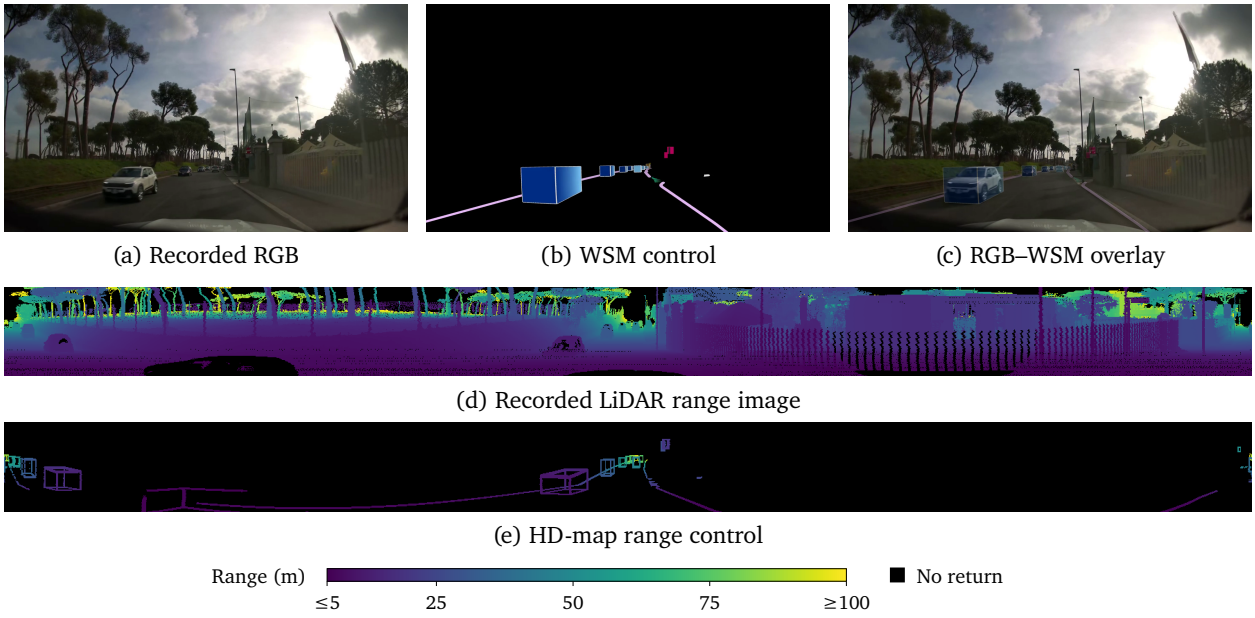
Figure 17: **Multi-view RGB–LiDAR generation.** Each column represents a camera or LiDAR stream. Colors denote language tokens (lavender), clean controls (green), and noisy target latents (light orange), as in Fig. 2. Each camera reads its own prompt and WSM control, while LiDAR reads all camera prompts and its HD-map control. Learned identity embeddings distinguish cameras $0, \ldots, N-1$ and LiDAR N, retaining their identities when selecting camera subsets. Arrows connect conditioning inputs to their consumers, and the lower bidirectional links show cross-sensor attention under the temporal visibility rules. Representative views are shown, with tokenizers and optional initial observations omitted. See Sec. 7.1 to 7.3 for the model architecture, caption preparation, and training recipe.

we adapt these connections to predict synchronized sensor chunks from the available history (Sec. 7.1.4).

7.1.1. Modality Representations

RGB signal. RGB generation inherits the video tokenization and transfer sequence-packing scheme of Cosmos 3 (Sec. 2 and 2.2). The same video VAE encodes RGB targets and rendered WSM controls into 48-channel latents with 16×16 spatial and $4\times$ temporal compression. Learned projections map 2×2 spatial latent patches into generator features. Control tokens remain clean while target tokens undergo denoising. The multi-view extension encodes each camera independently and packs the synchronized streams together, retaining their physical timestamps and adding learned camera identity embeddings. Each camera stream is associated with its own language prompt, while cross-view attention connects the visual streams under the visibility rules described in Sec. 7.1.3.

LiDAR signal. For the LiDAR signal, we use a separate hierarchical transformer VAE inspired by LidarVerse (Che et al., 2026), with interleaved spatial neighborhood attention and causal temporal attention. Each 360° sweep is projected onto a 128×1800 range image with range, intensity, and binary return-validity channels. The azimuth dimension is circularly padded to 1808 columns to accommodate the tokenizer's 16×16 spatial compression. There is no temporal downsampling: a sequence of T sweeps produces latents of shape $(128, T, 8, 113)$, ordered as channels, sweeps, height, and width, with the batch dimension omitted. The generator uses 1×1 spatial patches on this latent grid, yielding $8 \times 113 = 904$ tokens per sweep. A learned input projection maps each 128-channel latent patch into the shared generator feature space. A separate output projection predicts rectified flow velocity in LiDAR latent space (Liu et al., 2023). After denoising, the LiDAR decoder reconstructs range, intensity, and return validity from the completed latents.



(a) Recorded RGB (b) WSM control (c) RGB–WSM overlay

(d) Recorded LiDAR range image

(e) HD-map range control

Figure 18: **Sensor-aligned structural controls.** Recorded RGB, WSM, and their overlay (top) illustrate camera-control alignment in one data sample. A recorded LiDAR range image and its HD-map control are shown below.

Structural controls. WSM videos specify the scene layout and actor motion that guide RGB video generation (NVIDIA, 2025a).

The map layers include lane lines, road boundaries, crosswalks, road markings, wait lines, poles, traffic signs, and traffic lights. Projected 3D cuboids represent vehicles, pedestrians, and other road users. Semantic colors identify object categories, cuboid shading conveys heading, and successive frames describe motion. Rendering along the ego trajectory with each camera's intrinsics and extrinsics implicitly conveys ego motion and viewing geometry (Basant et al., 2026). The RGB–WSM overlay in Fig. 18 illustrates the resulting alignment.

For LiDAR generation, we render HD-map structure in the range-view projection used for the sensor sweeps. These controls provide spatially aligned map geometry for predicting LiDAR returns. Fig. 18 pairs the recorded LiDAR range image with its HD-map control at the same timestamp.

Sensor timing and temporal coordinates. Camera and LiDAR observations are placed on a shared physical timeline to align conditioning inputs and prediction targets (Fig. 19). The Wan 2.2 RGB tokenizer applies $4\times$ temporal compression to the 30-FPS camera streams, yielding latent time steps spaced $4/30$ seconds apart (7.5 Hz). The separate LiDAR tokenizer preserves the 10-Hz sweep rate without temporal downsampling, retaining a latent spacing of $1/10$ seconds (Sec. 2.2). These different latent sampling rates mean that equal temporal indices across modalities need not correspond to the same capture time. We therefore align the streams by physical timestamp while retaining each token's sensor identity. Capture time determines this alignment independently of diffusion time, rotary coordinates, and packed sequence order.

7.1.2. Positional Encoding and View Identity

Positional encoding. We retain the three-axis MRoPE formulation of Cosmos 3, using camera-local spatial coordinates and a shared temporal origin across synchronized views. Each camera uses the same spatial indexing convention, while temporal coordinates follow the FPS scaling described in Sec. 2. Consequently, tokens at matching image-grid locations and timestamps receive identical rotary positional encodings across cameras, irrespective of their order in the packed sequence. These coordinates describe position within each view, rather than a common 3D location or camera identity.

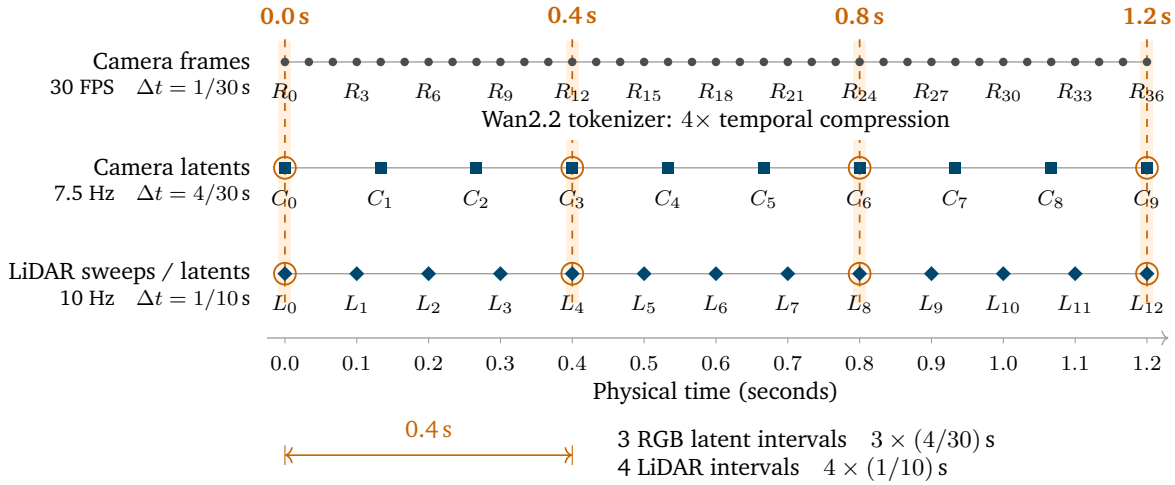


Figure 19: **Camera and LiDAR sampling on a shared physical timeline.** The illustrative grids share a temporal origin. Camera frames (R_i) are sampled at 30 FPS, and the Wan2.2 tokenizer applies $4\times$ temporal compression to produce camera latent steps (C_j) spaced by $4/30$ seconds (7.5 Hz). LiDAR sweeps and their latent steps (L_k) retain the same $1/10$-second spacing. Dashed guides and rings mark coincident camera-latent and LiDAR timestamps every 0.4 seconds, spanning three RGB latent intervals and four LiDAR intervals. Markers denote temporal positions rather than tokenizer receptive fields. Cross-sensor attention compares physical timestamps even when samples do not coincide.

Learned view embeddings. Inspired by Cosmos-Transfer2.5 Multiview (NVIDIA, 2025a, Sec. 6.3.1), we introduce learned per-view embeddings to distinguish cameras that share the same positional coordinates. After tokenization and projection into transformer features, we add a camera-specific embedding vector to every token from that camera. This acts as a per-view bias shared across spatial locations and time, learned jointly with the generator. The embedding complements the view-dependent geometry in WSM controls by providing a persistent camera identity across scenes. We find this simple design effective for our fixed training rig, where each identity consistently corresponds to a camera placement and field of view.

7.1.3. Multi-View and Multimodal Attention

Multi-view, multimodal attention is the core mechanism that brings our camera and LiDAR streams, language prompts, and structural controls together (Fig. 20). This section describes the attention design of Cosmos3-Nano-Transfer-Auto, including how tokens interact across views and modalities. The changes for Cosmos3-Nano-Sim-Auto are described in the Causal Modeling section (Sec. 7.1.4).

Language-to-sensor conditioning. Each camera has its own caption, which we tokenize separately. For language conditioning, the camera's visual tokens attend only to the tokens from that caption (Fig. 20(a)). This keeps descriptions of object visibility, appearance, and image-plane motion tied to the view they describe. LiDAR tokens attend to all camera caption tokens from the same recording. These captions cover the same synchronized sensor interval, providing language context from multiple viewpoints for the 360° LiDAR stream.

Control conditioning. WSM and HD-map controls enter the generator through the sequence-packing scheme for control-conditioned generation (Sec. 2.2). Their encoded tokens are packed alongside the sensor tokens and remain clean throughout denoising. Each camera stream attends to its own WSM tokens, while the LiDAR stream attends to its HD-map tokens (Fig. 20(a)). This access is separate from sensor-to-sensor attention: exchanging information across cameras does not give a camera direct access to another camera's control tokens. The available control sequence and conditioning masks determine temporal access.

Control tokens also attend to other control tokens within their own stream. When control-to-sensor attention is enabled, they additionally read sensor features from that stream, allowing the control representation to

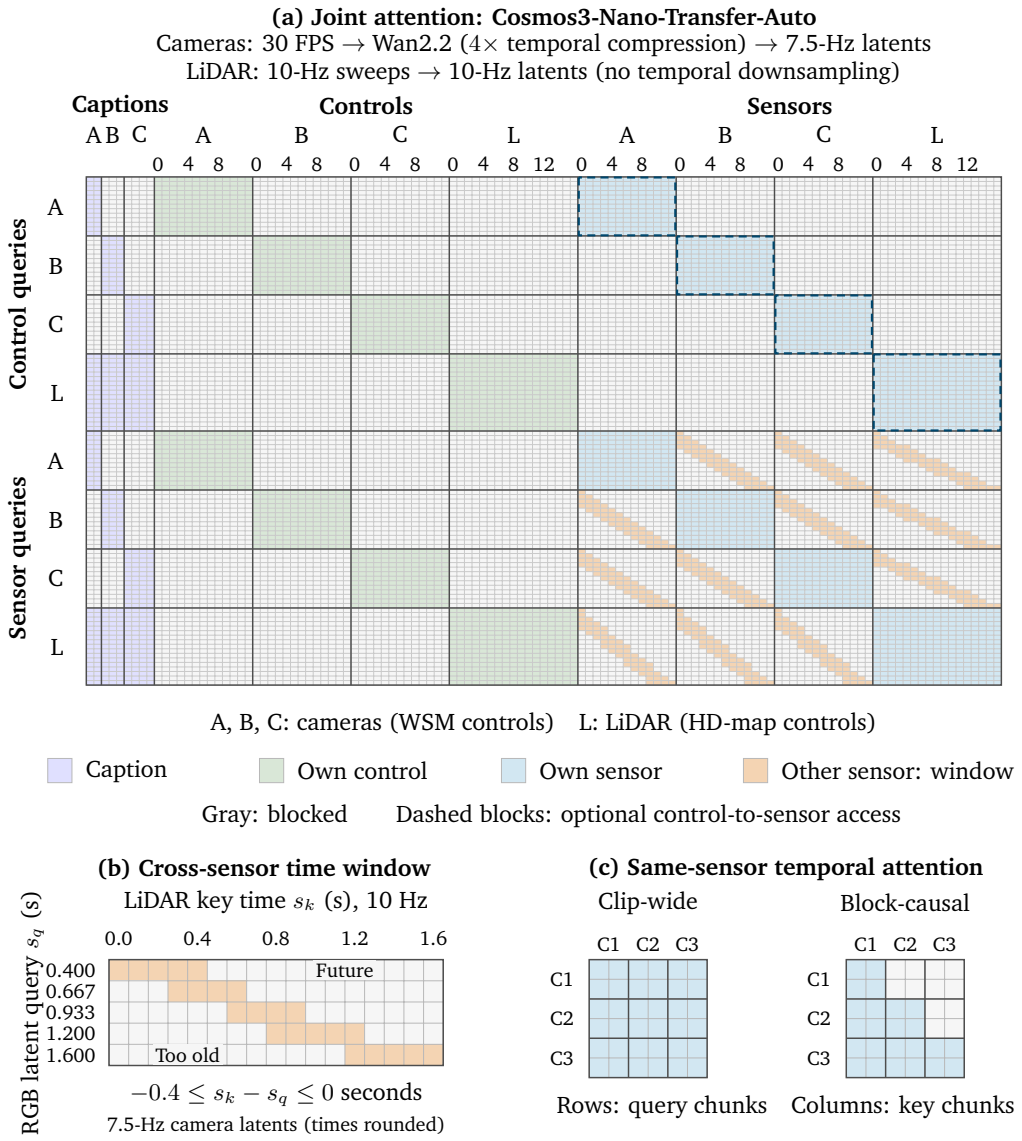


Figure 20: **Joint camera–LiDAR attention masks.** Rows are queries and columns are keys/values. (a) Cosmos3-Nano-Transfer-Auto with optional control-to-sensor access enabled (dashed blocks). Camera sensor and control queries read their own captions, while LiDAR queries read all camera captions (purple). Green and blue show own-control and own-sensor access. Only sensor queries read other sensors within the preceding 0.4-second window (orange). Wan2.2's $4\times$ temporal compression maps 30-FPS camera streams to 7.5-Hz latents, while LiDAR retains 10 Hz. Temporal indices span $[0, 1.6]$ seconds: 13 camera latent times and 17 LiDAR times make the moving 0.4-second window visible as a narrow band. Caption lengths are illustrative. (b) Physical timestamps bound the orange band, excluding keys older than 0.4 seconds and future keys. (c) Same-sensor attention is clip-wide for Cosmos3-Nano-Transfer-Auto and block-causal for Cosmos3-Nano-Sim-Auto, which reads the current chunk and permitted earlier clean history. Thick lines separate illustrative chunks C1–C3. For Cosmos3-Nano-Sim-Auto, control-to-sensor access is restricted to preceding clean history. Gray cells are blocked, spatial tokens are collapsed, and caption query rows are omitted. All routes respect conditioning visibility, recording boundaries, and available context.

incorporate sensor context. In Cosmos3-Nano-Sim-Auto, this context is restricted to clean sensor history from preceding chunks. The sensor-to-sensor routes described below connect supplied observations and prediction targets under their own temporal and clean/noisy visibility rules.

Sensor-to-sensor attention. A straightforward extension from single-view to multi-view generation lets every visual token attend to all tokens across cameras and time. The resulting cost grows quadratically with both the number of views and the sequence length, making long surround-view clips expensive to process. Inspired by OmniDreams (Basant et al., 2026), we decompose attention into temporal interaction within each view and information exchange across views. Where OmniDreams connects different views at the same time step, we extend cross-sensor interaction to a preceding 0.4-second window in physical time. This lets cameras and LiDAR exchange nearby observations even when their sampling times do not coincide. Same-sensor attention still allows each token to attend to all spatial and temporal tokens (clip-wide) within the same view.

For N_{v} camera views with N_{t} latent frames per camera and N_{s} spatial tokens per latent frame, unrestricted attention is quadratic in the total token count $N_{\mathrm{v}}N_{\mathrm{t}}N_{\mathrm{s}}$. An idealized decomposition that uses all same-view times and only the matching time across cameras instead has costs

$$C_{\text{full}} = \mathcal{O}\big((N_{\mathrm{v}}N_{\mathrm{t}}N_{\mathrm{s}})^2\big), \tag{28}$$

$$C_{\text{decomposed}} = \mathcal{O}(N_{\mathrm{v}}N_{\mathrm{t}}^2N_{\mathrm{s}}^2 + N_{\mathrm{v}}^2N_{\mathrm{t}}N_{\mathrm{s}}^2). \tag{29}$$

Taking the ratio of the leading terms in Eqs. (28) and (29) for 11 views and 100 latent frames per camera (397 RGB frames) gives an approximately $9.9\times$ theoretical reduction in camera attention computation.

Sliding window cross-sensor attention. Cross-sensor attention uses a sliding window spanning the preceding 0.4 seconds, including the query timestamp (Fig. 20(b)). RGB and LiDAR latent time steps are spaced by $4/30$ seconds and $1/10$ seconds, respectively. Attention visibility is determined by physical timestamps to account for the different sensor sampling rates. The 0.4-second window matches their smallest shared temporal period, spanning 3 RGB latent intervals and 4 LiDAR latent intervals. In our evaluation, this preceding window yielded better results than a symmetric window extending 0.2 seconds before and after the query timestamp. At each permitted timestamp, queries attend to the full spatial token grids of eligible streams from other sensors. The cross-sensor window explicitly excludes the query's own view, whose eligible keys are already handled by same-stream attention. Including that same-view window again in the cross-sensor branch yielded worse results in our evaluation.

7.1.4. Causal Modeling

Interactive driving simulation requires camera and LiDAR outputs to advance together as scene controls change. We adapt Cosmos3-Nano-Transfer-Auto for this setting by generating short, synchronized sensor chunks conditioned on the supplied controls and previously generated observations. Cosmos3-Nano-Sim-Auto retains the per-view language routing, camera embeddings, structural conditioning, and decomposed sensor-to-sensor attention described in Sec. 7.1.2 and 7.1.3.

Synchronized sensor chunks. RGB and LiDAR advance at different sampling rates, so a common frame count would cover different physical durations. We instead group their predictions by the shared 0.4-second interval introduced in Sec. 7.1.3. After the initial time step, each full chunk contains 12 RGB frames per camera, represented by 3 latent time steps, and 4 LiDAR sweeps. The model jointly denoises these aligned sensor intervals while preserving the distinction between same-stream and cross-sensor visibility.

Causal multi-view attention. Cosmos3-Nano-Sim-Auto replaces the clip-wide same-sensor attention of Cosmos3-Nano-Transfer-Auto with the shared block-causal formulation (Sec. 2.3). Within each sensor stream, queries attend to all spatial tokens throughout the current chunk and to permitted clean history from preceding chunks (Fig. 20(c)). Across streams, the preceding 0.4-second window remains unchanged (Fig. 20(b)), so a query cannot read another sensor's later timestamp even when both belong to the current chunk. The control and clean/noisy conditioning rules remain those described in Sec. 3.2 and 7.1.3. Future target chunks are excluded, allowing generation to advance without depending on future sensor outputs. Conditioning masks continue to prevent noisy targets from attending to their own clean reference copies.

Continuing the rollout. Once a chunk is complete, its clean sensor representations become history for the next prediction, which can incorporate newly supplied scene controls. Cosmos3-Nano-Sim-Auto uses the shared streaming mechanism in Sec. 4.1 to reuse this history and bound the retained cache during extended rollouts. Attention visibility determines which stored tokens may be read, while the cache policy determines which past chunks remain available. The Cosmos3-Nano-Sim-Auto runtime settings are described in Sec. 7.4. We train this causal generator through the shared teacher-forcing, consistency-distillation, and self-forcing stages, with settings and checkpoint handoffs for Cosmos3-Nano-Sim-Auto detailed in Sec. 7.3.

7.2. Data Preparation

Sources and sampling. We train on synchronized multi-camera driving recordings from MADS, previously used for WSM-conditioned generation in Cosmos 3 (NVIDIA, 2026a). Our captioned training selection contains approximately 988K source samples. Each recording is counted as one sample, encompassing its camera and LiDAR streams before temporal chunking. We hold out 91 clips as the Driving-Sim-Eval set[3] for benchmarking and exclude them from training. The evaluation protocol and results are described in Sec. 7.5. Our Cosmos3-Nano-Sim-Auto training inputs comprise up to 11 camera views, including wide-angle, telephoto, and fisheye cameras, together with 360° LiDAR sweeps. RGB videos are sampled at 30 FPS and LiDAR at 10 Hz.

For captioning, we partition each source sample into consecutive 400-frame RGB chunks with time-aligned structural controls and LiDAR. When the final chunk contains fewer than 400 frames, we retain and caption it as well. Each source sample yields an average of 1.56 caption chunks per camera, typically 1–2 chunks. The resulting collection is dominated by 400-frame chunks but also exposes the model to shorter sequences, providing diversity in training durations. Camera subsets retain their physical identities and timestamps within each synchronized interval. During training, we downsample the high-resolution RGB streams to a mixture of 480p and 720p clips, with stage-specific sequence lengths described in Sec. 7.3.

Structural controls and alignment. We render controls according to the WSM v4 schema, using each camera's calibrated lens model, including fisheye distortion. The WSM color and symbol encodings are illustrated in Fig. D.3 in the appendix. RGB targets and WSM controls use matching frame selection, resizing, and cropping so that the rendered geometry remains aligned with the recorded image. For LiDAR, HD-map geometry is projected onto the same range-view rays as the measured sweeps. We use LiDAR sweeps without motion compensation, retaining the spinning sensor's within-sweep motion distortion. RGB and LiDAR intervals are matched by physical timestamps rather than equal frame indices. We include the WSM rendering schema, reference renderer, and data-preparation guides in the accompanying open-source Cosmos Framework release.[4]

Caption preparation. WSM controls describe scene structure and actor motion, but leave much of the visual appearance unspecified. We complement them with dense, timestamped captions that describe visible objects and their appearance, together with time of day, lighting, weather, road-surface condition, and geographic region. Descriptions emphasize concrete visual attributes, such as vehicle color, surface reflections, and atmospheric visibility. An example timestamped caption is shown in App. D.2.3.

We caption each chunk with an in-house vision-language model that receives synchronized RGB and WSM observations from all camera views together. The joint input allows the captioner to reason about one physical scene before producing a separate description for each camera. RGB provides appearance and environmental evidence, while WSM supplies the corresponding geometry, actor trajectories, and signal states. Each caption describes only what its camera can observe, while shared objects retain consistent appearance and physical motion across views and time. Scene-level conditions must also agree across cameras. Captions account for each camera's viewpoint: the same vehicle may enter and leave different views at different times, or move leftward in one image and rightward in another, while following a consistent physical trajectory. Captioned

[3]Cosmos3-Driving-Simulation-Evaluation: huggingface.co/datasets/nvidia/Cosmos3-Driving-Simulation-Evaluation.
[4]Cosmos-Framework: github.com/NVIDIA/cosmos-framework.

Table 13: **Training recipe for Cosmos3-Nano-Transfer-Auto and Cosmos3-Nano-Sim-Auto.** Training uses 128 GB200 GPU nodes (512 GPUs); stage-specific allocations for causal training and distillation are listed in Tab. D.5. Training updates are counted separately within each bidirectional phase and causal stage. RGB counts are per view after alignment and include observed prefixes. Stage 0 learning rates denote schedule peaks. SF-DMD update counts and learning rates are listed as student / fake-score model.

Stage	Training updates	RGB frames	Learning rate	Teacher CFG
Stage 0: Bidirectional training				
Modality-separate	60,000	≤ 397	8×10^{-5}	—
Short-sequence joint	20,000	61	5×10^{-5}	—
Long-sequence joint	20,000	≤ 397	5×10^{-5}	—
Stage 1: TF	3,200	≤ 397	10^{-5}	—
Stage 2: TF-dCM	900	≤ 397	2×10^{-6}	1.5
Stage 3: SF-DMD				
Short sequence	250 / 1,250	93	10^{-6} / 2×10^{-7}	1.5
Long sequence	50 / 250	397	10^{-6} / 2×10^{-7}	1.5

events and motion must agree with the WSM controls.

At inference, users can perform *prompt upsampling* using WSM controls and four environmental attributes: time of day, weather, road condition, and geographic region. This produces a coherent set of per-camera prompts across all views. The workflow is described in App. D.2.3, with practical instructions in the accompanying Cosmos Framework guide.

Prompt prefixes. During training, we sample different camera subsets (Sec. 7.3), so each example can present a different combination of viewpoints. We therefore prepend a camera prefix that describes the currently sampled rig and identifies the current camera by its position or role, viewing direction, lens type, and nominal field of view. This provides the model with context about which views are present and the viewing direction and angular coverage associated with each caption. The task prefix explains WSM-conditioned generation: controls specify the supported objects, road layout, and motion, while captions supply appearance and unconstrained background details. The joint RGB–LiDAR task instruction additionally identifies the HD-map controls and calls for camera–LiDAR alignment. Examples of both prefixes are provided in App. D.2.3.

These prefixes are assembled by the caption formatter and text-tokenization pipeline during training and inference. The stored captions contain the scene descriptions, so the same rig and task context can be applied consistently without duplicating it in the annotations. The camera prefix complements the learned sensor embeddings in Sec. 7.1, while the task prefix states how language and structural controls are combined.

7.3. Post-training

We summarize our model's training stages as a recommended recipe for adapting Cosmos3-Nano to multi-view RGB–LiDAR generation and distilling it into a causal streaming model. The three phases of bidirectional training, collectively termed Stage 0, produce Cosmos3-Nano-Transfer-Auto. The three causal training and distillation stages introduced in Sec. 3 convert the bidirectional model into its streaming counterpart, Cosmos3-Nano-Sim-Auto. We periodically evaluate checkpoints on a validation set and advance to the next training phase or stage when benchmark scores plateau.

Trainable parameters. During this adaptation, we train the shared generator blocks, diffusion-time embedding, RGB and LiDAR input/output projections, and learned sensor embeddings. The pre-trained reasoner and both sensor tokenizers remain frozen. The causal training and distillation stages retain this parameter selection for the student, while the distillation teacher remains fixed. SF-DMD additionally trains the fake-score network, following the shared method in Sec. 3.4.

Objectives and modality balance. Stage 0 trains RGB and LiDAR generation with the rectified flow objective inherited from Cosmos 3 (NVIDIA, 2026a). The two modalities have different latent dimensions and token counts, so we compute their losses separately before combining them with RGB:LiDAR weights of $1:2$ during joint training. For RF and TF, following Cosmos 3, we average the squared velocity error within each stream, then across streams of the same modality. Observed prefixes and structural controls remain clean, with their prediction errors masked to zero before this reduction. The reduction retains these entries in its averaging denominators, including the zero-loss control streams. The causal training and distillation stages retain this $1:2$ weighting with the shared objectives in Sec. 3; their loss reductions are detailed in App. D.3.2.

Camera-rig sampling. Different applications require different camera coverage, from a single forward view to a complete surround rig. We expose Cosmos3-Nano-Transfer-Auto to these configurations by sampling predefined camera combinations rather than always presenting all 11 views. The catalog contains 15 commonly used camera combinations spanning 1–11 views, listed in Tab. D.4 in the appendix. For joint training, we sample 480p and 720p clips with probabilities 0.6 and 0.4, respectively. At 480p, the standard 7-camera rig receives probability 0.6, the full 11-camera rig receives 0.3, and the remaining 13 combinations share 0.1 uniformly. At 720p, we sample the full 7-camera rig with probability 0.9. To fit the training memory budget, we use 480p whenever all 11 cameras are selected together. At 720p, each example contains fewer cameras, but different sampled combinations expose the model to all 11 camera views. Learned view embeddings identify individual cameras: the front-wide camera, for example, always uses its front-wide embedding regardless of which other cameras are selected. The camera prefix describes the sampled rig, as explained in Sec. 7.2.

Conditioning robustness. To support RGB-only generation with the joint model, we drop LiDAR together with its HD-map control on 10% of joint-training examples. We also vary the available language conditioning to accommodate incomplete prompts and support classifier-free guidance. Clip-level caption dropout removes all camera captions together with probability 0.2. Independently, per-view caption dropout removes each camera's scene description with probability 0.2. We train on Transfer-T2V and Transfer-I2V modes equally. Transfer-I2V supplies 1 initial RGB frame per camera during training, while Transfer-T2V supplies none.

7.3.1. Cosmos3-Nano-Transfer-Auto Training

In practice, we found that directly adapting the single-view Cosmos3-Nano to joint multi-view RGB–LiDAR generation on 400-frame source chunks led to slow convergence, particularly in LiDAR prediction quality. These experiments motivate the recommended three-phase curriculum below: first learning the modalities separately, then training them jointly over short sequences, and finally increasing the joint sequence length. Tab. 13 gives the recommended training budgets and learning rates.

Modality-separate training. We first train RGB and LiDAR generation as separate tasks within the shared generator, establishing LiDAR generation while adapting RGB generation to the driving domain. RGB examples contain a single camera view with up to 397 aligned frames, while LiDAR examples contain 21 sweeps.

Short-sequence joint training. We next train on synchronized multi-view RGB and LiDAR together, using 61 RGB frames per camera and 21 LiDAR sweeps, approximately 2 seconds. This phase learns cross-view and cross-modal relationships over short temporal contexts.

Long-sequence joint training. Finally, we extend the joint training window to up to 397 RGB frames per camera and 133 LiDAR sweeps, approximately 13.2 seconds, to learn scene evolution over longer temporal contexts. In practice, we found that longer temporal contexts during bidirectional training benefit the stability of the subsequently distilled model. We choose 400-frame source windows to balance this benefit with training memory. When splitting a recording leaves fewer than 400 frames at the end, we also use that shorter segment for training (Sec. 7.2). Shorter clips, such as 300-frame videos, are also suitable, and clips of different lengths can be mixed during training.

7.3.2. Cosmos3-Nano-Sim-Auto Training

We follow the shared three-stage recipe in Sec. 3, using the synchronized RGB–LiDAR chunks and causal attention described in Sec. 7.1.4. At each causal-stage transition, we initialize the student from the preceding stage's regular weights rather than its exponential moving average (EMA) weights, and reset the optimizer state and iteration counter. Tab. 13 summarizes the training budgets and learning rates.

All three causal stages use AdamW (Loshchilov and Hutter, 2019) with weight decay 0.01, gradient norm clipping at 1.0, and no gradient accumulation. The momentum coefficients (β_1, β_2) are $(0.9, 0.999)$ in Stage 1 and $(0, 0.999)$ in Stages 2 and 3. The learning rates in Tab. 13 remain constant within each stage.

Stage 1: causal teacher forcing. We initialize the causal model from the Stage 0 bidirectional model and train it to predict synchronized RGB and LiDAR chunks from recorded sensor history, following the teacher-forcing (TF) method in Sec. 3.2.

Stage 2: consistency distillation. We next distill the causal model into a 4-step generator using TF-dCM (Zheng et al., 2026c), following Sec. 3.3. Both the student and frozen causal teacher initialize from the Stage 1 checkpoint. Training retains recorded sensor history and uses teacher CFG with scale 1.5.

Stage 3: self-forcing. Starting from the Stage 2 student, SF-DMD trains on RGB–LiDAR rollouts conditioned on the student's own generated history, following Sec. 3.4. The frozen DMD teacher and trainable fake-score network initialize from the Stage 0 bidirectional model, with teacher CFG again set to 1.5.

SF-DMD alternates updates to the student and fake-score networks at a $1:5$ ratio. We first train on 93-frame clips for 1,500 total updates, comprising 250 student updates and 1,250 fake-score updates. We then continue on 397-frame clips for 300 updates (50 student and 250 fake-score updates) while preserving the optimizer state and iteration counter.

In practice, too few SF-DMD updates left object artifacts, including floating objects, while too many introduced temporal flicker. We selected the update budget to balance object integrity and temporal consistency, guided by quantitative metrics and visual evaluation.

7.4. Infrastructure and Inference Optimization

7.4.1. Training and Attention Implementation

Training infrastructure. We pack synchronized RGB, LiDAR, structural controls, and language using a target budget of 1.5M tokens per sequence. On GB200 GPUs, the long-sequence configuration combines context parallelism and fully sharded data parallelism (FSDP) (Zhao et al., 2023), with BF16 computation and activation recomputation to limit memory use.

Attention implementation. We initially implemented the decomposed visibility pattern in Sec. 7.1.3 and 7.1.4 with PyTorch FlexAttention and a FlashAttention-4 backend (Zadouri et al., 2026). Sensor frame boundaries do not generally align with the kernel's query–key blocks, leaving partially visible blocks that require elementwise masking. This overhead reduces attention efficiency despite the smaller number of permitted query–key pairs, as illustrated in Fig. D.1 in the appendix.

Our multi-view backend applies the maskless approach described in Sec. 4.2 to same-view, cross-sensor, control, and language interactions. We gather permitted tokens into segments for variable-length attention (PyTorch Contributors, 2026b), then scatter and merge the outputs in the original query order (Fig. 21). Token grouping and joint normalization are detailed in App. D.1.2. The maskless backend was 2.5–$3\times$ faster than FlexAttention for the tested decomposed attention operations.

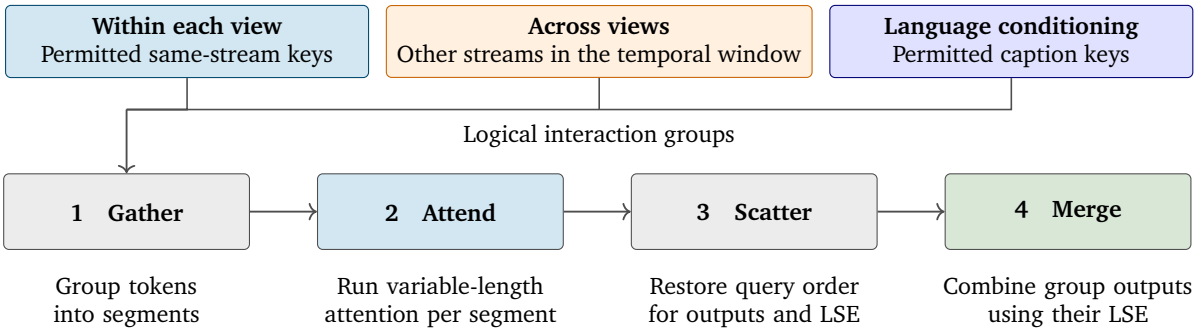


Figure 21: **Maskless execution of decomposed attention.** Query and key/value tokens are gathered into segments for the illustrated sensor and language interactions. Variable-length attention processes each segment, and the results are scattered back to the original query order. Each attention call returns a normalized output and log-sum-exp (LSE) statistics. LSE-weighted merging recovers joint normalization when key subsets are disjoint and attention scores match, as in Eq. (D.10). Grouping must preserve the model's temporal and conditioning restrictions. Arrows show data dependencies between logical operations.

Table 14: **Selected Cosmos3-Nano-Sim-Auto runtime operating points on GB200.** The released checkpoint generates 12-frame chunks at 832×480 resolution per camera with 4 denoising steps on the residential scene. Chunk times are medians of run medians, with repeat ranges in parentheses, and FPS is the measured rate presented to the client. Model and map ranks together give the total GPU allocation. Map ranks prefetch controls with a 1-chunk delay. Eager and optimized runs share an allocation, although the eager counterpart of the 8 model + 8 map row uses 16 model ranks. A dash indicates no eager comparison.

Views	Mode	GPUs	Layout	Chunk wall time (ms)		FPS
				eager	optimized	
1	RGB	1	1 model	524 (512–535)	364 (362–423)	32.6
4	RGB	16	8 model + 8 map	—	408 (377–440)	28.9
7	RGB	16	16 model	897	609	19.5
7	RGB	16	8 model + 8 map	950 (936–1,014)	583 (561–597)	20.5
7	RGB	32	16 model + 16 map	—	511 (510–511)	23.2
7	RGB + LiDAR	16	16 model	1,159	701 (693–714)	16.7

7.4.2. Inference Optimization

Streaming execution. We evaluate the released Cosmos3-Nano-Sim-Auto checkpoint in Interactive Drive, the FlashDreams demonstration client,[5] using the residential scene with prepared WSM controls on GB200 GPUs. The FlashDreams 0.2 release candidate generates 832×480 video per camera with 4 denoising steps and a bounded history cache. Each chunk produces 12 frames per camera at a video rate of 30 FPS, giving a 400 ms runtime target. These measurements cover up to 7 cameras plus LiDAR, rather than the full supported 11-camera rig.

Bounded history keeps steady-state shapes fixed for compiled denoising, control encoding, and decoding. The runtime uses maskless attention with a fused group merge and distributes camera decoding across model ranks. Separate map GPUs can prepare the next chunk's controls while the current chunk denoises, but introduce a 1-chunk control delay. Control-token processing and the clean K/V cache update remain on the model's execution path. Compilation also increases startup time, measured separately from steady-state throughput. The execution settings and optimization mechanisms are detailed in App. D.4 and Tab. D.6.

[5] FlashDreams: https://github.com/NVIDIA/flashdreams.

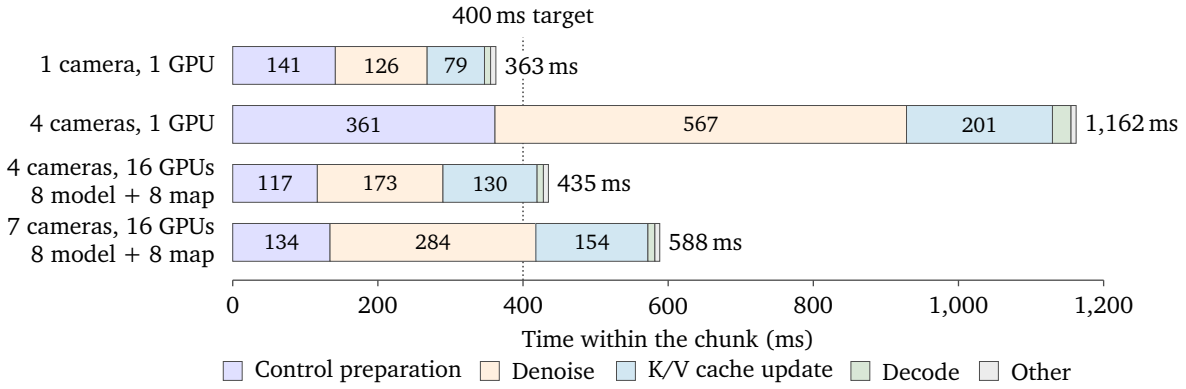


Figure 22: **Cosmos3-Nano-Sim-Auto per-chunk stage times of the shipped configurations on GB200.** Each bar is the median run of its configuration, split into the median stage times over its steady-state chunks on the primary rank. Control preparation covers WSM drawing and encoding, or the gather of prefetched latents, plus the control-token pass that writes control K/V. The bar length uses the backend's generation-call timer, and the gray remainder is work outside the 4 timed stages. Stage medians from a representative run need not sum to the across-run wall-time median.

Inference performance. Tab. 14 summarizes selected operating points from paired comparisons within the same GPU allocation. The shipped optimizations reduce median chunk wall time by 30% for 1 camera on 1 GPU and by 32% for 7 cameras on 16 model ranks. Compiled denoising provides the largest single gain in these comparisons. Maskless execution also matters at inference: replacing it and the fused merge with a block-sparse FlexAttention path increases chunk wall time by 195% for 7 cameras with compiled denoising on 8 model ranks and 8 map GPUs. This comparison measures the complete generation call, rather than the attention operations alone. The measurement protocol and full runtime results are provided in App. D.4.1 and Tab. D.7.

Separate map GPUs help only when enough model ranks remain for generation, as shown in the GPU-layout comparisons in Fig. D.4. For 4 cameras, 8 model ranks and 8 map GPUs take 408 ms compared with 545 ms for 16 model ranks in the same allocation. For 7 cameras, the two layouts fall within the repeat range, while adding LiDAR favors retaining all 16 GPUs as model ranks. The selected 7-camera configurations reach 23.2 FPS for RGB on 32 GPUs and 16.7 FPS for RGB plus LiDAR on 16 GPUs, both below the 30 FPS video rate. The 1-camera median meets the 400 ms target, while slower repeats and some 4-camera runs exceed it. These throughput measurements use prefetched controls, while responding to newly supplied conditions also requires control preparation.

Fig. 22 shows why faster denoising alone does not determine chunk latency. Control preparation and the clean K/V cache update remain eager and together exceed denoising time for the single-camera configuration, while approaching it for 7 cameras. The control-token pass remains necessary even when separate map GPUs prefetch the rendered and encoded controls. These costs limit the benefit of further denoising optimization and motivate measuring the complete chunk rather than only its diffusion steps. Additional stage and startup timings are provided in App. D.4.1.

7.5. Results

Evaluation setup. We evaluate on two held-out datasets. Driving-Sim-Eval (Sec. 7.2) contains 91 samples balanced across geography, control signals, and scene conditions. Traffic-Scenario-Eval is an internal collection of 319 samples for evaluating synthetic data generation.

Driving-Sim-Eval uses HD-map-derived WSM controls to assess generation quality. Traffic-Scenario-Eval benchmarks Transfer-T2V generation from noisy, perception-derived WSM controls and prompts specifying

Table 15: **Multi-view RGB results on Driving-Sim-Eval.** Models generate 297 frames per camera at 720p across 7 non-fisheye views, with (Transfer-I2V) or without (Transfer-T2V) initial RGB conditioning. Cosmos-Transfer2.5 denotes its multi-view variant. SG-V denotes StyleGAN-V features for FVD. Arrows indicate the preferred score direction. Bold and underlined values mark the best and second-best scores within each mode. Lighter and darker green identify the bidirectional and causal models, respectively.

(a) Transfer-T2V: no initial RGB frame

	Video quality				Temporal consistency		Cross-view alignment	
Model	**Frame FID ↓**	**FVD I3D ↓**	**FVD SG-V ↓**	**DOVER tech. ↑**	**TSE ↓**	**ACM ↑**	**CSE-2 ↓**	**CSE-5 ↓**
Cosmos3-Nano-Transfer-Auto	**45.5**	488	487	**9.01**	**0.473**	**0.906**	**2.009**	**1.633**
Cosmos3-Nano-Sim-Auto	48.6	**480**	**479**	7.30	0.503	**0.906**	2.599	2.244
Cosmos-Transfer2.5	61.9	772	770	8.56	0.530	0.885	2.511	2.073

(b) Transfer-I2V: with initial RGB frame

	Video quality				Temporal consistency		Cross-view alignment	
Model	**Frame FID ↓**	**FVD I3D ↓**	**FVD SG-V ↓**	**DOVER tech. ↑**	**TSE ↓**	**ACM ↑**	**CSE-2 ↓**	**CSE-5 ↓**
Cosmos3-Nano-Transfer-Auto	**27.8**	**238**	**237**	**8.27**	**0.472**	**0.902**	**2.246**	**1.798**
Cosmos3-Nano-Sim-Auto	33.6	325	324	7.51	0.505	0.900	2.359	2.085
Cosmos-Transfer2.5	41.4	388	387	8.25	0.519	0.881	2.935	2.203

road conditions, time of day, geographic location, and weather. We evaluate how well generated videos follow the controls and depict the requested attributes.

We evaluate joint RGB–LiDAR generation with Cosmos3-Nano-Sim-Auto and Cosmos3-Nano-Transfer-Auto, using the multi-view variant of Cosmos-Transfer2.5 as the RGB baseline (NVIDIA, 2025a). For each benchmark, every model, including the baseline, uses the prompt configuration that yields the highest generation quality among the variants we tested. Detailed inference settings and prompts are provided in App. D.5.1.

7.5.1. Evaluation Metrics

To assess both sensor realism and driving-scene fidelity, we combine general RGB and LiDAR quality metrics with traffic-object adherence metrics and human evaluation of traffic and environmental attributes.

RGB video quality metrics. RGB evaluation spans visual fidelity, temporal coherence, and cross-view geometric consistency. Frame Fréchet inception distance (FID) (Heusel et al., 2017) compares generated and reference frame-feature distributions. Fréchet video distance (FVD) (Unterthiner et al., 2019) compares video-feature distributions, with I3D (Carreira and Zisserman, 2017) and StyleGAN-V (Skorokhodov et al., 2022) results reported separately. The DOVER technical score (Wu et al., 2023) assesses perceptual video quality. Adjacent cosine mean (ACM) measures within-view visual consistency through CLIP-feature similarity to the preceding and first frames. Temporal Sampson error (TSE) measures geometric agreement between matched points in consecutive frames. Cross-view Sampson error (CSE) (Sampson, 1982) measures agreement across overlapping camera views, averaged over 2 or 5 view pairs for CSE-2 and CSE-5. Lower FID, FVD, TSE, and CSE are better, while higher DOVER and ACM are better. Reported scores are averages over the 7 non-fisheye views.

LiDAR metrics. For LiDAR, comparisons with real sweeps assess geometry and temporal consistency, while point support and detection agreement with ground-truth annotations measure object faithfulness. Bird's-eye-view (BEV) JSD and MMD^2 (Gretton et al., 2012) compare generated and real spatial distributions. Fréchet Utonia distance (FUD) and Utonia MMD^2 compare pre-trained point-cloud features (Zhang et al., 2026c). Lower values indicate closer distributions. Chamfer temporal consistency (CTC) measures sweep-to-sweep



Figure 23: **Multi-view RGB–LiDAR generation with Cosmos3-Nano-Transfer-Auto.** This urban intersection under strong sun glare illustrates joint generation across eleven cameras and LiDAR. The white van retains its appearance in the rear-right and rear-fisheye views, while the forward and right-facing cameras agree on the direction of sunlight. Overlaid WSM controls align with the generated road markings and vehicle positions. The views follow the rig's spatial layout around a central LiDAR sweep, whose height coloring distinguishes the road surface, nearby vehicles, and surrounding buildings.

geometric change after motion alignment, for full sweeps or tracked vehicles. Generated-to-real ratios closer to 1 are better. Object support recall measures the fraction of objects with sufficient real LiDAR points that also contain sufficient generated points within their ground-truth boxes. Detection F1 compares PointPillars (Lang et al., 2019) detections in generated point clouds with ground-truth object annotations. Higher recall and F1 indicate better object faithfulness.

Traffic-object adherence. AutoGrader uses object-level F1 to measure how faithfully generated vehicles, pedestrians, and cyclists follow WSM controls. Counts of detected, missing, and hallucinated objects come from the correspondence checks of Cosmos Evaluator (NVIDIA, 2026c). Higher scores indicate better adherence.

Human evaluation. Human evaluation (HUE) scores video quality, scene-element fidelity, and cross-view agreement on a 1–3 scale. For overall video quality, 3 denotes realistic, clean, temporally stable imagery; 2 allows minor artifacts but no physically impossible content; and 1 denotes severe degradation or any physically impossible event, even briefly. Scene-element and cross-view questions assess the specified objects and their consistency, with rubric details in App. D.5. Road-surface and weather scores are the percentages of evaluated samples whose generated conditions human raters judge to match the prompt. Higher scores are better. Following the annotation protocol of Cosmos 3 (NVIDIA, 2026a), 2 annotators independently rate each (video, question) pair. Disagreements are resolved by a third quality-control reviewer, whose decision is final.



Figure 24: **Multi-view RGB–LiDAR generation with Cosmos3-Nano-Sim-Auto.** Camera views are arranged by their viewing directions around the vehicle, with generated LiDAR shown centrally. WSM controls are overlaid on the RGB frames to highlight adherence to the supplied scene structure. Buildings, vehicles, and cyclists retain consistent geometry and appearance across overlapping views. The LiDAR point cloud is color-coded by height and captures corresponding scene geometry.

7.5.2. Results

Generation quality on Driving-Sim-Eval. We first assess the quality of the synchronized RGB and LiDAR outputs on Driving-Sim-Eval. Figs. 23 and 24 show Transfer-I2V examples from Cosmos3-Nano-Transfer-Auto and Cosmos3-Nano-Sim-Auto, respectively.[6] Each example contains 11 camera views and the corresponding LiDAR sweep, with camera-aligned WSM controls overlaid at 20% opacity.

RGB results. Cosmos3-Nano-Transfer-Auto improves on the multi-view Cosmos-Transfer2.5 baseline across every reported RGB metric in both Transfer-T2V and Transfer-I2V modes (Tab. 15). Frame FID improves by 26–33% and both FVD variants by 37–39%, indicating that the generated frame and video distributions more closely match the reference data. Higher DOVER scores further indicate improved technical video quality. The gains also extend to consistency: lower TSE and higher ACM indicate more coherent videos over time, while roughly 20% lower CSE-2 and CSE-5 reflect stronger geometric agreement across camera views.

The four-step causal Cosmos3-Nano-Sim-Auto improves frame FID, FVD, and temporal consistency over the Cosmos-Transfer2.5 baseline in both generation modes. In Transfer-T2V, it also slightly surpasses its bidirectional teacher, Cosmos3-Nano-Transfer-Auto, in I3D FVD (480 versus 488), while matching its ACM score of 0.906. Cross-view consistency improves over the baseline in Transfer-I2V, whereas Transfer-T2V Sampson errors are modestly higher, by 4–8%. DOVER scores also favor the baseline. Overall, the causal model combines four-step streaming generation with substantial gains in image- and video-level distributional quality and temporal

[6] See the project website for animated visualizations: `research.nvidia.com/labs/cosmos-lab/cosmos-simulation`.

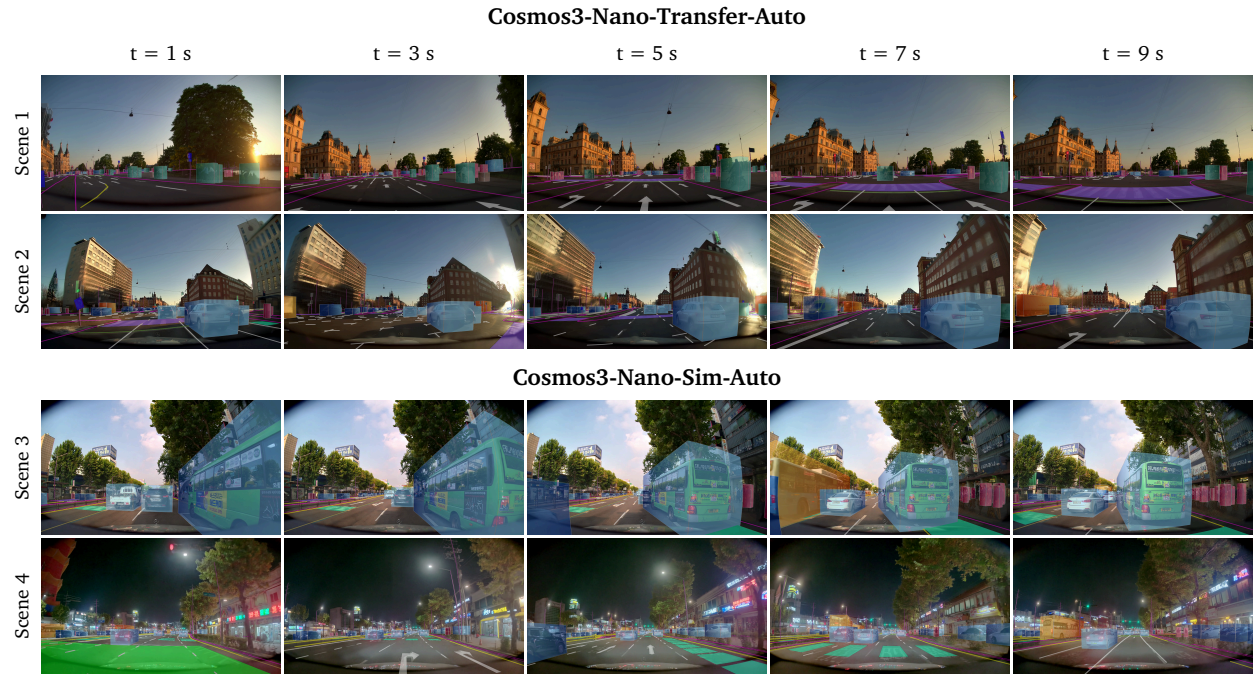


Figure 25: **Spatial control following in generated front-wide video.** WSM overlays show correspondence with vehicles, road markings, and traffic signals at five times after conditioning. The top two rows use Cosmos3-Nano-Transfer-Auto and the bottom two use Cosmos3-Nano-Sim-Auto.

consistency, alongside metric- and mode-specific trade-offs.

Qualitatively, the WSM overlays in Fig. 25 show close correspondence between generated vehicles and their control cuboids, together with alignment of road markings and traffic signals across the sampled frames.

LiDAR results. We evaluate the jointly generated LiDAR from the same Transfer-T2V rollouts as the RGB results, with 99 sweeps at 10 Hz (Tab. 16). LidarVerse is evaluated on 21- and 116-sweep rollouts, providing comparisons at both shorter and longer horizons than ours.

Cosmos3-Nano-Transfer-Auto achieves better quality scores than both LidarVerse settings on eight of the nine reported metrics. Global BEV JSD improves by 65–73% and Fréchet Utonia distance by 62–71%, indicating closer agreement with real LiDAR in both spatial point distributions and learned geometric features. Lower local BEV JSD and MMD^2 show that the gains also hold at the clip level. Temporal CTC is closer to 1, while higher object support recall and detection F1 indicate better preservation of annotated objects. Vehicle CTC is the only exception: 2.53 over our 99-sweep rollouts, versus 2.51 for the 21-sweep baseline. In the two Transfer-I2V scenes shown in Fig. 26, the generated LiDAR closely resembles the ground truth in road geometry, roadside structures, and nearby object placement.

Cosmos3-Nano-Sim-Auto also improves all five distribution-fidelity metrics over both LidarVerse settings, retaining strong spatial and geometric fidelity with 4-step streaming generation. Its temporal CTC, vehicle CTC, and object support recall improve on the 116-sweep baseline but trail the shorter 21-sweep baseline. Detection F1 is slightly lower than both baselines, at 0.188 versus 0.200–0.201. The causal distilled variant retains the distributional advantages over LidarVerse but falls short of Cosmos3-Nano-Transfer-Auto across the LiDAR metrics. Preserving the bidirectional model's LiDAR fidelity, temporal coherence, and object detail through multimodal causal distillation is a priority for future work.

Together, these results illustrate the strong capability of joint RGB–LiDAR generation with our shared generator. Causal distillation enables faster, 4-step streaming generation, with trade-offs in technical video quality,

Table 16: **LiDAR generation results in Transfer-T2V mode.** Cosmos 3 results use 99 sweeps at 10 Hz from the joint RGB–LiDAR rollouts. LidarVerse (Che et al., 2026) is evaluated on 21- and 116-sweep rollouts. Arrows indicate the preferred score direction, while CTC ratios target 1. Scale factors in the headers multiply the displayed values; FUD denotes Fréchet Utonia distance. Bold and underlined values mark the best and second-best scores. Lighter and darker green identify the bidirectional and causal models, respectively.

(a) Distribution fidelity

Model	BEV JSD ↓ global ($\times 10^{-2}$)	BEV JSD ↓ local	BEV MMD^2 ↓ local ($\times 10^{-3}$)	FUD ↓	Utonia MMD^2 ↓ ($\times 10^{-2}$)
Cosmos3-Nano-Transfer-Auto	**4.01**	**0.220**	**3.47**	**12.8**	**2.08**
Cosmos3-Nano-Sim-Auto	5.52	0.277	5.93	20.1	3.54
LidarVerse, 21 sweeps	15.1	0.400	12.0	33.6	6.36
LidarVerse, 116 sweeps	11.6	0.300	6.44	44.8	9.07

(b) Temporal consistency and object faithfulness

	Temporal consistency		Object faithfulness	
Model	CTC / real (target 1)	Vehicle CTC / real (target 1)	Object support recall ↑	Detection F1 at 3 m ↑
Cosmos3-Nano-Transfer-Auto	**1.17**	2.53	**0.885**	**0.237**
Cosmos3-Nano-Sim-Auto	1.28	3.59	0.816	0.188
LidarVerse, 21 sweeps	1.20	**2.51**	0.843	0.201
LidarVerse, 116 sweeps	1.35	3.73	0.788	0.200

Table 18: **Human evaluation on Traffic-Scenario-Eval.** Settings match Tab. 17. Cosmos-Transfer2.5 denotes its multi-view variant. Video quality, scene-element fidelity, and cross-view agreement use a 1–3 scale. Environment scores give the percentage of samples matching the prompted conditions. All scores are higher-is-better. Video quality uses 319 samples per model; other categories use subsets, including 316 samples for Cosmos3-Nano-Sim-Auto's road-surface and weather scores. Bold and underlined values mark the best and second-best scores. Light and dark green distinguish the bidirectional and causal models.

(a) Video quality and scene-element fidelity (1–3)

	Video quality	Scene-element fidelity				
Model	Video quality ↑	Lane ↑	Traffic light ↑	Traffic sign ↑	Pedestrian ↑	Bicyclist ↑
Cosmos3-Nano-Transfer-Auto	**1.72**	2.43	1.83	2.17	1.64	**1.85**
Cosmos3-Nano-Sim-Auto	1.39	**2.54**	**2.24**	**2.45**	**1.70**	1.83
Cosmos-Transfer2.5	1.65	2.28	1.59	1.78	1.20	1.38

(b) Cross-view agreement and environmental fidelity

	Cross-view agreement (1–3)					Environment scores	
Model	Lane/road markings ↑	Traffic light ↑	Stop sign ↑	Pedestrian ↑	Bicyclist ↑	Road surface ↑	Weather ↑
Cosmos3-Nano-Transfer-Auto	2.93	**2.93**	**2.98**	**2.94**	**2.98**	92.8%	**90.9%**
Cosmos3-Nano-Sim-Auto	**2.98**	2.91	2.94	2.94	2.96	87.0%	81.0%
Cosmos-Transfer2.5	2.77	2.68	2.76	2.58	2.63	**93.1%**	89.7%

cross-view consistency, and LiDAR fidelity. Nevertheless, Cosmos3-Nano-Sim-Auto retains many of Cosmos3-Nano-Transfer-Auto's gains over the video and LiDAR baselines. Beyond these general quality metrics, our models also improve traffic-object adherence over the video baseline, as we examine next.

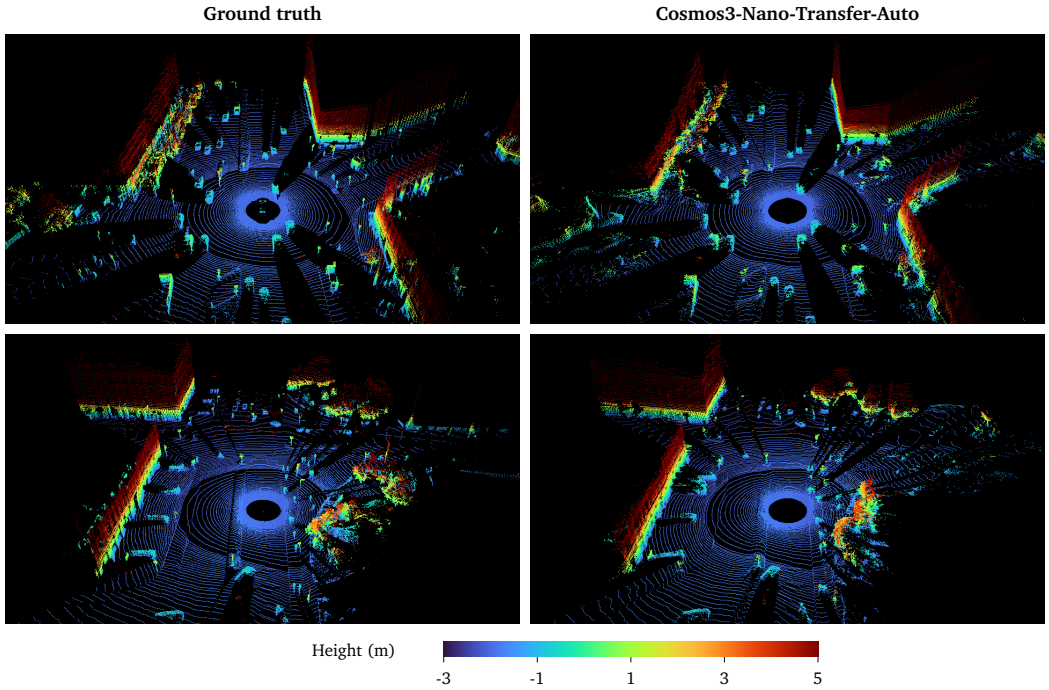


Figure 26: **LiDAR geometry with Cosmos3-Nano-Transfer-Auto.** Two Transfer-I2V scenes compare ground truth (left) with generated sweeps (right), using matched viewpoints and a shared height scale. The examples reproduce the main road and roadside structure, with clear individual objects.

Table 17: **AutoGrader object following on Traffic-Scenario-Eval.** F1 measures correspondence to WSM-controlled objects across 319 Transfer-T2V samples. Inference settings are given in App. D.5.1. Higher F1 is better on the 0–1 scale. Bold and underlined values mark the best and second-best scores. Lighter and darker green identify the bidirectional and causal models, respectively.

Model	Vehicle F1 ↑	Pedestrian F1 ↑	Cyclist F1 ↑
Cosmos3-Nano-Transfer-Auto	**0.908**	<u>0.386</u>	**0.537**
Cosmos3-Nano-Sim-Auto	<u>0.853</u>	**0.389**	<u>0.428</u>
Cosmos-Transfer2.5	0.741	0.137	0.151

Traffic-scenario fidelity on Traffic-Scenario-Eval. We next turn to Traffic-Scenario-Eval to assess how the generated videos follow traffic controls and depict the requested scene attributes.

Traffic-object adherence results. Cosmos3-Nano-Sim-Auto combines 4-step streaming generation with stronger traffic-object adherence than Cosmos-Transfer2.5 across all three categories (Tab. 17). The gains are particularly pronounced for pedestrians and cyclists. Compared with Cosmos3-Nano-Transfer-Auto, pedestrian F1 is similar (0.389 versus 0.386), while vehicle and cyclist F1 are 6% and 20% lower, respectively. Despite these gaps, Cosmos3-Nano-Sim-Auto combines substantial control-following gains over the video baseline with 4-step streaming generation.

Human evaluation results. Human evaluation complements the object-level metrics by assessing visual quality, scene attributes, and agreement across views.

The 4-step Cosmos3-Nano-Sim-Auto receives higher human ratings than Cosmos-Transfer2.5 in all scene-element and cross-view categories (Tab. 18). The largest scene-element gains are for traffic signs (2.45 versus 1.78) and traffic lights (2.24 versus 1.59). It also scores higher than Cosmos3-Nano-Transfer-Auto in four of the five scene-element categories, while cross-view scores differ by at most 0.05 on the 1–3 scale.

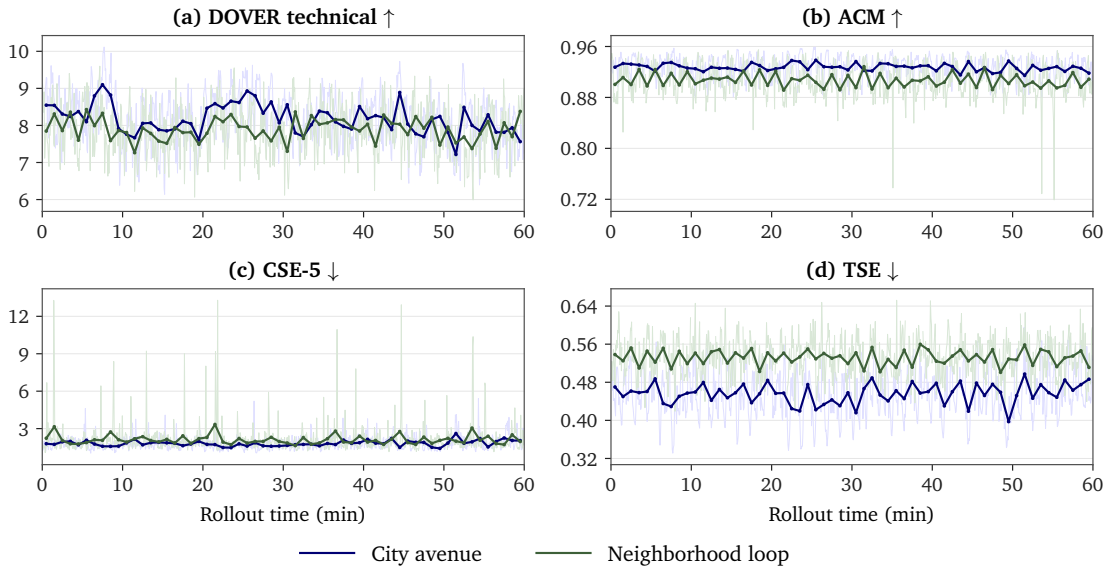


Figure 27: **One-hour rollout stability of Cosmos3-Nano-Sim-Auto.** We generate two driving scenes continuously for 1 hour, cycling their WSM controls and captions with a rolling K/V cache. Video quality (DOVER), temporal consistency (ACM and TSE), and cross-view consistency (CSE-5) show no sustained degradation over the rollout. Faint curves show 4-second-window scores and bold curves show 1-minute averages.

The 4-step causal model trades some video quality for streaming efficiency, consistent with the DOVER results. Road-surface and weather agreement are 87.0% and 81.0%, respectively, also below both comparison models. Nevertheless, the causal distilled variant retains stronger scene-element fidelity and cross-view agreement than the video baseline while enabling 4-step streaming generation.

Long-rollout stability. To assess long-rollout stability, we run Cosmos3-Nano-Sim-Auto continuously for up to 1 hour of generated video, repeatedly cycling the WSM control sequence and associated text prompts while maintaining a rolling K/V cache. Under this setting, the evaluated metrics show no substantial degradation over the 1-hour horizon (Fig. 27). Qualitatively, scene structure and control following remain consistent, although object appearance and physical plausibility can deteriorate over extended rollouts. See App. D.6.1 for additional ablations of training stages, cross-view attention, and attention-sink and K/V cache sizes.

Summary. Cosmos3-Nano-Transfer-Auto and Cosmos3-Nano-Sim-Auto extend Cosmos3-Nano into controllable sensor generation with 1–11 camera views, supporting both RGB-only and joint RGB–LiDAR outputs. The bidirectional model produces high-quality synchronized clips for offline data generation, while its causal distilled counterpart combines strong generation quality with fast, few-step streaming generation. Both models achieve substantial gains in frame and video distribution fidelity over the strong Cosmos-Transfer2.5 baseline, together with better temporal coherence, traffic-object adherence, and human-rated cross-view consistency. They also improve all five LiDAR distribution-fidelity metrics over LidarVerse while jointly generating RGB and LiDAR. These results support high-quality sensor generation for offline synthesis and streaming.

8. Related Work

Interactive world simulation connects learned representations of the physical world with controllable prediction and responsive execution. World foundation models supply reusable visual and multimodal priors (NVIDIA, 2025c, 2026a), while causal generation and distillation make these priors available for interactivity (Huang et al., 2025b; Zheng et al., 2026c). Their use in an ongoing simulation also depends on how observations retain

the consequences of earlier controls and how quickly new inputs can affect generated outputs. We review these connections to position our shared post-training recipe and stateful streaming execution within the broader development of interactive world models.

8.1. Controllable World Models for Physical AI

World models learn how environments evolve so agents can choose actions through imagined futures. Early latent dynamics models used compact rollouts for planning and behavior learning (Ha and Schmidhuber, 2018; Hafner et al., 2019, 2020), and predictive representation learning extends this idea without rendering every predicted state (Assran et al., 2025). Generative world models expanded the interface to visible futures that can be inspected and used as simulated observations (Bruce et al., 2024; NVIDIA, 2025c,a). Multimodal foundations broaden this interface by coupling scene interpretation with the generation of future observations. Cosmos 3 (NVIDIA, 2026a) couples a reasoner tower for multimodal understanding with a generator tower for synthesis, sharing representations of language, observations, and heterogeneous actions. Using these priors in an ongoing simulation additionally requires predictions to depend on available history and to become usable as new controls arrive. Our framework inherits this backbone and its shared action interface, then adapts prediction and execution for incremental observation generation.

Grounding these visual futures in executable interventions made prediction useful for control, from early model-predictive replanning (Finn and Levine, 2017) to policy learning with action-conditioned diffusion (Yang et al., 2024b). Autoregressive token models additionally bring observations, actions, and rewards into one predictive sequence (Wu et al., 2024). Across these formulations, the learned response still depends on how training observations are associated with executable commands. As visual priors broadened, control supervision became a central constraint. Human videos offer varied experience through inferred latent actions that require adaptation to robot commands (Gao et al., 2026a), while measured actions provide direct supervision with coverage limited by available interaction data (Huang et al., 2026c). Cosmos3-Nano-Sim-Bimanual jointly learns camera navigation and robot manipulation from camera, robot, and human interaction data through a shared action interface (Sec. 5).

Camera control exposes the spatial consequences of this grounding: requested motion must produce a compatible change of viewpoint (He et al., 2025; Xu et al., 2024). Gen3C (Ren et al., 2025b) renders a retained 3D cache to guide new views, while global poses, geometric memory, and metric annotations provide other spatial anchors (Nam et al., 2026; Zhang et al., 2026b; Zhu et al., 2026a). Decoupling human and camera motion helps compose controls (Wang et al., 2026a), and sustained navigation makes historical scene evidence important (Robbyant et al., 2026; Gao et al., 2026b; Wang et al., 2026b; Qian et al., 2026). Reference video and rendered scene evidence reduce uncertainty about observed content, while generation still needs to complete unseen regions (InSpatio Team et al., 2026; Song et al., 2026). These conditions constrain what is observed, while inference-time camera guidance (Singh et al., 2026; Ouasfi et al., 2026) and reconstruction feedback (Zhou et al., 2026b) constrain how predictions follow the requested view. Revisits in changing or shared scenes additionally motivate memory of interactions across observers (Xiong et al., 2026; Savva et al., 2026). Our camera-navigation model accepts pose controls chunk by chunk and continues from generated visual history (Sec. 4.1 and 5).

Structured scene conditions move part of this consistency problem outside the learned generator. Geometry and spatial controls constrain layout and appearance (Gao et al., 2024; NVIDIA, 2025b). Explicit 3D worlds retain geometry across viewpoints (Bahmani et al., 2026; Shen et al., 2026; Team HY-World et al., 2026; World Labs, 2026), while spatially grounded image context connects view synthesis with reconstruction (World Labs Team, 2026). An external simulator can maintain evolving physical state and supply it through conditions, leaving the generator to synthesize appearance and unspecified dynamics. When this observation interface expands to multiple sensors, consistency also needs to hold across streams. Shared scenario maps anchor views in a common layout (NVIDIA, 2025a), and staged causal conversion supports mixed camera projections (Hu et al., 2026).

Joint camera–LiDAR models further extend consistency to complementary sensor representations (Waymo, 2026; Cheng et al., 2026). Cosmos3-Nano-Sim-Transfer supports depth, segmentation, edges, and blur, while the AV models use camera-aligned WSM and range-view HD-map controls. Inspired by the decomposed multi-view attention in OmniDreams (Basant et al., 2026), our AV models use time-aligned, deduplicated cross-sensor attention for joint RGB–LiDAR generation (Sec. 7.1.3).

These observation models also support decision-making through inverse dynamics, joint video and action modeling, or planning over imagined futures (Feng et al., 2025; Ye et al., 2026; Zhang et al., 2026a). Policies can issue actions before video denoising completes through asynchronous sampling (Guo et al., 2026a) or intermediate denoising features (Zheng et al., 2026a). Comparing policy outcomes in generated and real environments tests simulation utility (Jiang et al., 2025; Guo et al., 2026b). Our models provide the complementary observation generator, returning RGB or joint RGB–LiDAR outputs in response to supplied controls.

8.2. Causal Video Generation and Few-step Distillation

Moving from generated clips to ongoing interaction changes what temporal information a model can use. Bidirectional video diffusion coordinates observations across a complete target clip, while an interactive video generator must commit to the next prediction from the available history and controls. Diffusion Forcing (Chen et al., 2024) provided a sequence formulation with varying noise levels, supporting different prediction and conditioning patterns. CausVid (Yin et al., 2025) showed how a causal student could inherit a bidirectional video prior through distillation. We first adapt Cosmos3-Nano to each simulation domain, then convert it to causal generation (Sec. 3.1 and 3.2).

Causality makes incremental control possible, but the cost of repeated denoising remains. Consistency models reduce sampling steps by learning predictions that agree along a denoising trajectory (Song et al., 2023; Lu and Song, 2025). Distribution matching instead refines the fast student's output distribution using teacher and auxiliary score estimates (Yin et al., 2024). These objectives can be combined through score regularization, preserving trajectory information while matching the generated distribution (Zheng et al., 2026d). In autoregressive generation, this complementarity also motivates a staged procedure: consistency distillation initializes subsequent training on the student's own rollouts (Lin et al., 2026; Zheng et al., 2026c). Teacher forcing supplies an offline, mode-covering initialization before on-policy distribution matching, whose optimization is sensitive to the starting model (Zheng et al., 2026c). Using the same clean reference history for the teacher and student makes consistency targets conditional on the same task and allows their construction online (Zheng et al., 2026c; Zhao et al., 2026).

A fast causal predictor still encounters a further mismatch: its context history contains its own imperfect outputs at rollout time. Self Forcing (Huang et al., 2025b) addresses this exposure bias by training with autoregressive student-generated histories and a loss over the resulting video. Once training includes generated history, the next issue is how far that supervision extends. Long-context teachers provide a training signal across longer student rollouts (Zhu et al., 2026b; Basant et al., 2026), while forward-and-return cycle consistency offers a constraint without teacher-based distillation (Huang et al., 2026a). Adversarial policy curricula additionally address control coverage by finding high-error trajectories near the behavior distribution and prioritizing them for training (Güzel et al., 2026). Following Causal-rCM (Zheng et al., 2026c), our shared curriculum combines teacher forcing, discrete consistency distillation, and self-forcing distribution matching for action-conditioned and spatially controlled simulation (Sec. 3).

8.3. Efficient Inference for Streaming Video Generation

As streaming generation outgrows a fixed clip, memory must control computation without losing useful scene history. K/V caching reuses unchanged observations, but retaining every entry increases memory and attention cost with the session. Rolling windows and compact histories bound active context (Liu et al., 2026; Zhang

et al., 2025a; Yang et al., 2026), while initial sinks preserve startup information (Xiao et al., 2024; Basant et al., 2026; Zheng et al., 2026c). Matching training and deployment windows can reduce context-length mismatch, although reference and generated histories can still differ in distribution (Shin et al., 2026). Older evidence can be compressed into a fixed-budget representation (Wu et al., 2026) or recovered through retrieval (Yu et al., 2025; Chen et al., 2026b). Retrieving context relevant to the next prediction (Xu et al., 2026a; Qian et al., 2026; Yi et al., 2026) trades storage and selection costs against the information lost through compression. Memory associated with time or interaction events can also support retrieval of scene content after intervening actions (Zheng et al., 2026e; Xiong et al., 2026). We bound active context with configurable history windows and refresh cached features from completed predictions (Sec. 4.1).

Bounded history also makes the recurring workload predictable, enabling compilation and graph replay (Basant et al., 2026). Efficient attention reduces the cost of each denoising step (Ye et al., 2025; Agarwal et al., 2026). Low-precision execution can further reduce arithmetic cost and memory use by representing model tensors with fewer bits. LongLive-2.0 (Chen et al., 2026a) combines NVFP4 inference with K/V cache quantization, applying lower precision to both model computation and retained history. Distributing generation across devices offers additional parallelism at the cost of communication (Jacobs et al., 2023; Huang et al., 2026b). Our runtime combines compilation and attention parallelism with asynchronous decoding where supported by the model's execution dependencies (Sec. 4.2).

Chunking determines when a newly received control can affect the output (Xu et al., 2026b), while generation across multiple streams adds synchronization requirements (Team Odyssey, 2026). We therefore report both throughput and latency using explicit observation-delivery boundaries (Sec. 4.2).

9. Conclusion

We presented a shared post-training framework that turns Cosmos world foundation models into interactive simulators for Physical AI. By combining causal learning, few-step distillation, and stateful streaming, the framework generates observations in response to incoming actions and scene controls. The three Cosmos3-Nano examples span camera navigation and robot manipulation, video transfer, and multi-view AV simulation, supporting both prediction of visual action consequences and observation generation from an external simulator's physical state. Selected configurations reach real-time generation throughput, and the calibrated robot study demonstrates policy feedback through generated observations. The central opportunity is to make simulation a continuing source of experience shaped by an agent's decisions. With sufficiently responsive observations, an agent can act, observe how the scene changes, and use that experience to decide what to do next. We expect interactive world simulation to broaden the environments, appearances, and interventions available for Physical AI training and evaluation. By coupling visual diversity with repeated, controllable interaction, these environments could shorten the cycle of learning, testing, and refining agents, helping developers identify failures before deployment.

A. Contributors and Acknowledgments

Contributors in each group are listed alphabetically by last name.

A.1. Contributors

Supervision

Chen-Hsuan Lin, Ming-Yu Liu

Model Architecture

Francesco Ferroni, Zhengfei Kuang, George Kurian, Xiang Li, Chen-Hsuan Lin, Qianli Ma, Trung Pham, Jiaxiang Tang, Zhonghao Wang, Kaiwen Zheng

Post-training Recipe

Xiao Fu, DeLesley Hutchins, Chen-Hsuan Lin, Qianli Ma, Mustafa Munir, Jiaxiang Tang, Kaiwen Zheng

Cosmos3-Nano-Sim-Bimanual

Data Preparation

Xiao Fu, Xin Kong, Aleksandr Korovko, Zhengfei Kuang, Zhaoshuo Li, Chen-Hsuan Lin, Qianli Ma, Jiaxiang Tang, Shitao Tang, Shijie Wang, Mengyao Xu

Post-training

Xiao Fu, Zhengfei Kuang, Chen-Hsuan Lin, Jiaxiang Tang, Shijie Wang

Inference Optimization

Anis Ahmad, Jesse Archer, Maciej Bala, Aarti Basant, Taras Bolzhelarskyi, Yong He, William Lew, Martin Ding Ma, Pengcuo Zeren, Yuliya Zhautouskaya

Cosmos3-Nano-Sim-Transfer

Data Preparation

Francesco Ferroni, Qianli Ma, Zhonghao Wang

Post-training

Zhonghao Wang

Inference Optimization

Maciej Bala, Pengcuo Zeren, Yuliya Zhautouskaya

Cosmos3-Nano-Sim-Auto

Data Preparation

Francesco Ferroni, Justin Leung, Xiang Li, Qianli Ma, Trung Pham, Andrew Z. Wang, Mengyao Xu, Andrew Zhu

Post-training

Francesco Ferroni, Xiang Li, Qianli Ma, Trung Pham, Zhonghao Wang

Inference Optimization

Jesse Archer, Maciej Bala, Aarti Basant, Vivek Goel, George Kurian, Janarbek Matai, Jonathan McCaffrey, Yuliya Zhautouskaya

Infrastructure

Dan Blick, Liang Feng, Aryaman Gupta, Ali Hassani, Nathan Hayes-Roth, DeLesley Hutchins, Suneel Indupuru, Ashna Khetan, Hui Kuang, George Kurian, Chen-Hsuan Lin, Xiangyu Lu, Qianli Ma, Miguel Martin, Jiaxiang Tang, Sergei Vasilev, Thomas Volk, Hao Wang, Jing Zhang

Benchmarks

Jon Allen, Malik Aqeel Anwar, Dan Blick, Tae Eun Choe, Yuzhu Dong, Francesco Ferroni, Xiao Fu, Aryaman Gupta, Pooya Jannaty, Jingyi Jin, Janmesh Kalra, Zhengfei Kuang, Jayjun Lee, Chen-Hsuan Lin, Yuhang Lu, Qianli Ma, Ashkan Mirzaei, David Pankratz, Trung Pham, Marilyn Reeb, Jiaxiang Tang, Tolou Tavakkoli, Alexander Temirov, Zhonghao Wang, David Wehr, Shumian Xin, Sophia Zalewski, Cindy Zha, Andrew Zhu

Other Contributions

Piotr Barejko, Henry Che, Jin Chen, Yukang Chen, Nat Duca, Mohit Gambhir, Rony Goldenthal, Michael Honke, Shengyu Huang, Wei Huang, Jean-Francois Lafleche, Aditya Mahajan, Qing Miao, Sangmin Oh, Gavriel State, Yue Tang, Hweimi Tsou, Yao Xu, Shuai Yang, Rick Zhuang

A.2. Acknowledgments

Yogesh Balaji, Julius Berner, Tiffany Cai, Yan Chang, Chao-Yeh Chen, Tiffany Chen, Ruben D'Sa, Riccardo de Lutio, Jiaojiao Fan, Jun Gao, Shenyuan Gao, Lanxiang Hu, Jiahui Huang, Christian Laforte, Laura Leal-Taixe, Hongyu Li, Tsung-Yi Lin, Kaichun Mo, Seungjun Nah, Anjul Patney, Hesam Rabeti, John Shao, Wei-Cheng Tseng, Jialiang Wang, Hongchi Xia, Chenchao Xu, Yajie Yan, Liangkai Zhang

B. Cosmos3-Nano-Sim-Bimanual: Additional Details

B.1. Data and Caption Construction

Captions provide scene information appropriate to each control task. For camera navigation, the caption describes the environment and may include content outside the initial view. It retains object appearance, spatial relationships, and independently moving subjects, but excludes camera motion and apparent scene motion caused by viewpoint changes. For robotics and human interaction, the caption excludes the controlled agent's actions and their effects on the scene. The modality-specific rewrite removes the robot's motion and changing pose in AgiBot/YAM (Bu et al., 2025; I2RT Robotics, n.d.), or the camera wearer's actions in Mecka (Mecka AI, n.d.). However, an object's eventual position or state can still reveal the outcome of a future manipulation.

To remove these outcome descriptions, we apply an additional text rewrite to the AgiBot, YAM, and Mecka annotations. We extract the first temporal segment, remove its timestamp, and rewrite it as an initial-state description. The rewriting model receives only this segment, without later segments, the structured scene breakdown, or the video. The prompt preserves the setting, appearance, initial configurations, existing contacts, and independent background activity. It removes camera motion, controlled-agent actions, and their consequences, and prohibits inferring an initial state from a removed action or outcome. For example, an object already held at the start may be described, but its subsequent displacement or placement is excluded. Because the source caption was generated from video, this procedure does not establish that every retained fact is visible in the initial frame.

The video generator receives the final caption text without the structured scene breakdown. All four training stages use the same final captions, matched to the covered video samples by source identifier. Camera videos use the camera captioner's scene descriptions, while Behavior-1K (Li et al., 2023) uses separately supplied task descriptions.

Robot evaluation captions are constructed for the exact temporal window paired with each benchmark video and action sequence. For the 100 AgiBot-GEAR and 232 AgiBot-World clips in Sec. 5.5, descriptions are generated from video frames sampled at 2 FPS. The training first-segment rewriting prompt then produces the final initial-state captions, with nonempty descriptions for all 332 clips.

B.2. Training Configuration

Stage 0 initializes the bidirectional teacher from the public `nvidia/Cosmos3-Nano` release. Training predicts RGB targets conditioned on supplied actions. All four stages use the same forward-dynamics conditioning rule: the first RGB frame and the complete aligned action sequence are provided as clean conditions, while the remaining RGB frames are prediction targets. Stage 0 processes the targets bidirectionally; TF and TF-dCM condition each causal chunk on recorded preceding frames, whereas SF-DMD conditions subsequent chunks on the student's own completed predictions. The causal stages predict chunks of 2 latent frames and retain the complete available history without sink tokens.

Noise-time sampling. The following noise-time sampling rules and resolution-dependent shifts apply to all three models throughout bidirectional adaptation, causal training, and distillation. Following Cosmos 3 (NVIDIA, 2026a), we use mode sampling with scale 1.29 for bidirectional adaptation and TF, as in Waver (Zhang et al., 2025b). We draw $r \sim \mathcal{U}(0,1)$ and transform it as

$$u = 1 - r - 1.29\left[\cos^2\left(\frac{\pi r}{2}\right) - 1 + r\right].$$

We then shift the sampled noise time using

$$\tau = S_\kappa(u) = \frac{\kappa u}{1 + (\kappa - 1)u},$$

The shift factors are $\kappa = 3$ at 256p, 5 at 480p, and 10 at 720p. TF-dCM instead draws u uniformly from $(1/48, 1)$ and applies this shift to the pair $(u, u - 1/48)$. SF-DMD draws score times uniformly from $(0, 1)$ before applying the same shift.

All stages use a per-GPU microbatch size of one clip and a gradient-accumulation factor of 1. The $2 : 2 : 1$ weights distribute camera, robot, and human clips across training GPUs, with minor differences caused by rounding. Each GPU processes one clip using the action-processing pipeline for its source, and gradients from all GPUs are combined before one optimizer step.

The bidirectional and teacher-forcing (TF) stages use FusedAdam with a base learning rate of 10^{-5}. The bidirectional stage uses betas $(0.9, 0.99)$ and weight decay 0.05, while the TF stage uses betas $(0.9, 0.999)$ and weight decay 0.01. The action-to-model projection and action-modality embedding use 5 times the base learning rate.

The TF-dCM stage uses FusedAdam with a learning rate of 2×10^{-6}, betas $(0, 0.999)$, and weight decay 0.01. Its teacher uses classifier-free guidance (CFG) with a scale of 3. The reference branch replaces the scene caption with an empty string while retaining the task prompt, supplied actions, and recorded sensor history. The SF-DMD stage uses FusedAdam with student and auxiliary score-model learning rates of 10^{-6} and 2×10^{-7}, respectively, betas $(0, 0.999)$, and weight decay 0.01. Its teacher does not use CFG. Training alternates 1 student update with 5 auxiliary score-model updates and uses only 480p clips of 901 RGB frames, including the initial frame.

Model computation and gradient reduction use BF16, while master weights, optimizer states, and exponential moving average (EMA) weights are maintained in FP32 under FSDP (Zhao et al., 2023). All reported final checkpoints use the corresponding EMA weights. The reported SF-DMD model uses 4 denoising steps at inference.

B.3. Decoder Studies

Additional decoder studies use camera-navigation workloads separate from the final Cosmos3-Nano-Sim-Bimanual benchmarks. In an 80-sample quality evaluation, replacing the Wan 2.2 VAE with TAEHV Super reduces the PAI-Bench-G score from 71.8 to 70.9. Separate 480p GB200 timing tests report 1.42–$1.91\times$ speedups with synchronous decoding and 1.06–$1.18\times$ with asynchronous decoding. These quality and timing workloads differ from each other and from the final benchmarks, so they do not establish a matched quality–speed tradeoff for the final model. The reported GB300 decoder runtime comparison appears in Tab. 3.

B.4. Evaluation Metrics

Video quality uses the PAI-Bench-G (Zhou et al., 2026a) overall score, and quality drift is its absolute start/end difference as defined in Sec. 5.5. The metrics below complement these scores with reference-frame agreement for robot manipulation and camera-trajectory agreement for camera navigation.

Reference-frame agreement. Robot manipulation evaluation compares generated and reference frames using peak signal-to-noise ratio (PSNR, dB), structural similarity (SSIM) (Wang et al., 2004), and learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018). PSNR measures pixel reconstruction error on a logarithmic scale, SSIM measures structural image similarity, and LPIPS measures learned perceptual feature distance. Higher PSNR and SSIM and lower LPIPS indicate closer reference-frame agreement. Frame synchronization, resizing, image range, the LPIPS backbone, and valid frame coverage are fixed. These metrics are action-following proxies and do not independently validate contact or task outcomes.

Camera trajectory alignment. Camera navigation evaluation compares ViPE-estimated poses (Huang et al., 2025a) from generated videos with supplied trajectories after Sim(3) Umeyama alignment (Umeyama, 1991). Absolute translation error (ATE) is the L2 distance between aligned inferred and supplied translations. Relative

Table B.1: **Bidirectional teacher training clip length.** Cosmos3-Nano-Sim-Bimanual teachers are trained with clips up to 961 frames or fixed 253-frame clips. Both configurations are evaluated at iteration 3,500 on 961-frame Sana-WM-Bench camera-easy sequences.

Training clips	Quality↑	ATE↓	RTE↓	RRE↓
Long, up to 961 frames	**75.165**	**1.756**	**0.029**	**0.286**
Short, fixed 253 frames	74.113	2.007	0.089	0.316

Table B.2: **Full fine-tuning versus LoRA for bidirectional teacher adaptation.** Cosmos3-Nano-Sim-Bimanual teachers use the long-clip camera recipe and are evaluated at iteration 2,000 on 961-frame Sana-WM-Bench camera-easy sequences. LoRA uses rank 16 and alpha 32 with EMA disabled, while full fine-tuning retains EMA.

Adaptation	Quality↑	ATE↓	RTE↓	RRE↓
Full fine-tuning	**77.086**	**1.594**	**0.018**	**0.081**
LoRA, rank 16	74.979	1.954	0.048	0.224

translation error (RTE) is the L2 distance between inferred and supplied relative translations. Relative rotation error (RRE) is the angular distance between their relative rotation matrices. Lower values indicate closer trajectory agreement after translation, rotation, and scale alignment. The relative pose interval, alignment granularity, translation units, and angular units are fixed across models. These scores do not establish accuracy at absolute metric scale or independently validate physical outcomes.

B.5. Training Ablations and Development Studies

We study training choices for Cosmos3-Nano-Sim-Bimanual on the Sana-WM-Bench camera-easy subset (Zhu et al., 2026a). The studies cover teacher preparation, distillation configuration, and alternative training recipes. Comparisons within each table use the same evaluation cases and protocol. Checkpoints, datasets, and training recipes can differ across studies, so absolute scores are not directly comparable across tables. Quality denotes the PAI-Bench-G overall score, while ATE, RTE, and RRE measure camera trajectory agreement as described in App. B.4. Higher quality and lower trajectory errors are preferred. The findings characterize the tested configurations, including development recipes that differ from the final model.

B.5.1. Teacher Preparation

Training clip length. We compare bidirectional teacher adaptation using clips up to 961 frames with adaptation using fixed 253-frame clips. Both configurations are evaluated at iteration 3,500 on 961-frame camera navigation sequences. Long-clip training increases quality by 1.052 points and reduces ATE, RTE, and RRE (Tab. B.1). RTE falls from 0.089 to 0.029. The long-clip configuration improves all four reported metrics at this checkpoint and evaluation horizon.

Full fine-tuning versus LoRA. We compare full fine-tuning with LoRA (Hu et al., 2022) using the long-clip camera recipe. Both bidirectional checkpoints are evaluated at iteration 2,000 over 961-frame camera navigation sequences (Tab. B.2). LoRA uses rank 16 and alpha 32 and trains only adapter parameters, while full fine-tuning updates the configured model parameter set. Full fine-tuning has higher visual quality by 2.107 points and lower trajectory errors. This result favors the full fine-tuning configuration over the tested rank-16 LoRA configuration.

B.5.2. Distillation Configuration

Teacher guidance. We vary classifier-free guidance (CFG) (Ho and Salimans, 2021) in the distillation teacher separately for TF-dCM and SF-DMD. CFG controls the teacher predictions used to train the student. The TF-dCM and SF-DMD comparisons use separate recipes and evaluation checkpoints.

Table B.3: **TF-dCM teacher guidance.** Cosmos3-Nano-Sim-Bimanual checkpoints use the same short-video training recipe and initialization, with causal teacher CFG scales 1, 3, and 5. All checkpoints are evaluated at iteration 2,000 on 961-frame Sana-WM-Bench camera-easy sequences.

Teacher guidance	Quality↑	ATE↓	RTE↓	RRE↓
1	78.237	0.259	0.018	0.694
3	**78.431**	0.244	0.014	0.367
5	78.287	**0.225**	**0.012**	**0.145**

Table B.4: **SF-DMD teacher guidance.** Cosmos3-Nano-Sim-Bimanual checkpoints use bidirectional teacher CFG scales 1 (no CFG) and 6. Both are evaluated at iteration 1,000 on 961-frame Sana-WM-Bench camera-easy sequences.

Teacher guidance	Quality↑	ATE↓	RTE↓	RRE↓
1 (no CFG)	**76.344**	**0.943**	**0.012**	**0.149**
6	74.587	1.233	0.015	0.282

Table B.5: **SF-DMD rollout selection and denoising schedule.** Cosmos3-Nano-Sim-Bimanual checkpoints combine cycling or maximum rollout selection with 2-step or 4-step schedules. All four are evaluated at iteration 3,500 on 480p, 961-frame Sana-WM-Bench camera-easy sequences.

Selection	Schedule	Quality↑	ATE↓	RTE↓	RRE↓
Cycle	2	74.540	**1.012**	0.015	0.082
Max	2	75.024	1.039	**0.012**	0.201
Cycle	4	73.933	1.138	0.015	0.095
Max	4	**75.169**	1.474	0.016	**0.064**

The TF-dCM comparison uses the same short-video training recipe and initialization, with causal teacher guidance scales 1, 3, and 5. All checkpoints are evaluated at iteration 2,000 over 961 frames (Tab. B.3). Guidance 3 has the highest recorded quality, exceeding scales 1 and 5 by 0.194 and 0.144 points. Guidance 5 has the lowest ATE, RTE, and RRE. We select guidance 3 for the production TF-dCM stage to prioritize visual quality.

The SF-DMD comparison uses bidirectional teacher guidance scales 1 and 6. Both checkpoints are evaluated at iteration 1,000 over 961 frames (Tab. B.4). Without teacher CFG, quality is higher by 1.757 points and ATE, RTE, and RRE are lower. We use an unguided teacher for the production SF-DMD stage.

Rollout selection and denoising step schedule. We compare cycling and maximum rollout selection with 2-step and 4-step denoising schedules in the camera SF-DMD recipe. Cycling varies the rollout depth up to the configured limits, while maximum selection always uses those limits. The four checkpoints are evaluated at iteration 3,500 on 480p, 961-frame camera navigation sequences (Tab. B.5). With the 2-step schedule, maximum selection increases quality by 0.484 points and reduces RTE, but increases ATE and RRE. Maximum selection with the 4-step schedule has the highest quality and lowest RRE, but also the highest ATE and RTE. No configuration dominates every measure. We adopt maximum selection with the 4-step schedule in the production model to prioritize visual quality and rotational agreement.

B.5.3. Alternative Training Recipes

Mixed bidirectional and causal training. We compare a single model trained with alternating bidirectional and causal updates against models trained in only one attention mode. The bidirectional:causal update patterns are $1:1$, $1:0$, and $0:1$, with shared initialization, chunks of 4 latent frames, and context parallelism of degree

Table B.6: **Mixed versus single attention-mode training.** Cosmos3-Nano-Sim-Bimanual models share initialization, chunks of 4 latent frames, and context parallelism of degree 4. All are evaluated at iteration 4,500 on 961-frame Sana-WM-Bench camera-easy sequences. Mixed training alternates bidirectional and causal updates at $1:1$, and its two inference rows use the same checkpoint. Each inference group compares mixed training with training in that attention mode alone.

Training	Quality↑	ATE↓	RTE↓	RRE↓
Bidirectional inference				
Bidirectional only	75.792	**1.245**	**0.016**	**0.079**
Mixed 1 : 1	**76.055**	1.496	0.019	0.081
Causal inference				
Causal only	**68.263**	**1.479**	**0.025**	**0.555**
Mixed 1 : 1	67.580	1.521	0.035	0.693

4. All three models are evaluated at iteration 4,500 on 961-frame camera navigation sequences (Tab. B.6). The same mixed-training checkpoint is evaluated in both bidirectional and causal inference modes.

Under bidirectional inference, mixed training increases quality from 75.792 to 76.055 but increases ATE, RTE, and RRE. Under causal inference, the causal-only model has higher quality and lower errors on all three trajectory measures. These results show a quality–control tradeoff under bidirectional inference and favor causal-only training under causal inference at the evaluated checkpoints.

Equal total iterations give the mixed model approximately half as many updates per mode as each specialized model. The comparison therefore combines the attention-mode choice with different per-mode training exposure. The production curriculum first adapts a bidirectional teacher and then fine-tunes it with causal attention. The study does not directly compare alternating training with this sequential curriculum.

Streaming long tuning. Streaming long tuning (SLT) carries generated state through successive training windows with a bounded K/V history (Yang et al., 2026; Chen et al., 2026a). It enables training on long student rollouts using a short-context teacher. We compare SLT with standard long-rollout SF-DMD using maximum selection with 4 denoising steps on camera navigation. SLT uses a short-clip teacher, whereas standard SF-DMD uses a long-context teacher. The comparison therefore evaluates complete recipes rather than isolating the effect of streaming training.

Both recipes reach their highest observed camera-easy quality at iteration 2,000 (Tab. B.7). Standard SF-DMD has 6.600 points higher quality than SLT and lower ATE and RRE. SLT has lower RTE, so neither recipe is best on every reported metric.

We observe repeating visual patterns in the inspected SLT rollouts. The SLT experiments use captions describing events throughout the source video. This historical conditioning differs from the initial-state captions described for robot and human interaction in Sec. 5.2. A mismatch between these whole-video event descriptions and the short-context teacher's local training windows may contribute to repetition, but this explanation was not tested separately. The production model uses standard SF-DMD with maximum selection, 4 denoising steps, long rollouts, and a long-context teacher.

Table B.7: **Standard SF-DMD versus streaming long tuning.** The Cosmos3-Nano-Sim-Bimanual camera recipes are evaluated on Sana-WM-Bench camera-easy. Each row reports its highest observed quality checkpoint at iteration 2,000, with trajectory metrics from the same checkpoint. Standard SF-DMD uses a long-context teacher, while SLT uses a short-clip teacher.

Training recipe	Quality↑	ATE↓	RTE↓	RRE↓
SF-DMD	**77.167**	**1.434**	0.017	**0.088**
SLT	70.567	1.539	**0.016**	0.164

C. Cosmos3-Nano-Sim-Transfer: Additional Details

This appendix supplements the Cosmos3-Nano-Sim-Transfer methods and results in Sec. 6. After dataset accounting and sampling details, we describe runtime profiling, supplementary benchmark analysis, and Isaac Lab evaluation. We conclude with training-stage comparisons and failure cases.

C.1. Data and Sampling Details

Dataset retention. Tab. C.1 breaks down the exclusions from the eligibility index described in Sec. 6.2. Each rejected source entry is assigned to one category, so the four exclusion counts partition the 156,946 rejected entries. The table counts source entries by metadata and caption-window eligibility.

Table C.1: **Cosmos3-Nano-Sim-Transfer dataset retention.** The index retains one complete eligible caption window per indexed video. The four exclusion counts sum to the total rejected population.

Population or exclusion	Count
Source RGB video entries	2,153,548
Indexed eligible videos	1,996,602
Total excluded entries	156,946
Aspect-ratio distortion above 3%	114,453
Source FPS outside 5–60	10,870
Invalid caption envelope	145
No complete eligible caption window	31,478

The three causal stages train at 480p. When averaging losses across latent sequences, bidirectional adaptation includes zero-loss control sequences, whereas TF and TF-dCM exclude sequences that contain only conditioning inputs.

Teacher guidance. The distillation stages in Sec. 6.3 construct supervision from positive and negative teacher predictions. Both branches receive identical spatial controls and include duration, FPS, and resolution metadata in their text conditioning. The positive branch uses the sample's caption condition after text dropout, and the negative branch uses the canonical Cosmos3-Nano-Sim-Transfer negative caption.

Student sampling and training history. The SDE sampler evaluates fixed diffusion times $[1, 15/16, 5/6, 5/8]$, corresponding to a shift-5 schedule, before reaching the clean endpoint. Each nonterminal transition draws fresh Gaussian noise for the next evaluation. Student evaluation uses guidance 1 for both language and controls. During Stage 3 training, the K/V cache uses a window of 51 latent frames and a 1-latent-frame attention sink. Evaluation history windows follow the workload-specific settings in Sec. 6.5.

C.2. Runtime Profiling Details

Offline runtime configuration. The compilation results in Tab. 12 use four combinations of history policy and execution mode. Tab. C.2 reports the software, worker layout, and callback storage measurements for that

comparison. Each configuration runs two measured videos in each of three processes, with a warmup before every measured video. Decoded-output timing covers latent indices 51–149, and component timing pools 588 measurements from six videos. The distributions pool chunk measurements from all six videos.

Table C.2: **Tested offline configuration and storage for Cosmos3-Nano-Sim-Transfer.** The model, video geometry, and history policies are specified with the performance results in Tab. 12. RTF is compute time divided by generated video time over the decoded-output measurement window. Memory entries average the PyTorch allocated and reserved storage recorded at callbacks during these 597-frame generations.

Field	Configuration
Model / conditioning	SF-DMD iteration 1,500; single-view depth; no RGB prefix; seed 42
GPU / parallelism	1 GB200 per worker; 4 independent workers on 1 node; CP=CFGP=1; batch size 1
Optimization	Eager; or `torch.compile` (`compiled_region=language`) with block CUDA Graphs
Decoder / preprocessing	Wan2.2 streaming VAE; PyAV input decoding; `direct_resize`; offline whole-video control preparation
Runtime	Torch `2.13.0a0+9186a08b2c.nv26.07`; CUDA 13.3
Timing per configuration	3 processes × 2 measured videos, each preceded by warmup; 594 matched output chunks / 2,376 RGB frames

History / execution	RTF↓	Callback mean allocated / reserved (GiB)
Full / eager	4.3603	50.4828 / 60.0859
Full / compiled	2.6300	67.3645 / 72.6318
Sink + recent / eager	3.3185	39.5165 / 47.7598
Sink + recent / compiled	1.8926	46.1787 / 50.7305

We also compare the depth inferred from generated and reference RGB over the beginning of each video. PAIBench-C v1.4.0 uses VideoDepthAnything-small (Chen et al., 2025) to estimate both depth sequences. Across 16 videos per configuration, the first 121 RGB frames (31 latent positions) give an SI-RMSE of 5.2124 for all four configurations. This measurement describes depth agreement over that shared prefix.

Component timing and startup. To explain the generation costs behind the decoded-output intervals in Tab. 12, Tab. C.3 separates model computation, control insertion, and decoding. For each chunk, model-only time is the sum of denoising and clean-history refresh. We compute its p50 and p95 from these per-chunk sums. Control K/V insertion is timed separately. GPU component measurements use CUDA events on the current stream and include waits for their dependencies. The VAE measurement ends before output clamping and device-to-host transfer, while the broader denoise-to-history-ready interval includes decoding and device-to-host waits. Because these timing intervals overlap, their sum counts some work more than once.

Preparation and process startup use elapsed wall time. Startup begins before framework import and model setup and ends when the first decoded output reaches CPU memory. This measurement is reported separately from warmed generation, and compiler disk caches are reused across launches. The startup percentiles summarize three process launches per configuration.

Table C.3: **Offline component, preparation, and startup times for Cosmos3-Nano-Sim-Transfer.** Each entry gives the median (p50) and 95th percentile (p95) for the workload in Tab. C.2. F-E/F-C denote full-history eager/compiled execution, and S-E/S-C denote sink-plus-recent eager/compiled execution. Panel (a) reports ms/chunk and pools observations from six measured videos per configuration. Each component has 588 valid measurements from 588 attempts. Panel (b) reports offline preparation in ms/video over 6/6 valid/attempted videos and startup in ms/process over 3/3 process launches.

(a) Warmed generation components

Component	F-E	F-C	S-E	S-C
Denoising / guidance	337.13 / 462.66	198.36 / 261.76	255.61 / 301.77	127.09 / 146.81
Clean-history refresh	87.91 / 123.48	52.37 / 67.86	67.10 / 79.30	35.34 / 37.35
Model-only chunk	425.85 / 582.01	250.58 / 330.19	322.60 / 383.47	163.14 / 182.20
Control K/V insertion	87.31 / 122.47	51.89 / 68.38	66.92 / 79.03	33.60 / 45.85
Streaming VAE decoding	46.89 / 50.84	42.86 / 43.03	45.07 / 45.84	42.32 / 42.50
Denoise to history ready (includes decode)	473.01 / 632.13	293.80 / 373.42	367.90 / 429.44	205.79 / 224.90

(b) Offline preparation and process startup

History / execution	Whole-video preparation (ms/video)	Entry to first decoded output (ms/process)
Full / eager	5302.11 / 5399.97	26619.47 / 61970.27
Full / compiled	5266.26 / 5367.48	36489.38 / 78128.99
Sink + recent / eager	5187.23 / 5330.32	62803.09 / 66504.84
Sink + recent / compiled	5262.77 / 5328.87	36555.66 / 78673.18

C.3. Supplementary Benchmark Analysis

Benchmark media alignment. The full-benchmark comparison in Sec. 6.5 includes two media corrections to keep RGB and controls aligned. For 5 source videos, we apply the same center crop to both streams. For 17 OpenDV sources (Yang et al., 2024a), we remux timestamps in 34 depth and segmentation files while preserving their decoded frames. The corrected files provide matching spatial and temporal positions for evaluation.

Depth error normalization. The TransferBench depth metric in Tab. 8 compares VideoDepthAnything (Chen et al., 2025) estimates from generated and reference RGB. For each frame, $\widehat{d}$ and d denote these two depth maps, and $\mathcal{P}_{\text{valid}}$ contains pixels with positive depth in both maps. When this set is nonempty, we compute

$$s = \frac{\text{median}(d)}{\text{median}(\widehat{d})}, \qquad \text{SI-RMSE} = \sqrt{\frac{1}{|\mathcal{P}_{\text{valid}}|} \sum_{\mathbf{p} \in \mathcal{P}_{\text{valid}}} \text{clip}\Big(s\widehat{d}(\mathbf{p}) - d(\mathbf{p}), -10, 10\Big)^2}. \tag{C.1}$$

Both medians are computed over $\mathcal{P}_{\text{valid}}$, and the residual is clipped to $[-10, 10]$ before squaring. The evaluator averages the frame scores over the clip, assigning zero to frames with no valid pixels. The VisDex analysis below instead uses an uncapped error against simulator ground truth.

Full-benchmark paired uncertainty. Tab. C.4 supplements the full TransferBench results in Sec. 6.5 with all nine paired TF-dCM-minus-TF metric differences and their source-bootstrap intervals. The analysis uses the benchmark's 600 paired source videos, with 200 sources each from AgiBot, Ego-Exo4D, and OpenDV. The differences include each stage's deployed sampler and guidance settings.

We use each source video as the unit of the paired bootstrap (Efron, 1979). For video-level metrics, we average each source's four control variants, while control-specific metrics use the score for the corresponding modality. Each replicate samples 200 source IDs with replacement within each domain and keeps both stages

Table C.4: **Paired source-bootstrap intervals for TF-dCM relative to TF.** The selected EMA checkpoints are TF iteration 6,000 and TF-dCM iteration 2,250. Differences use each metric's original scale, and arrows indicate the preferred direction of the underlying metric. Every metric has 600 paired source scores.

Metric	TF-dCM − TF	95% interval
DOVER ↑	-0.170816	$[-0.236949, -0.106727]$
Segmentation IoU ↑	$+0.012182$	$[+0.003538, +0.020914]$
Blur SSIM ↑	-0.002144	$[-0.003209, -0.001059]$
Edge F1 ↑	$+0.007809$	$[+0.006058, +0.009492]$
Depth SI-RMSE ↓	-0.032128	$[-0.059830, -0.003746]$
Quality drift ↓	$+0.003466$	$[+0.002943, +0.003996]$
Sampson error ↓	$+0.011013$	$[+0.010184, +0.011842]$
Prompt ↑	-0.033750	$[-0.088333, +0.021250]$
Quality ↑	$+0.038750$	$[-0.039583, +0.118760]$

and all control variants together for every selected source. This preserves the pairing between stages and the relationship between outputs generated from the same video. We give the three domain means equal weight, then take the 2.5th and 97.5th percentiles of 10,000 replicates using linear interpolation, NumPy PCG64, and seed 20261019. The resulting 95% intervals describe uncertainty from sampling source videos with the recorded generations and judge outputs held fixed. They do not include variability from changing generation seeds or repeating judge evaluations. Each is a marginal interval for one metric, with no adjustment for simultaneous coverage across metrics.

Long-rollout quality. To examine rendering quality over longer sequences, we evaluate the final student and GA reference on all 160 VisDex episodes under both depth and segmentation control. Each model produces 320 scored videos of 600 frames, using the same source episodes, controls, and prompts. We compare depth and segmentation extracted from generated RGB with references derived from the simulator's controls.

For depth, we align the predicted and reference medians separately in each frame, then compute the uncapped RMSE in normalized control-depth units over nonbackground, unsaturated pixels. For segmentation, we compare Grounded-SAM2 masks extracted from generated frames with control masks labeled as robot, object, and table, counting missing detections as errors. These class labels come from visual inspection of source-frame control colors, not exported simulator semantic IDs. These spatial metrics use a different reference protocol from the 121-frame TransferBench evaluation. We therefore interpret these values within the VisDex comparison.

Tab. C.5 shows that the student scores higher in segmentation IoU and Quality on these long rollouts. The GA reference retains better DOVER, depth error, drift, Sampson error, and Prompt scores. Together with the runtime measurements in Tab. 11, these results describe the quality and speed of the two deployed pipelines.

Each pipeline retains its own sampling and history settings for this comparison. GA uses 50-step UniPC with text guidance 3 and diffusion caching, while SF-DMD uses 4-step SDE with text guidance 1 and a causal K/V cache. The control guidance also differs: GA uses 1.5 for depth and 2 for segmentation, while SF-DMD uses 1 for both modalities.

Table C.5: **Matched 600-frame VisDex quality.** Metrics cover all 160 episodes under depth and segmentation: 320 videos per model, with 160 for each control metric. Both models use identical source episodes, controls, and prompts. Depth uses simulator control values, and segmentation uses visually labeled control masks.

Metric	Cosmos 3 GA reference	Cosmos3-Nano-Sim-Transfer
DOVER ↑	11.5036	9.1682
Quality ↑	2.1469	2.3500
Prompt ↑	5.2875	5.2250
Segmentation IoU ↑	0.5517	0.5727
Direct-GT depth SI-RMSE ↓	0.085053	0.087610
Drift ↓	0.014210	0.014365
Sampson ↓	0.081120	0.150449

History window and attention sink ablation. We select a K/V window of 30 latent frames and 3 sink frames for long rollouts based on the six configurations in Tab. C.6. The sweep uses a development SF-DMD EMA checkpoint at iteration 1,250 on 160 depth-conditioned VisDex episodes. Each configuration generates 593 frames at 30 FPS and 640×640, with a one-latent-frame chunk, four-step SDE sampling, seed 42, and identical rewritten prompts. Only the window size W and sink size S change. Both sizes count latent frames, with W including the S pinned initial frames and the current frame. Thus, $W = 30, S = 3$ retains 3 initial frames and 26 recent history frames alongside the current frame.

With 3 sink frames, increasing the window from 10 to 30 raises DOVER from 9.652 to 10.056, Prompt from 5.412 to 5.444, and Quality from 2.225 to 2.438. Expanding the window to 50 slightly raises Quality to 2.450 but lowers DOVER to 9.933 and Prompt to 5.425. We select window 30 for the highest DOVER in this sweep and nearly the same Quality as window 50, using 20 fewer latent-frame positions.

At window 30, increasing the sink from 3 to 6 raises Prompt from 5.444 to 5.556 but lowers DOVER from 10.056 to 9.997 and Quality from 2.438 to 2.419. We retain 3 sink frames for higher DOVER and Quality, accepting the lower Prompt grade while allocating more of the fixed window to recent observations.

Table C.6: **History window and attention sink ablation for long rollouts.** Each row aggregates 160 depth-conditioned VisDex videos from the same iteration-1,250 SF-DMD checkpoint. W and S denote the total K/V window and pinned sink, measured in latent frames. Gemini 3.1 Pro provides Prompt and Quality grades on a 0–10 scale using the original captions. Bold marks the selected setting.

W	S	DOVER↑	Prompt↑	Quality↑
6	3	9.550	5.487	2.237
10	3	9.652	5.412	2.225
30	**3**	**10.056**	**5.444**	**2.438**
50	3	9.933	5.425	2.450
10	6	9.615	5.475	2.194
30	6	9.997	5.556	2.419

C.4. Isaac Lab Evaluation Details

Initial states and seed assignments. The closed-loop results in Tab. 9 use the runtime settings in Tab. C.7 and fixed seed assignments for the 2 batches of 16 environments. The first batch uses simulator seed 106931, policy seed 0, and renderer seeds 0–15 in environment order. The second batch uses simulator seed 106932, policy seed 1, and renderer seeds 16–31. All appearance conditions reuse these initial states and seed assignments, keeping the starting conditions fixed as the rendered appearance changes. We freeze depth inputs and stop policy queries for terminated environments, discarding padded generated RGB.

Table C.7: **Isaac Lab apple-task runtime configuration.** These hardware, software, and execution settings accompany the closed-loop schedule in Sec. 6.5.

Field	Configuration
Simulator	Isaac Lab 3.0.0b2 (Mittal et al., 2025); Isaac Sim 6.0.0.1 (NVIDIA, 2026e) on RTX PRO 4000
Renderer and policy hardware	GB300
Renderer software	Cosmos Framework 1.2.2; Python 3.13.16; PyTorch 2.10.0+cu130; CUDA 13.0; torchvision 0.25.0+cu130; transformers 4.57.6
Compilation	`torch.compile` and CUDA Graphs enabled, `compiled_region=language`
cuDNN	`benchmark=False`, `deterministic=True`, applied after model loading
Rectified flow shift	5

Table C.8: **Simulator-state success conditions for apple pick-and-place.** The evaluator requires placement, support, contact, settling, and hand release simultaneously. Distances and heights are in meters.

Condition	Threshold
Horizontal placement	Apple–plate root distance in the horizontal plane ≤ 0.08932
Relative height	$\lvert z_a - z_p \rvert \leq 0.1028$ and $z_a - z_p > -0.01$
Table support	$z_a, z_p > z_s - 0.05$ and $z_a < z_s + 0.10$
Apple–plate contact	Contact force > 0.44 N
Settling	Apple linear speed < 0.1 m/s
Hand release	Each thumb task link is more than 0.10 m from the apple root

Depth-to-control mapping. To turn the simulator's depth images into renderer inputs, we apply a fixed mapping from depth to grayscale intensity. For depth d in meters, the control value is

$$I(d) = \mathrm{round}\left[255\left(1 - \mathrm{clip}\left(\frac{d - 0.15}{0.75}, 0, 1\right)\right)\right].$$

This mapping gives nearby surfaces brighter intensities and distant surfaces darker intensities, using the same limits for every frame. Nonpositive and nonfinite depths map to black, and we repeat the grayscale values across 3 channels. We resize the depth controls to 736×552 with bilinear antialiasing, then center-crop to 736×544. We resize the generated RGB to 640×480 with Pillow bilinear filtering for the policy.

Task completion. Tab. C.8 gives the exact simulator-state thresholds used to determine task completion. All conditions must hold at the same time, using the current apple and plate poses. In the table, z_a, z_p, and z_s denote the apple-root, plate-root, and tabletop heights, respectively. Hand-release distances run from the apple root to the left and right thumb task links, `left_hand_thumb_2_link` and `right_hand_thumb_2_link`. The episode ends as soon as the complete predicate is satisfied, with no additional interval during which success must be maintained.

Policy fine-tuning. The policy-training record lists 402 real demonstrations for fine-tuning GR00T N1.7 (NVIDIA, 2026d), comprising 90,710 frames at 15 Hz and 84,680 eligible action-chunk start timesteps. During a single 40,000-step run, we train the projector and action model while keeping the vision and language backbones frozen. All six evaluated checkpoints come from this run. Policy fine-tuning uses only the real demonstrations.

Policy fine-tuning runs on 1 GB300 in BF16, using AdamW with a global batch size of 96, weight decay 10^{-5}, and gradient clipping at 1. The learning rate reaches 10^{-4} after a 2,000-step warmup and then follows a cosine schedule to zero at step 40,000. Training applies state dropout of 0.2 and color jitter with brightness, contrast,

Table C.9: **Paired apple-task outcomes across policy checkpoints.** Each policy is evaluated on 32 matched calibration starts under the baseline appearance, and the four outcome counts sum to 32.

Policy steps	Both succeed	RTX only	Transfer only	Both fail
5,000	13	4	7	8
10,000	10	6	2	14
15,000	20	2	7	3
20,000	17	7	4	4
30,000	13	5	5	9
40,000	13	4	6	9

saturation, and hue settings of 0.3, 0.4, 0.5, and 0.08, respectively.

Appearance calibration. We encode the appearance prompts as compact structured JSON captions that also instruct the renderer to preserve the geometry and motion supplied by depth. The calibrated baseline requests a yellow/orange/red apple, a shallow white plate, a charcoal fabric-covered table and background, silver robot wrists, black fingers, and diffuse neutral lighting. Starting from this baseline, the three appearance variants request a green apple, a red plate, or a green apple together with a red plate and white tablecloth. The combined variant keeps the charcoal background. We refine the baseline prompt against real demonstration images using the appearance evaluation's calibration starts and 15,000-step policy.

Paired outcomes across policy checkpoints. Tab. C.9 supplements the aggregate completion rates with the joint outcome of each matched start. It separates starts on which both renderers lead to success, only Isaac RTX does, only Cosmos3-Nano-Sim-Transfer does, or both fail. All six policy checkpoints use the same baseline appearance and calibration starts used during prompt refinement.

C.5. Stage Comparisons and Failure Cases

Matched training-stage comparisons. The aggregate scores summarize benchmark quality and control fidelity, while matched visual examples show how each training stage renders the same controlled scene. Figs. C.1 to C.4 use example 2 of the model's sample data to compare the bidirectional teacher, TF, TF-dCM, and SF-DMD under depth, segmentation, edge, and blur controls. For each modality, we prepare the control from the same source video and pair it with the same structured caption. All four stages use seed 2025 with no RGB conditioning frames, so each model generates its appearance from the supplied language and controls.

We use the selected checkpoints and stage-specific samplers described in Sec. 6.5, with control guidance fixed at 1 for all four stages. All causal stages use full history without attention sinks. We directly resize the controls and generate 121 frames at 832×480 and 30 FPS for every output. Each grid places the stages at common timestamps—frames 0, 60, and 120—so readers can compare their RGB renderings against the same supplied structure as the scene develops.



Figure C.1: **Matched training-stage comparison with depth control.** Rows show the source RGB, supplied depth, and outputs from the bidirectional teacher, TF, TF-dCM, and SF-DMD. All stages share the same example 2 video, prompt, control, and seed. Columns show frames 0, 60, and 120 (0, 2, and 4 seconds at 30 FPS).

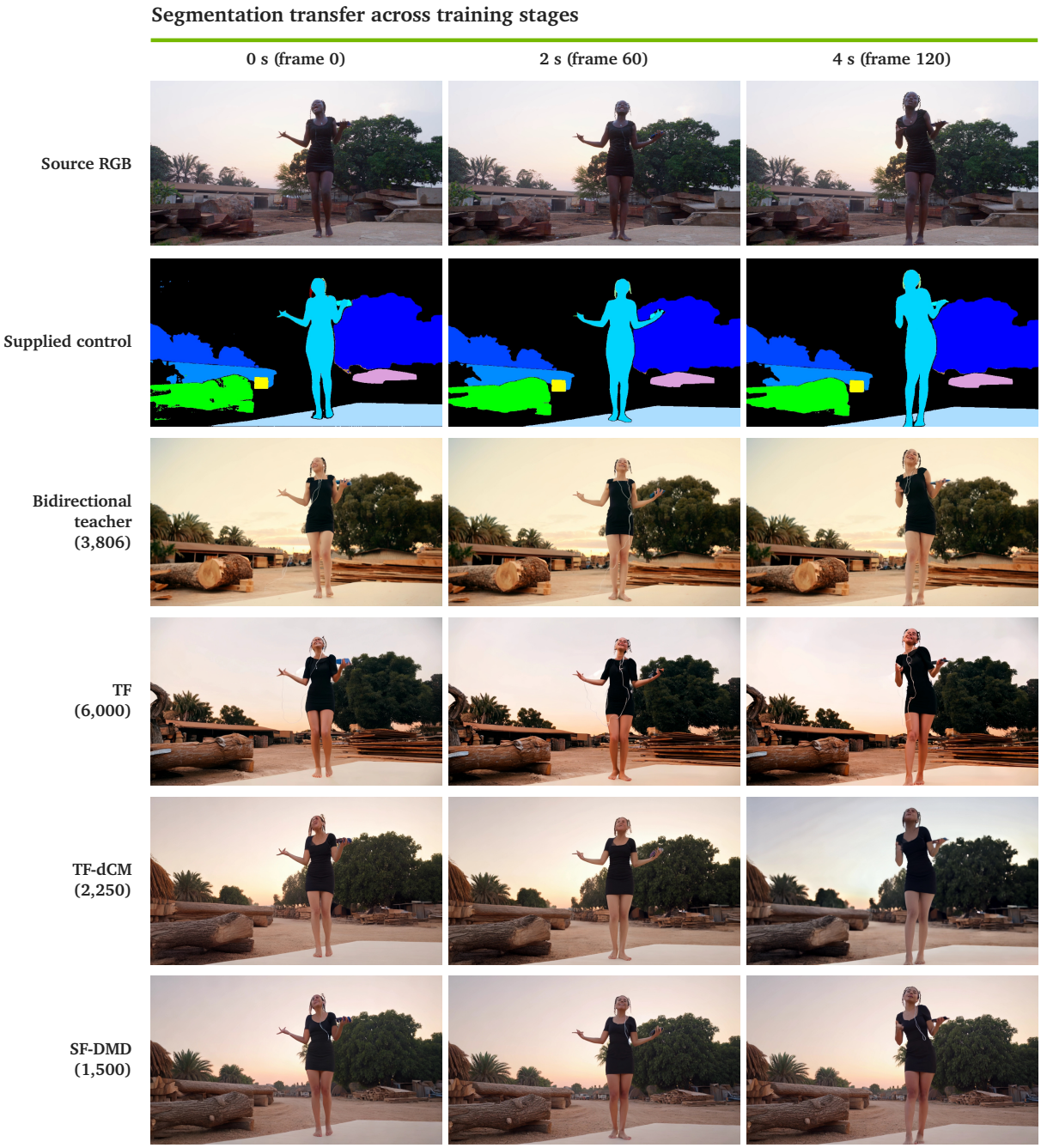


Figure C.2: **Matched training-stage comparison with segmentation control.** Rows show the source RGB, supplied segmentation, and outputs from the bidirectional teacher, TF, TF-dCM, and SF-DMD. All stages share the same example 2 video, prompt, control, and seed. Columns show frames 0, 60, and 120 (0, 2, and 4 seconds at 30 FPS).

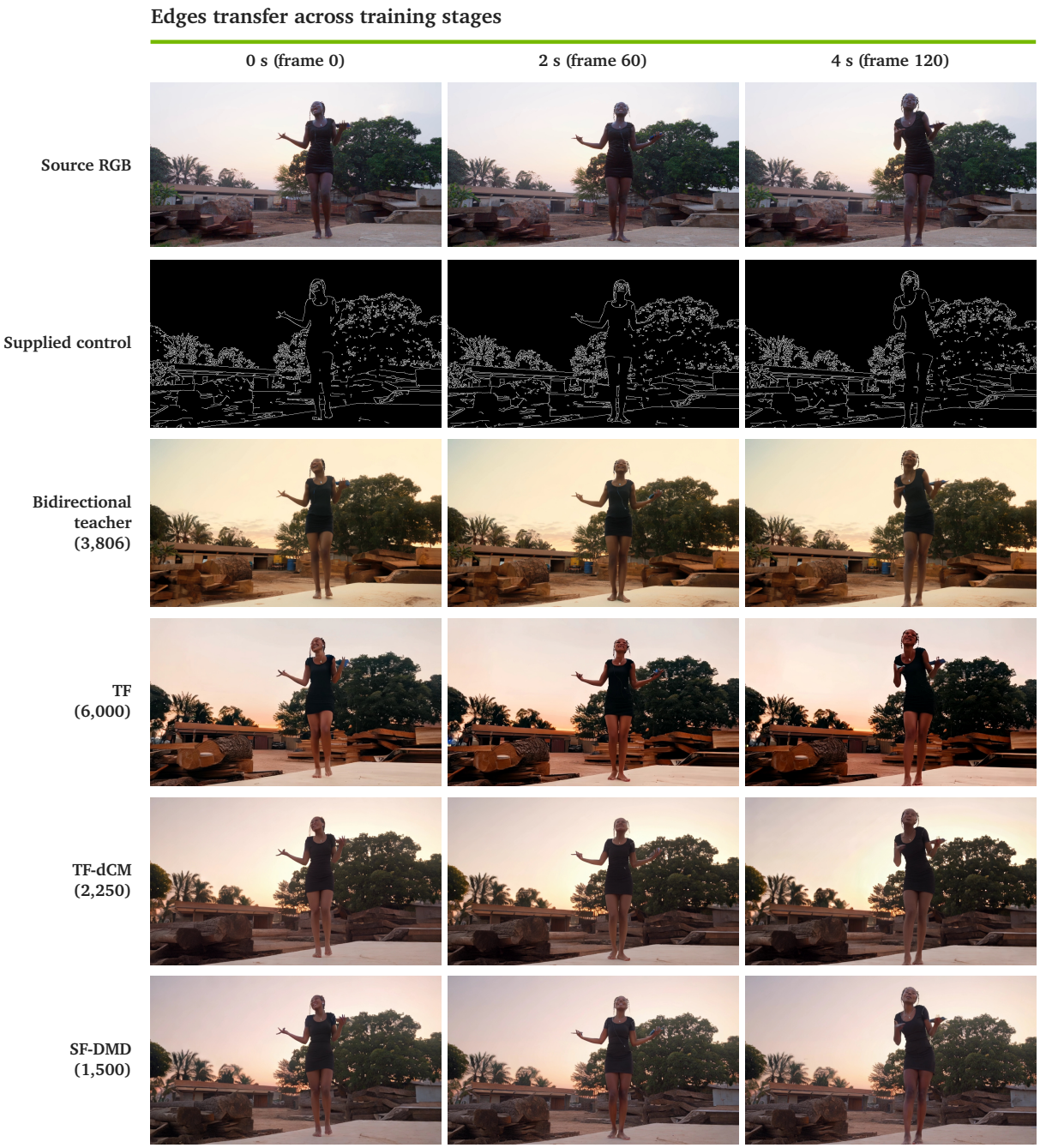


Figure C.3: **Matched training-stage comparison with edge control.** Rows show the source RGB, supplied edges, and outputs from the bidirectional teacher, TF, TF-dCM, and SF-DMD. Edges are computed from the resized source RGB with the same extraction preset for all stages. Columns show frames 0, 60, and 120 (0, 2, and 4 seconds at 30 FPS).



Figure C.4: **Matched training-stage comparison with blur control.** Rows show the source RGB, supplied blur, and outputs from the bidirectional teacher, TF, TF-dCM, and SF-DMD. Blur controls are computed from the resized source RGB with the same blur preset for all stages. Columns show frames 0, 60, and 120 (0, 2, and 4 seconds at 30 FPS).

Failure patterns. The QA recordings and saved VisDex rollouts illustrate five failure patterns in Cosmos3-Nano-Sim-Transfer: temporal degradation, object hallucination, appearance drift, geometric deformation, and errors in following the supplied controls. The examples cover all four control modalities, with source RGB, supplied control, and generated observations shown side by side. All runs use 4 denoising steps, language and control guidance 1, and flow shift 5. The edge and blur examples generate 832×480 video with seed 42, while the depth and segmentation examples generate 640×640 video with seed 2025. Each caption specifies the recording length and history settings.

Temporal degradation and visual artifacts. The recorded long rollouts exhibit colored spots, ghosting, and loss of structural detail. Fig. C.5 compares an earlier and a later moment in the same robot sequence. Near the end of the rollout, the generated arms and table appear smeared even though their outlines remain distinct in the source RGB and edge controls. The QA review reports partial mitigation after changes to the history settings, with artifacts remaining in the tested cases.

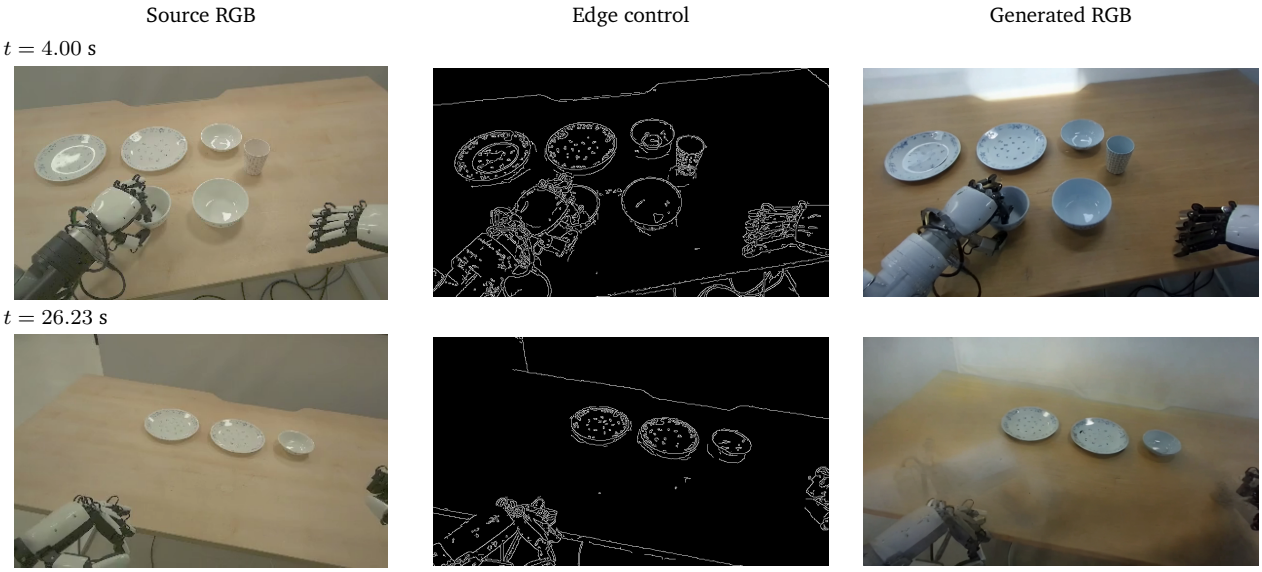


Figure C.5: **Temporal degradation in Cosmos3-Nano-Sim-Transfer.** Matched source, edge-control, and generated frames at 4.00 and 26.23 s show late-rollout ghosting around the robot arms and across the table. The recorded generation contains 788 frames at 30 FPS and uses a 198-latent-frame K/V window with 3 sink frames. These settings differ from the fixed-window VisDex runtime evaluation.

Object hallucination and duplication. Another failure changes which objects appear in the scene. In the warehouse example in Fig. C.6, the source RGB and edge control show open floor below a pallet. The generated observation fills this region with additional boxes and a supporting structure. The QA review also reports pedestrian duplication in generated videos.

Identity and appearance drift. In Fig. C.7, the generated red vehicle has inconsistent body proportions and front–rear appearance across the three displayed frames of a passing maneuver. The vehicle is visible in all three frames. The QA review also reports offscreen appearance drift that persisted with an explicit appearance-preservation caption.

Deformation and unrealistic structure. The blur-conditioned example in Fig. C.8 preserves the bowl's location while changing its internal geometry. The generated inner bowl is smaller relative to the outer bowl, and the spacing between their rims differs from the source. This changes the proportions of the nested bowls even though the surrounding tabletop remains recognizable.

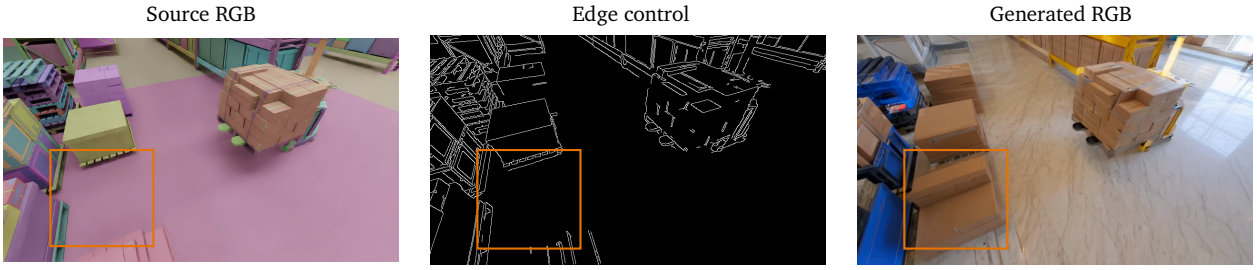


Figure C.6: **Object hallucination in Cosmos3-Nano-Sim-Transfer.** At 4.17 s, the highlighted floor region below the pallet contains additional boxes and shelf-like structure in the generated observation. The source and edge control do not contain these objects. This 122-frame, 29-FPS example uses a 32-latent-frame K/V window and 3 sink frames.

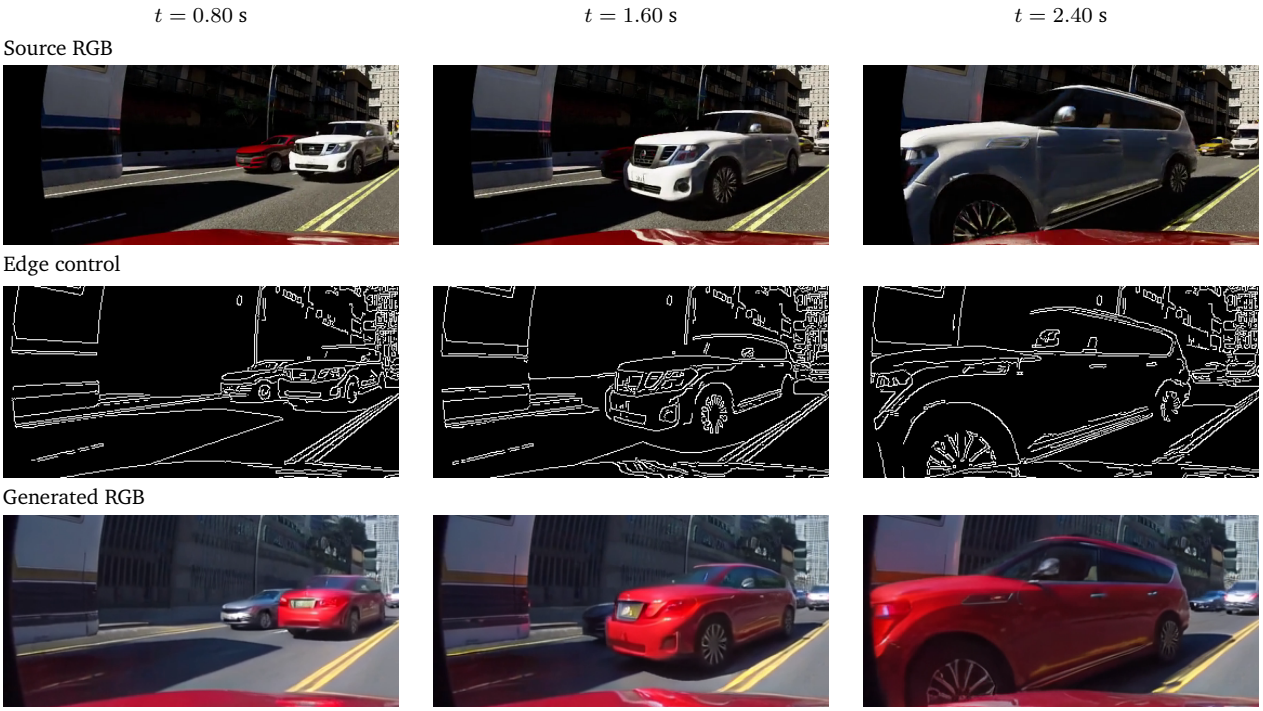


Figure C.7: **Appearance inconsistency in Cosmos3-Nano-Sim-Transfer.** Matched crops show inconsistent body proportions and front–rear interpretation of the generated red vehicle during a passing maneuver. The vehicle is visible in all three illustrated frames. The 121-frame, 30-FPS generation uses a 31-latent-frame K/V window and 3 sink frames.

Object motion and control following. The depth and segmentation examples show disagreement between generated object positions and the supplied controls. The prompts describe a shoe and a blue turtle on the counter. In both rows of Fig. C.9, the source RGB shows the target in the robot's raised hand and no target on the table. The generated observation instead places the prompted object on the table.



Figure C.8: **Bowl deformation under blur conditioning.** Matched crops at 10.00 s show a smaller inner bowl and altered rim spacing in the generated observation compared with the source. The supplied control uses the medium blur preset. This 800-frame, 30-FPS recording uses a 201-latent-frame K/V window and 3 sink frames.

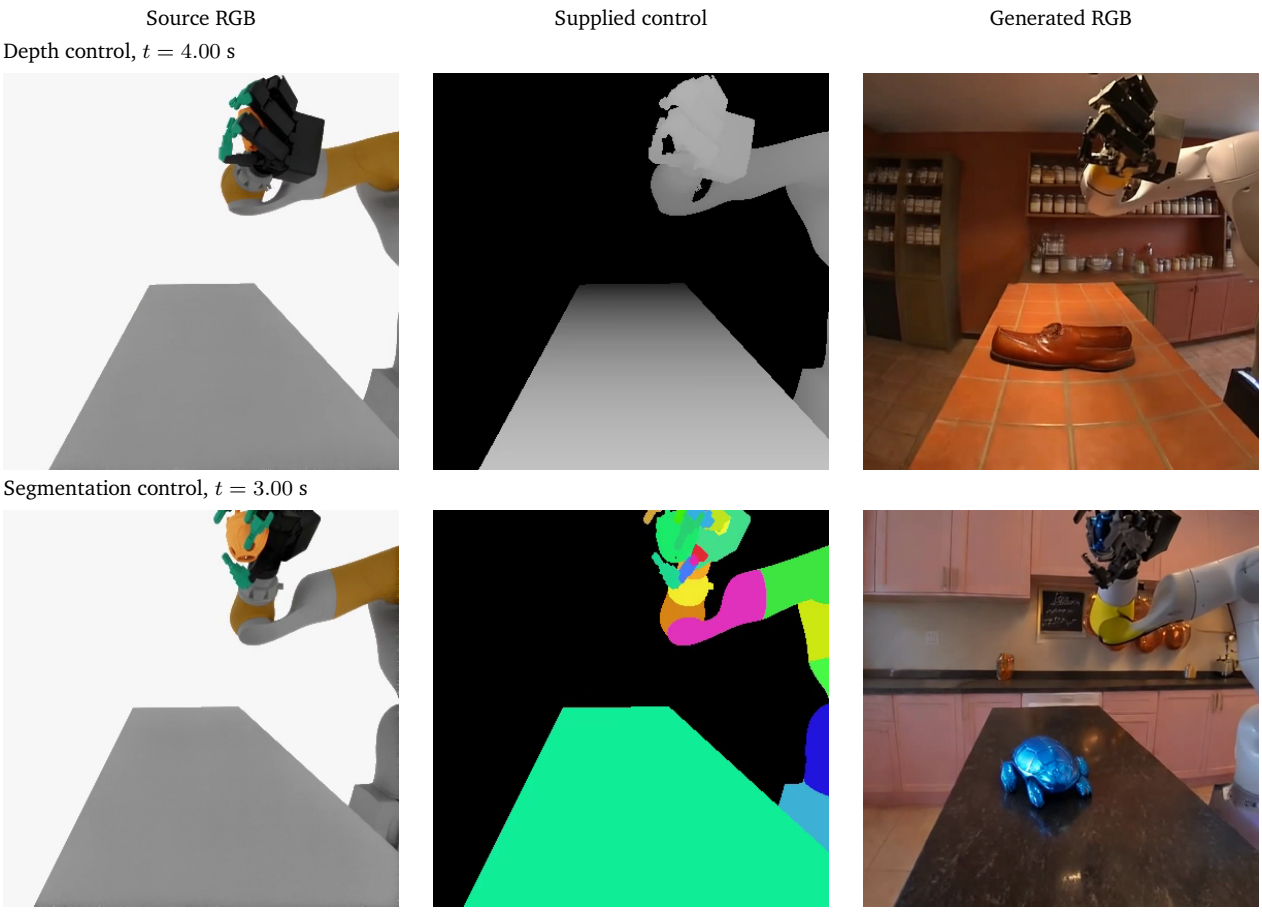


Figure C.9: **Object-motion errors under depth and segmentation conditioning.** The depth example at 4.00 s and the segmentation example at 3.00 s show the simulator holding the target above an empty table. The prompts describe the shoe and turtle on the counter, and the generated observations place them there despite the lifted target in the source. Both examples come from 600-frame, 30-FPS VisDex rollouts using the iteration-1,500 SF-DMD checkpoint, a 30-latent-frame K/V window, and 3 sink frames.

D. Cosmos3-Nano-Transfer/Sim-Auto: Additional Details

D.1. Architecture Details

D.1.1. LiDAR Tokenizer

The joint RGB–LiDAR model uses our LiDAR tokenizer, a continuous variational autoencoder with a hierarchical attention encoder and decoder. Its architecture is inspired by the LiDAR-native tokenizer in LidarVerse (Che et al., 2026), particularly its spatial neighborhood attention, causal temporal attention, and preservation of 1 latent time step per sweep. It maps each sweep to a spatial latent grid while preserving the sequence's temporal resolution. The tokenizer is trained for reconstruction and then held fixed during joint generative training. Its reconstruction objective is separate from the RGB and LiDAR diffusion losses used to train the joint model.

Range representation and normalization. Following the calibrated range-view representation used for driving LiDAR generation (Ren et al., 2025a), each Pandar128 sweep is rasterized onto a 128×3600 grid. Rows retain the calibrated beam elevations, and columns index azimuth. We reduce the width to 1800 by selecting the nearest return within each pair of adjacent columns and retaining that return's intensity. Four columns from the opposite azimuth boundary are appended at each side, yielding a 128×1808 network canvas. This circular padding makes the width divisible by 16 without rescaling the 1800 measured angular bins. The padded columns are cropped before reconstruction losses and point-cloud evaluation.

Each input has three channels: metric range r, unit-scaled intensity i, and binary validity m. The valid range interval is $[5, 100]$ m. Within the network, range and intensity are transformed as

$$r_{\mathrm{n}} = 2\frac{r-5}{100-5} - 1, \qquad i_{\mathrm{n}} = 2i - 1. \tag{D.1}$$

Missing returns and ranges outside the interval receive $m = 0$ and a value of -1 in both data channels. The explicit validity channel distinguishes a missing return from a valid measurement at a normalization boundary.

Encoder and decoder. The encoder applies a 2×2 patch embedding independently to each sweep, followed by three 2×2 spatial merges. The four stages have channel widths $(128, 256, 512, 1024)$, depths $(3, 3, 3, 3)$, and attention-head counts $(4, 4, 8, 8)$. Tab. D.1 lists the corresponding feature grids. At the first three stages, spatial mixing uses neighborhood attention (Hassani et al., 2023) with a 5×45 window, unit dilation, and circular padding along azimuth. The first stage has no temporal attention. At the next two stages, each spatial block is followed by causal temporal attention at each spatial position. The final stage uses three joint spatiotemporal attention blocks over the $T \times 8 \times 113$ bottleneck grid, with three-axis rotary positional encoding and a causal mask along time. The feed-forward expansion ratio is 3. Learned spatial embeddings are added at the encoder input and decoder bottleneck, and local spatial attention also uses axial rotary coordinates.

The posterior has 128 latent channels, with separate predicted means and log variances. The decoder projects a latent slice to 1024 channels and mirrors the hierarchy through joint bottleneck attention, three spatial expansions, and a final 2×2 unpatchification. At intermediate decoder resolutions, temporal attention precedes spatial attention. The output contains normalized range, normalized intensity, and a validity logit. After inverse normalization, the default decoding rule retains rays with sigmoid validity probability at least 0.5 and sets invalid range and intensity to zero.

Spatial stride and temporal causality. For a batch of B clips containing T sweeps, the encoder maps

$$\mathbb{R}^{B\times 3\times T\times 128\times 1808} \longrightarrow \mathbb{R}^{B\times 128\times T\times 8\times 113}. \tag{D.2}$$

The effective stride in time, elevation, and azimuth is $(1, 16, 16)$ relative to the network input. Including the initial column pooling, the sampling stride is 16 rows and 32 columns relative to the native 128×3600 raster. Attention mixes information beyond these sampling intervals, so they do not bound a latent token's spatial

Table D.1: **Our LiDAR tokenizer encoder.** The grid is shown per sweep after the input patch embedding or the preceding merge. Every stage preserves the number of sweeps. The decoder reverses the spatial hierarchy, with temporal attention before spatial attention at the intermediate stages.

Stage	Grid	Channels	Heads	Attention blocks
1	64×904	128	4	3 spatial neighborhood blocks
2	32×452	256	4	3 spatial–temporal pairs
3	16×226	512	8	3 spatial–temporal pairs
4	8×113	1024	8	3 joint spatiotemporal blocks
Posterior	8×113	128	—	Mean and log variance per latent coordinate

receptive field. There is no temporal downsampling or upsampling, so each 10-Hz input sweep retains 1 latent time step at a 0.1-second interval. The joint diffusion model uses 1×1 patches on this latent grid, giving $8 \cdot 113 = 904$ LiDAR tokens per sweep. This diffusion patch size is distinct from the tokenizer's 2×2 input patches and its accumulated 16×16 spatial stride.

Both encoder and decoder restrict temporal attention to keys at times $t' \leq t$ for a query at time t. At the joint bottleneck, all spatial positions at an allowed time are visible, including other positions in the current sweep. Consequently, encoding sweep t does not require future input sweeps, and decoding its latent does not require future latent slices. Spatial attention, spatial resampling, and the channel projections operate within individual sweeps and preserve this temporal dependency.

Long clips are processed with temporal key/value caches. The joint long-clip recipe uses blocks of 20 new sweeps and a context cap of 21 sweeps, leaving one preceding sweep's cached keys and values at a full block boundary. The cap bounds retained positions, whose features can already contain information from earlier sweeps. Within each block, the causal mask still prevents an early sweep from reading later sweeps in that block. These are tokenizer execution settings, separate from the simulation model's 4-sweep generation chunks. The inference interface uses the posterior mean and applies channelwise standardization with latent means and standard deviations stored in the tokenizer checkpoint. Decoding reverses this standardization before reconstructing the range view.

Reconstruction objective. Let j index all $N = BTHW$ rays in a batch after removing the circular padding, with $H = 128$ and $W = 1800$. Let $\widehat{r}_{\mathrm{n},j}$ and $\widehat{i}_{\mathrm{n},j}$ be the normalized reconstructions and ℓ_j the predicted validity logit. Range and intensity use masked absolute errors averaged over all rays:

$$\mathcal{L}_{\text{range}} = \frac{1}{N} \sum_j m_j \left| \widehat{r}_{\mathrm{n},j} - r_{\mathrm{n},j} \right|, \tag{D.3}$$

$$\mathcal{L}_{\text{intensity}} = \frac{1}{N} \sum_j m_j \left| \widehat{i}_{\mathrm{n},j} - i_{\mathrm{n},j} \right|. \tag{D.4}$$

The denominator includes missing rays, so these terms scale with the valid-return fraction. Validity is supervised over all rays using binary cross-entropy with logits, $\mathcal{L}_{\text{mask}} = N^{-1} \sum_j \mathrm{BCEWithLogits}(\ell_j, m_j)$. For calibrated unit ray directions $\mathbf{d}_j$, the point term is

$$\mathcal{L}_{\text{point}} = \frac{1}{3N} \sum_j m_j \left\| \widehat{\mathbf{p}}_j - \mathbf{p}_j \right\|_1, \qquad \mathbf{p}_j = r_j \mathbf{d}_j. \tag{D.5}$$

Point coordinates are in meters and use corresponding rays rather than nearest-neighbor point matching. In the implemented unprojection, decoded ranges at or outside the open interval $(5, 100)$ m are mapped to the origin before this term is evaluated. The same boundary rule is applied to the targets.

The gradient term penalizes spatial variation in the masked normalized range residual. At each sampling stride $s \in \{1, 2, 4, 8\}$, let $e_j^{(s)} = m_j^{(s)}(\widehat{r}_{\mathrm{n},j}^{(s)} - r_{\mathrm{n},j}^{(s)})$ on the strided grid. Let $\mathcal{E}_s$ contain horizontal and vertical adjacent ray pairs on that grid. Then

$$\mathcal{L}_{\text{grad}} = \sum_{s\in\{1,2,4,8\}} \frac{\sum_{(j,k)\in\mathcal{E}_s} m_j^{(s)} m_k^{(s)} \rho\left(e_j^{(s)} - e_k^{(s)}\right)}{\max\left(1, \sum_j m_j^{(s)}\right)}, \tag{D.6}$$

where $\rho(u) = u^2/2$ for $|u| < 1$ and $\rho(u) = |u| - 1/2$ otherwise. Only pairs with two valid returns contribute, and finite differences do not wrap across the cropped azimuth boundary. The point and gradient terms supervise range geometry, while intensity has its own reconstruction term.

The complete objective uses the effective weights

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{range}} + \mathcal{L}_{\text{intensity}} + \mathcal{L}_{\text{point}} + \mathcal{L}_{\text{mask}} + 2\mathcal{L}_{\text{grad}} + 10^{-6}\mathcal{L}_{\text{KL}}. \tag{D.7}$$

For K latent coordinates with posterior means μ_k and log variances a_k, we use

$$\mathcal{L}_{\text{KL}} = \frac{1}{2K}\sum_k \left(\mu_k^2 + \exp(a_k) - 1 - a_k\right). \tag{D.8}$$

Training samples the Gaussian posterior through reparameterization. The normalized reconstruction errors, metric point errors, and validity and KL terms retain their respective units and reductions.

Tokenizer training recipe. Our tokenizer is trained on approximately 1M LiDAR sweeps. The reference recipe draws contiguous windows of 8 LiDAR sweeps from MADS (NVIDIA, 2026a). The encoder and decoder are optimized together. The documented refinement of an existing tokenizer checkpoint runs for 30K updates in float32, with a per-GPU batch size of 2 and gradient-norm clipping at 1. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and zero weight decay. The learning rate warms up linearly for 1K updates to 10^{-5} and then decays linearly to 10^{-6} over the remaining updates. This refinement follows earlier tokenizer training. The tokenizer checkpoint stores the latent normalization statistics and metric range bounds alongside the weights.

Tab. D.2 compares our tokenizer with the public NVIDIA PAI tokenizer distributed with LidarVerse (Che et al., 2026; NVIDIA, 2026b). The released checkpoint reconstructs range and validity, while our tokenizer also reconstructs intensity.

Reconstruction evaluation. Tab. D.3 reports reconstruction on 846 sweeps from 94 LiDARBench MADS clips, with both inputs and targets restricted to $[5, 100]$ m. Range and mask metrics use a shared 128×3600 sensor raster. Our cropped 128×1800 output and LidarVerse's 128×1024 output are mapped to this raster by nearest calibrated azimuth, including circular wraparound and preceding-column tie breaking. The scores therefore include angular sampling loss. Range errors use all ground-truth-valid rays, including predicted-invalid rays, with absolute relative error defined as $|\widehat{r} - r|/r$. Mask IoU measures the valid-return class, using a validity threshold of 0.5 and predicted ranges in $[5, 100]$ m.

Chamfer distance compares calibrated Pandar128 point clouds in $[5, 100]$ m against the same native reference cloud, without upsampling decoded outputs. For reference and reconstructed point sets P and Q, the unsquared symmetric distance in meters is

$$\mathrm{CD}(P, Q) = \frac{1}{2}\left(\frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}\|p - q\|_2 + \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P}\|q - p\|_2\right). \tag{D.9}$$

Table D.2: **LiDAR tokenizer configuration comparison.** Settings describe our joint RGB–LiDAR tokenizer and the public NVIDIA PAI release of LidarVerse (NVIDIA, 2026b). Spatial grids are per sweep, with strides relative to the corresponding network input.

Property	Ours	LidarVerse
Input canvas	128×1808	128×1024
Reconstructed quantities	Range, intensity, validity	Range, validity
Metric range	5–100 m	2.5–100 m
Input patch	2×2	2×2
Stride (t, h, w)	$(1, 16, 16)$	$(1, 8, 8)$
Latent grid	8×113	16×128
Latent channels	128	64
Parameters	148.5M	64.5M
Stage widths	$(128, 256, 512, 1024)$	$(128, 256, 512)$
Stage depths	$(3, 3, 3, 3)$	$(4, 2, 4)$
Attention heads	$(4, 4, 8, 8)$	$(4, 16, 16)$
Spatial neighborhood	5×45, circular azimuth	5×35, circular azimuth
Temporal attention	Causal, factorized and joint	Causal, factorized
Bottleneck	Joint spatiotemporal attention	Spatial and temporal attention

Table D.3: **LiDAR tokenizer reconstruction on matched sensor targets.** Both inputs and evaluated outputs use $[5, 100]$ m returns on the same 846 sweeps. Range and mask metrics share the native 128×3600 target raster, and Chamfer uses the same native reference point cloud with a 50K-point cap. Lower is better for errors and higher is better for mean mask IoU. Scale factors multiply the displayed values. The green row identifies our tokenizer, with best and second-best point estimates bold and underlined in each column.

Tokenizer	Range MAE (m) ↓	Median abs. error (10^{-2} m) ↓	Abs. relative error ($\times 10^{-2}$) ↓	Chamfer distance (m) ↓	Mean mask IoU ↑
Ours	**0.4163**	**3.597**	**1.262**	**0.1674**	**0.7554**
LidarVerse	<u>0.9742</u>	<u>9.667</u>	<u>3.183</u>	<u>0.1731</u>	<u>0.7414</u>

Each cloud is independently sampled without replacement to at most 50K points, using seeds $2t$ and $2t + 1$ for reference and reconstruction at sweep ordinal $t \in \{0, \ldots, 8\}$. Both models share the reference sample, and all 846 sweeps are scoreable. Metrics average 9 sweeps per clip and then 94 clips equally, including the per-sweep median errors and foreground mask IoUs. Paired 95% percentile bootstrap intervals use 10K clip resamples.

Tokenizer reconstruction. How accurately does the LiDAR tokenizer preserve range geometry and valid returns? We compare our tokenizer with the public NVIDIA PAI tokenizer distributed with LidarVerse on the same sensor targets. With matched $[5, 100]$ m inputs, ours reduces MAE by 57.3% and Chamfer distance by 3.3%, while increasing mean mask IoU by 1.398 pp (Tab. D.3). Our tokenizer thus provides higher-fidelity reconstructions and, together with the joint transformer, supports the improved LiDAR generation results in Tab. 16.

D.1.2. Attention Visibility and Execution

The main text introduces our decomposed multi-view and multimodal attention and its causal extension (Sec. 7.1.3 and 7.1.4). Here we explain how maskless token grouping avoids FlexAttention's partial-block masking overhead and how deduplicating overlapping key groups prevents excess same-view attention.

Sliding-window attention. A same-view sliding window restricts the temporal context available within each camera stream. Causal windows end at the query time and clip at the beginning of the sequence, while bidirectional neighborhoods can include both earlier and later frames. The cross-view window is configured independently, allowing temporal context within a view to differ from the interval used to exchange information

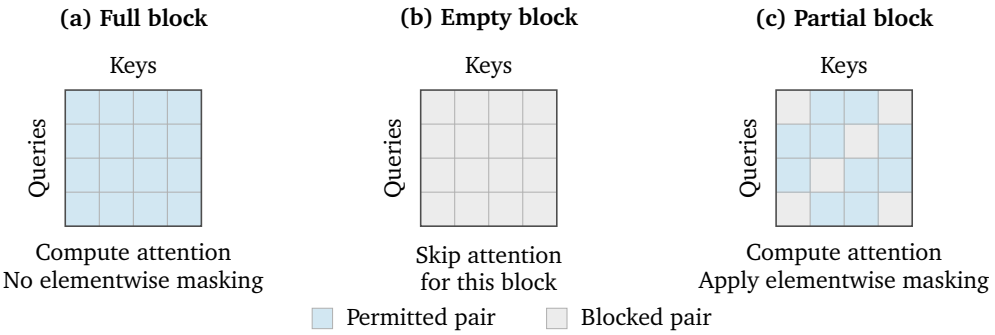


Figure D.1: **FlexAttention block types.** Rows are queries and columns are keys, with blue cells denoting permitted pairs and gray cells denoting blocked pairs. (a) Full blocks execute attention without elementwise masking, (b) empty blocks skip attention, and (c) partial blocks execute attention with elementwise masking. The cell counts and visibility patterns are schematic.

across views. The current configuration specifies both windows in seconds and maps same-view bounds to latent frame offsets using each stream's latent frame period. These windows constrain which keys a query can read, while the cache retention policy in Sec. 4 determines which history keys remain stored.

The sensor-to-sensor attention pattern in Fig. 20 covers supplied observations and prediction targets within the sensor streams, subject to their stage-specific visibility restrictions.

Supplied control streams retain separate directed connections. Sensor queries read supplied control keys/values associated with the same camera or LiDAR stream, with temporal access determined by the conditioning configuration and input availability. Control queries read control keys/values within the same view. They can additionally read sensor keys/values in that view when control-to-sensor attention is enabled, making control features depend on the sensor targets. Cross-view sensor attention does not automatically grant access to controls from other views.

Attention implementation details. We initially implemented decomposed multi-view attention with PyTorch FlexAttention and a FlashAttention-4 backend (Zadouri et al., 2026). FlexAttention expresses permitted query–key connections through a Boolean `mask_mod` function, while `score_mod` can modify attention scores before softmax. We primarily used `mask_mod` to encode sample boundaries, view restrictions, and temporal windows, constructing a `BlockMask` once per batch. This interface supports packed samples with different numbers of views, resolutions, and frame counts.

Partial-block overhead and alignment. FlexAttention skips empty blocks and processes fully visible blocks without elementwise masking. Partially visible blocks require mask evaluation and application inside the attention kernel, reducing efficiency for our decomposed pattern. The evaluated FlashAttention-4 backend typically used 256×128 query–key blocks on Blackwell and 128×128 blocks on Hopper. Our example 480p and 720p inputs contain 390 and 920 latent tokens per frame, respectively, so frame boundaries do not align with these block dimensions. Fig. D.1 illustrates the full, empty, and partial blocks and their masking requirements.

The measured attention workload reaches 45–55% model FLOP utilization (MFU) with unrestricted cross-view attention but only 10–20% with decomposed attention. To test whether block alignment contributes to this gap, we use a synthetic 1024×1024 input that produces 1024 tokens per frame with the Wan 2.2 tokenizer, aligning frame boundaries with the kernel blocks. Decomposed attention then reaches 45–55% MFU, supporting block alignment as a practical constraint. Padding frames to aligned lengths and using `score_mod` to exclude padding tokens works functionally but does not recover the desired performance. These observations motivate the

maskless implementation below, whose inference performance is evaluated in App. D.4.1.

Maskless token grouping. Rather than applying a complex visibility mask to the entire packed sequence, we gather query and key/value tokens into groups for same-stream, cross-sensor, control, and language attention. The groups follow the temporal and conditioning rules in Sec. 7.1.3 and 7.1.4. Each group is processed with efficient dense attention kernels, avoiding FlexAttention's partial-block masking overhead.

Deduplicating same-view keys. Efficient execution should preserve the intended balance between same-view and cross-view information. If the cross-sensor group also includes the query's own stream, keys in the overlapping temporal window appear in both the same-stream and cross-sensor groups. Combining these groups without deduplication counts those keys twice in the joint softmax, giving the duplicated same-view evidence excess weight relative to other views. This introduces an undesirable inductive bias toward within-view information rather than exchanging evidence across cameras. We therefore exclude the query's own stream from the cross-sensor group, retaining its keys only in the same-stream group. App. D.6.1 discusses its effect on cross-view consistency. With disjoint key groups, we can now combine their outputs to recover a single softmax over all permitted keys.

Joint attention normalization. For a fixed query and attention head, let $\mathbf{o}_b$ denote the locally normalized attention output for a nonempty key group b. Let Z_b be its softmax denominator: the sum of exponentiated attention scores over the keys in that group. We recover the joint output by weighting each group output by its share of the total denominator:

$$\mathbf{o} = \sum_b \frac{Z_b}{\sum_c Z_c} \mathbf{o}_b. \tag{D.10}$$

Here, c ranges over the same groups, and $Z_b / \sum_c Z_c$ is the total attention probability assigned to group b. Each attention call returns the log-sum-exp statistic $\ell_b = \log Z_b$. We use these statistics to compute the weights as $\exp(\ell_b - \mathrm{LSE}_c\, \ell_c)$. With disjoint key groups and the same attention scores as the unsplit computation, this weighted combination is mathematically equivalent to a single softmax over all allowed keys (Ye et al., 2025).

D.2. Data Preparation

D.2.1. Camera Rig

Fig. D.2 schematically illustrates the cameras' relative positions on the vehicle and their fields of view. Both Cosmos3-Nano-Transfer-Auto and Cosmos3-Nano-Sim-Auto sample camera combinations from this rig during training (Sec. 7.3), using the catalog in Tab. D.4.

D.2.2. WSM Rendering

WSM v4 uses semantic colors and symbols to distinguish map features and object states, as illustrated in Fig. D.3. Traffic signs use colored rectangles to distinguish categories such as stop, yield, and one-way, with dark-gray backs for recognized classes. Traffic lights encode signal state and direction, including directional triangles and pedestrian stop/go symbols. Road markings include seven directional-arrow shapes and filled regions for STOP and KEEP CLEAR. Road boundaries use six color groups for barriers or fences, curbs, road seams, guide posts, walls, and other boundaries. Poles and explicit wait lines use colored polylines, while crosswalks use filled polygons. Sign categories and STOP/KEEP CLEAR areas are represented without printed lettering. The Cosmos Framework WSM guide provides the full color palette and annotated examples.[7]

D.2.3. Captions

The examples in Boxes D.1–D.4 illustrate a timestamped per-camera caption and its prompt prefixes. We show the joint RGB–LiDAR task prefix and the camera prefix for an 11-camera selection, followed by the stored

[7] WSM rendering guide: github.com/NVIDIA/cosmos-framework/blob/main/.agents/skills/cookbooks/cosmos3/simulation/auto/wsm/docs/wsm4.md.

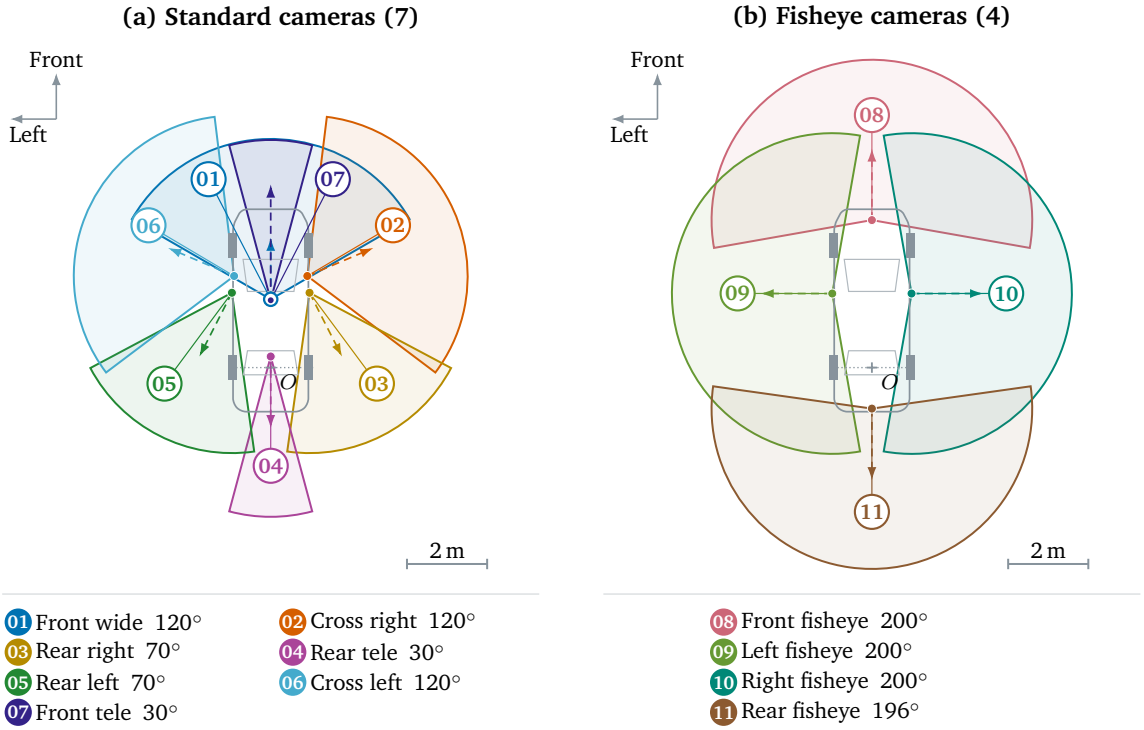


Figure D.2: **Symmetric schematic of the complete 11-camera rig.** (a) Seven standard cameras cover wide-angle and telephoto views. (b) Four fisheye cameras provide wider angular coverage. Colors and numbered labels identify individual cameras, and dashed arrows indicate viewing directions.

Table D.4: **Predefined camera combinations for training Cosmos3-Nano-Transfer-Auto and Cosmos3-Nano-Sim-Auto.** A tick marks an included camera; a blank cell marks an omitted camera. Headers give each camera's name and nominal field of view (FOV). The first 7 camera columns form the standard 7-camera rig. Tele denotes telephoto.

	Standard 7-camera rig							Additional fisheye cameras			
Count	Front wide 120°	Cross right 120°	Rear right 70°	Rear tele 30°	Rear left 70°	Cross left 120°	Front tele 30°	Front fisheye 200°	Left fisheye 200°	Right fisheye 200°	Rear fisheye 200°
1	✓										
2	✓						✓				
4								✓	✓	✓	✓
4	✓	✓		✓		✓					
5	✓	✓	✓		✓	✓					
5	✓							✓	✓	✓	✓
6	✓	✓	✓		✓	✓					✓
6	✓	✓				✓			✓	✓	✓
7	✓	✓	✓		✓	✓	✓				✓
7	✓	✓	✓	✓	✓	✓	✓				
8	✓	✓	✓		✓	✓	✓	✓			✓
8	✓	✓	✓	✓	✓	✓	✓	✓			
10	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

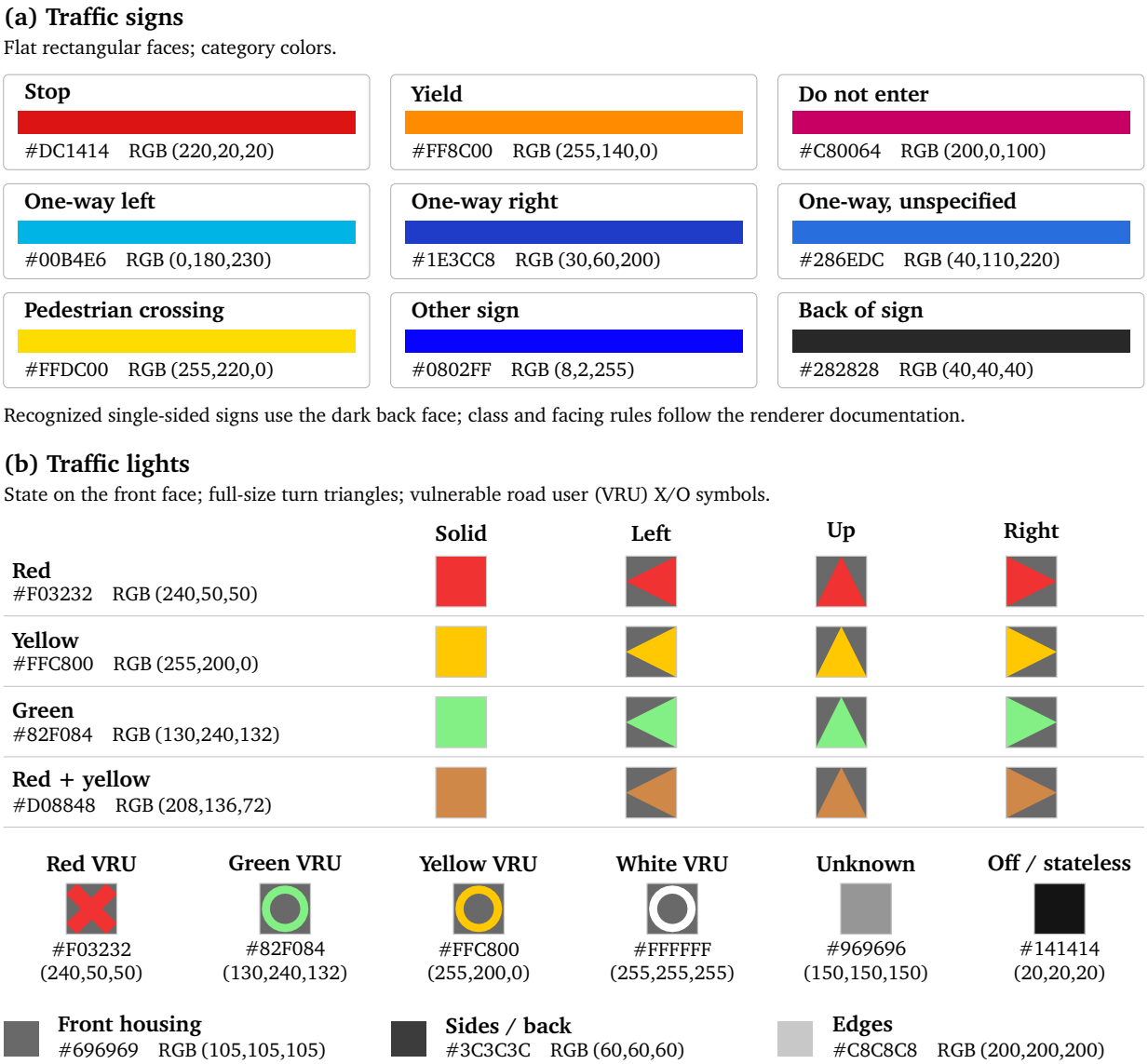


Figure D.3: **WSM v4 encodings, redrawn from the reference renderer documentation.** Traffic-sign category colors (top) and traffic-light states and directional symbols (bottom). Chart labels and backgrounds are explanatory and are not part of the rendered controls.

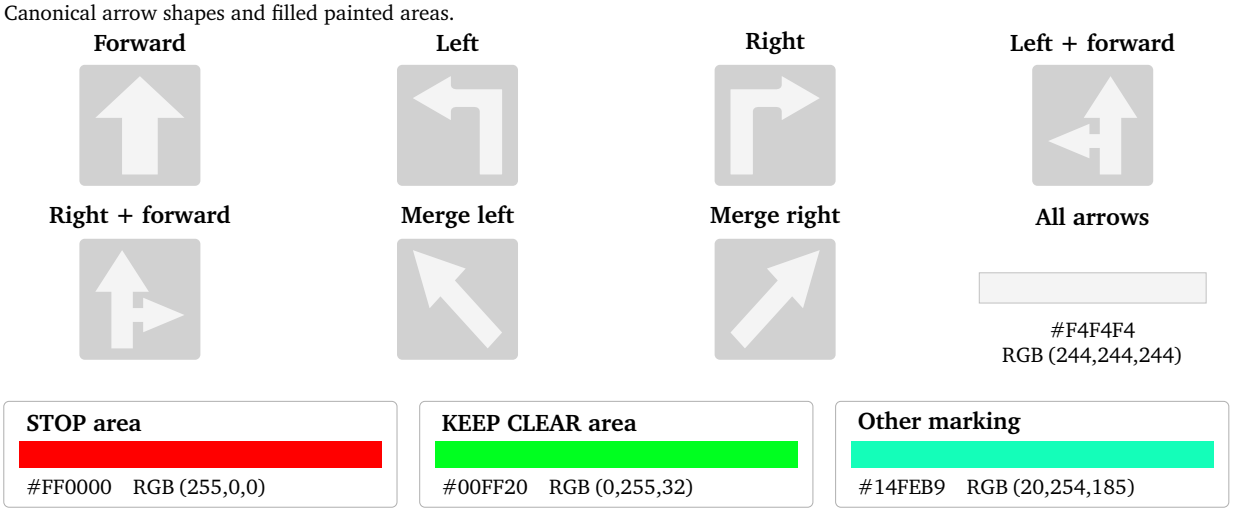
scene caption and, for comparison, the Cosmos-Transfer2.5 prompt for the same clip. Box titles identify the components and are excluded from the model input.

Prompt prefixes. Our prompt prefixes have two components, illustrated in Boxes D.1–D.2. The task prefix is a system prompt that specifies joint RGB–LiDAR generation from camera WSMs and LiDAR HD-map controls, with instructions for synchronized and consistent sensor outputs. The camera prefix describes the selected rig, listing each camera's viewing direction and field of view, before identifying the current camera. It is prepended to that camera's scene caption, with the rig description updated to match the sampled camera selection.

Prompt upsampling. Prompt upsampling turns concise scene specifications into dense, timestamped descriptions that match the style of our training captions. Synchronized WSM observations establish the scene geometry and motion, while the requested environmental attributes guide its appearance. We first construct

(c) Road markings

Canonical arrow shapes and filled painted areas.



Arrows follow the annotated footprint. STOP and KEEP CLEAR are filled areas without lettering.

(d) Road boundaries

Six color groups; all boundary subtypes use solid strokes.

Boundary group	Color	Hex / RGB
Barrier / fence		#FD01E8 RGB (253,1,232)
Curb / low curb / tall curb		#CFFFBF RGB (207,255,191)
Road boundary / seam		#009F60 RGB (0,159,96)
Obstacle bars / guide posts		#FFCF9F RGB (255,207,159)
Wall		#00809F RGB (0,128,159)
Other / unknown boundary		#EFBFFF RGB (239,191,255)

The renderer documentation lists the subtype-to-color mapping, including the fallback group.

Figure D.3: **WSM v4 encodings (continued).** Directional road markings and filled areas (top), and road-boundary colors (bottom), redrawn from the reference renderer documentation. The light-gray tiles behind near-white arrows provide contrast and are not part of the rendered controls.

a coherent scene description, assigning consistent appearance to objects shared across views and expanding environmental attributes into concrete visual cues, such as reduced contrast for distant objects in fog. We then express this shared scene from each camera's viewpoint, producing captions whose visible content and event timing follow the corresponding WSM sequence.

Prompt upsampling emphasizes concrete descriptions of object appearance rather than generic quality statements. Off-the-shelf, state-of-the-art VLMs can perform this workflow using our prompt upsampling guide in Cosmos Framework.

Box D.1: Task prefix: joint RGB–LiDAR generation

```
You are a helpful assistant that jointly generates temporally synchronized, geometrically consistent
autonomous-driving camera videos and LiDAR range-view sequences from per-camera scene descriptions and
provided control signals: per-camera World Scenario Map (WSM) control videos depicting the controlled
objects and road layout, and an HD-map control for LiDAR. Treat all camera views and LiDAR sweeps as
synchronized observations of the same driving scene, preserving each camera's viewpoint, shared ego
motion, road layout, object identity and motion, weather, lighting, cross-view consistency, and
camera-LiDAR alignment.

Follow WSM controls for vehicles (including trucks), cyclists, pedestrians, traffic lights, traffic signs,
road markings, lane boundaries, and road boundaries. Do not add objects or road features in these
categories that are absent from WSM. Use captions for appearance and unconstrained background details;
WSM takes precedence in any conflict.
```

Box D.2: Camera prefix: 11-camera rig and front-wide view

```
This multiview driving sequence contains time-aligned recordings from 11 vehicle-mounted cameras: front
wide-angle camera (forward-facing, 120° FOV); right-side wide-angle camera (right-facing, 120° FOV);
rear-right camera (rear-right-facing, 70° FOV); rear telephoto camera (backward-facing, 30° FOV);
rear-left camera (rear-left-facing, 70° FOV); left-side wide-angle camera (left-facing, 120° FOV); front
telephoto camera (forward-facing, 30° FOV); front fisheye camera (forward-facing, 200° FOV); left-side
fisheye camera (left-facing, 200° FOV); right-side fisheye camera (right-facing, 200° FOV); rear fisheye
camera (backward-facing, 200° FOV).

The description below is for the front wide-angle camera mounted on the vehicle. This camera is facing
forward and has a 120° field of view:
```

Box D.3: Scene caption excerpt: front-wide camera

```
[0.0s-1.0s]: The view is a first-person forward-facing dashcam perspective from inside a vehicle, with
the light-colored hood and a small hood ornament visible in the immediate foreground. The vehicle travels
on a paved asphalt road initially marked with solid yellow lines along both edges of the travel lane,
flanked by tall concrete retaining walls topped with metal safety railings on both sides; a lattice-style
electricity pylon rises above the top of the right retaining wall at the start of the segment. A black
passenger car drives directly ahead in the same lane, maintaining a consistent following distance. On the
left side, an elevated parallel roadway runs behind the concrete barrier, carrying white box trucks and
other vehicles; a blue square regulatory sign with a red diagonal stripe over a white straight-ahead
arrow is mounted on the left barrier, with a round silver-backed traffic sign positioned below it. On the
right retaining wall, a matching blue square regulatory sign with a red diagonal stripe over a white
right-curving arrow is mounted above a circular no-entry sign (red border, white field, black horizontal
bar). Tall curved streetlight poles line both sides of the road, and rolling green tree-covered hills are
visible in the far distance under a clear, cloudless bright blue sky.
[1.0s-4.5s]: As the vehicle moves steadily forward, the initial yellow edge lines transition to standard
white dashed lane dividers between travel lanes and solid white edge lines. The initial signs on both
barriers pass out of the frame as the vehicle progresses. Red-and-white striped construction delineator
posts appear spaced evenly along the top of the left concrete barrier, separating the main travel lane
from the adjacent left roadway. Multiple vehicles become visible traveling in the opposite direction on
the left parallel road: a dark gray hatchback, a small red compact car, and additional passenger cars. A
large metal gantry structure holding a digital display comes into view on the left before passing out of
frame. On the right side, a two-story building with white plaster walls, red brick trim, multiple
chimneys, and a sloped roof becomes visible behind the retaining wall, followed by a large roadside
billboard displaying printed imagery and unreadable text. The black car ahead remains centered in the
travel lane, continuing forward at a steady speed.
[4.5s-7.5s]: The vehicle continues forward past the end of the enclosed retaining wall section. The
right-side retaining wall gives way to a grassy roadside verge lined with metal guardrail, and a blue
circular mandatory straight-ahead sign (white upward arrow on blue background) appears on the right
shoulder. White painted chevron merge markings begin appearing on the asphalt of the right portion of the
road, indicating an upcoming lane merge or exit area. Additional opposite-direction traffic passes on the
left parallel road, including a black van, a white box truck, and more passenger cars. Small white
numbered markers, including one marked "13", are affixed to the face of the left concrete barrier at
regular intervals. The distant green hills remain visible, with several tall electricity pylons standing
on the hilltops under the clear sky.
```

Box D.4: Cosmos-Transfer2.5 prompt: front-wide camera

```
The video captures a daytime driving scene under clear, sunny weather conditions. The ego vehicle travels
along a multi-lane highway, maintaining a steady pace in the rightmost lane. Ahead, a black SUV drives in
the same lane, keeping a consistent distance from the ego vehicle. To the left, there is a concrete
barrier separating the opposing traffic lanes, marked with red-and-white striped poles for visibility. On
the right side of the road, a stone wall runs parallel to the highway, topped with a metal railing.

As the ego vehicle progresses, it passes several streetlights and utility poles lining the road. In the
distance, power lines stretch across the horizon, adding depth to the landscape. The surrounding
environment includes patches of greenery and some buildings visible beyond the walls. The road surface
appears well-maintained, with clear lane markings guiding the flow of traffic. No significant changes in
speed or direction are observed from either the ego vehicle or the lead vehicle throughout the sequence.
```

Table D.5: **Post-training configuration of the Cosmos3-Nano-Sim-Auto model.** Frame counts are per camera view after sensor alignment, and the upper bounds in Stages 1 and 2 describe the longest sampled clips. Clip lengths include the initial RGB frames and LiDAR sweeps used for conditioning. Stage 3 iterations count alternating student and fake-score updates. The long phase continues the short phase's counter from 1,500 to 1,800.

	Stage 1	**Stage 2**	**Stage 3: SF-DMD**	
Setting	**TF**	**TF-dCM**	**Short**	**Long**
RGB frames per view	≤ 397	≤ 397	93	397
Clip duration (seconds)	≤ 13.2	≤ 13.2	3.1	13.2
RGB latent frames per view	≤ 100	≤ 100	24	100
LiDAR sweeps	≤ 133	≤ 133	31	133
Causal chunks	≤ 34	≤ 34	9	34
GB200 GPUs	256	768	512	256
Training iterations	3,200	900	1,500	300
Student learning rate	10^{-5}	2×10^{-6}	10^{-6}	10^{-6}
Fake-score learning rate	—	—	2×10^{-7}	2×10^{-7}

D.3. Training Configuration

D.3.1. Cosmos3-Nano-Transfer-Auto Training

Modality-separate sampling for Stage 0. The modality-separate phase samples 480p RGB, 720p RGB, and LiDAR examples with probabilities 0.15, 0.15, and 0.70, respectively. RGB examples select one of the 11 cameras uniformly. LiDAR examples use the front-wide caption.

Optimization and initialization. We use global batches of 16 examples for modality-separate training, 32 for short-sequence joint training, and 16 for long-sequence joint training. In each phase, the learning rate increases linearly to the peak specified in Tab. 13 over the first 2,000 updates, then decreases linearly to 10^{-5}.

Training starts from Cosmos3-Nano with newly initialized LiDAR projections and sensor embeddings. The joint phases build on the preceding phase's exponential moving average (EMA) weights, carrying forward the learned modality projections and sensor embeddings. Each phase starts with fresh optimizer state and restarts the learning-rate schedule, update counter, and EMA averaging.

D.3.2. Cosmos3-Nano-Sim-Auto Training

The three causal stages use the data and sensor rates in Sec. 7.2, with resolution sampling and conditioning dropout as in Sec. 7.3. To fit the training memory budget, examples containing all 11 cameras simultaneously use 480p. At 720p, each training example uses fewer cameras, but the sampled combinations collectively cover

all 11 camera views. At inference, the trained model can generate all 11 views together at 720p. Each clip supplies 0, 1, or 2 observed RGB latent frames per view with probabilities 0.7, 0.2, and 0.1, paired with 0, 1, or 3 observed LiDAR sweeps, corresponding to T2V, I2V, and V2V. In the synchronized chunk layout of Sec. 7.1.4, the initial chunk contains 1 RGB latent frame per camera and 1 LiDAR sweep. When a training sequence ends with an incomplete causal chunk, we retain the remaining aligned RGB frames and LiDAR sweeps for training.

Training noise levels are shared across the views, sensors, and chunks of each clip. TF-dCM computes the loss in Sec. 3.3 for each camera or LiDAR latent sequence, then averages these losses within each modality. Clean control sequences contribute zero loss but are included in the sequence count used for averaging. For SF-DMD, the teacher-residual normalizer and losses are computed separately for RGB and LiDAR, using only generated latents. For each modality, the student surrogate loss and auxiliary loss are each averaged over channels, spatial locations, and generated latent frames across the full rollout, including all camera views for RGB. We combine the RGB and LiDAR losses with weights $1 : 2$. During chunkwise replay, each chunk uses the same full-rollout normalization, so summing the chunk losses reproduces the full-rollout average.

Stage 1: causal teacher forcing. We use replay-based teacher forcing with gradients through the clean history representations (Sec. 3.2).

Stage 2: teacher-forcing discrete consistency distillation. We follow the TF-dCM objective and teacher update in Sec. 3.3, using the guidance scale in Tab. 13. The teacher's CFG reference branch replaces each camera's scene caption with the standard structured negative prompt,[8] while retaining the task prompt, per-camera metadata, structural controls, and recorded sensor history. Both resolutions use noise levels $(1, 15/16, 5/6, 5/8)$, with fresh Gaussian re-noising between evaluations and a final clean prediction.

Stage 3: self-forcing distribution matching distillation. Both score models retain bidirectional temporal visibility within each view and the teacher's past 0.4-second cross-view and cross-sensor window. We use the final-step gradient estimator in Sec. 3.4, with detached history. The longer phase keeps persistent keys and values for completed chunks to avoid repeatedly encoding the same history.

D.3.3. Training Infrastructure

Sequence packing. The 1.5M-token packing budget in Sec. 7.4.1 includes RGB, WSM, LiDAR, HD-map, and language tokens. For example, 7 720p views with 397 RGB frames per view, 133 LiDAR sweeps, and their corresponding controls occupy approximately 1.53M tokens before language conditioning. Stages 1 and 2 can pack multiple clips per sequence, whereas Stage 3 uses 1 clip per sequence.

Parallelism and memory. We distribute each packed sequence across 32 GPUs with context parallelism. On GB200 GPUs, the long-sequence configuration combines this with 64-way fully sharded data parallelism (FSDP) (Zhao et al., 2023). Context parallelism partitions each packed sequence, while FSDP shards parameters, gradients, and optimizer state over overlapping rank groups. We use BF16 computation and gradient reduction with FP32 master parameters. Full activation recomputation reduces retained intermediate tensors at the cost of additional computation, without CPU activation or parameter offloading in this configuration.

Input pipeline and checkpointing. Asynchronous batch construction prepares the next packed batch while the model processes the current one, with bounded prefetching to limit host memory use. Distributed checkpointing stages a consistent snapshot in CPU memory before writing it in a background process, allowing storage I/O to overlap subsequent training.

[8] See `github.com/NVIDIA/cosmos-framework/blob/main/cosmos_framework/inference/defaults/neg_prompts.json`.

Table D.6: **Inference optimizations of the Cosmos3-Nano-Sim-Auto runtime.** Option is the Interactive Drive launcher flag or, for the attention options that the launcher does not expose, the backend's rollout option, and Shipped gives the launcher's setting, where with LiDAR means on in LiDAR sessions.

Optimization	Stage	Mechanism	Option	Shipped
Compiled denoising	Denoise	The 4 denoising steps run under `torch.compile` with fixed shapes	`-compile-denoise`	on
Compiled control encoding	Control prep.	Compiled Wan 2.2 VAE encoder of the drawn WSM controls	`-compile-control-encode`	on
Compiled decoder	Decode	The TAEHV decoder runs under `torch.compile`	`-decoder-compile`	on
Parallel decoding	Decode	Each rank decodes its cameras and sends the pixels to rank 0	`-parallel-decode`	on
Map GPUs	Control prep.	A map GPU per model rank prepares the next chunk's WSM controls, which lag the input by 1 chunk	`-prefetch-controls`, `-prefetch-controls-gpu-offset`	on for 4+ nodes
Context parallelism	All passes	Contiguous token range per rank; keys, values, and flow are all-gathered	ranks per launch	on
Tensor parallelism	All passes	Head-group sharding with an all-reduce per layer; excludes compiled denoising	`COSMOS3_NANO_SIM_AUTO_TENSOR_PARALLEL`	off
Maskless attention	All passes	Flash attention per gathered key group, merged as in Fig. 21, instead of a masked pass	`maskless_attention`	on
Fused group merge	Eager passes	One kernel merges the key groups outside compiled denoising	`fused_attention_merge`	on
Fused RMSNorm	All passes	Fused kernel instead of the checkpoint's RMSNorm	`accelerated_rms_norm`	off
FlexAttention mask	All passes	Block-sparse masked pass instead of maskless attention	`flex_attention`	off
LiDAR rank roles	Control prep., decode	Camera-free ranks render the 4 sweeps; the first runs the LiDAR codec	with `-lidar`	with LiDAR
LiDAR range culling	Control prep.	Sweep renderer culls returns outside the sensor range	`-culled-lidar-renderer`	with LiDAR
LiDAR TF32 codec	Control prep., decode	The LiDAR codec runs its matmuls in TF32	`-lidar-codec-tf32`	with LiDAR
LiDAR contiguous projections	Control prep., decode	Contiguous inputs to the codec's linear layers	`-lidar-contiguous-linears`	with LiDAR

D.4. Runtime Optimization

Execution configuration. With the generation settings in Sec. 7.4.2, each chunk performs 4 denoising steps followed by a clean pass that writes its keys and values for subsequent chunks. The K/V cache keeps the prompt and the opening frame with its map as a permanent sink, 1 chunk of clean generated history, and the controls of that chunk and of the chunk being generated. Together with the cross-sensor window in Sec. 7.1, this keeps the attention context fixed from the second chunk onward. The GB200 nodes contain 4 GPUs, assigned to model ranks for generation or map ranks for control preparation. Tab. D.6 lists the optimization settings evaluated in App. D.4.1.

Table D.7: **Cosmos3-Nano-Sim-Auto chunk wall time and presented FPS on GB200, eager against the shipped optimizations.** Each row compares configurations within the same GPU allocation. Chunk wall time is the median of run medians of the recorded generation call on the primary rank, with the range across repeated runs in parentheses. The eager and shipped runs of a row share one allocation, and the eager run of the 7-camera map layout uses all 16 GPUs as model ranks. Change is the shipped median relative to the eager median, so -30% is a 30% shorter chunk. A dash marks a configuration without an eager run. FPS is the median rate presented to the client and Mem. the median peak device memory of the primary rank's GPU over the 1 to 7 shipped runs of the row, out of 184 GiB per GPU.

Views	Mode	GPUs	Layout	Chunk wall time (ms)		Change	FPS	Mem.
				eager	shipped			GiB
Single GPU								
1	RGB	1	1 model	524 (512–535)	364 (362–423)	−30%	32.6	33.6
4	RGB	1	1 model	1,457	1,166	−20%	10.2	37.6
Multi-GPU								
1	RGB	2	1 model + 1 map	—	315	—	38.1	31.2
4	RGB	8	8 model	783	483	−38%	24.6	41.4
4	RGB	16	8 model + 8 map	—	408 (377–440)	—	28.9	40.9
4	RGB + LiDAR	8	8 model	—	651	—	18.1	51.6
7	RGB	16	16 model	897	609	−32%	19.5	49.0
7	RGB	16	8 model + 8 map	950 (936–1,014)	583 (561–597)	−39%	20.5	48.2
7	RGB	32	16 model + 16 map	—	511 (510–511)	—	23.2	48.2
7	RGB + LiDAR	16	16 model	1,159	701 (693–714)	−39%	16.7	61.6

Compilation. Rotary-embedding and gated-activation kernels are always compiled, in addition to the optional compiled stages in Tab. D.6. The K/V cache update runs eagerly, and compiled denoising is disabled with tensor parallelism because its all-reduce would fall inside the compiled region. Interactive Drive uses no CUDA-graph replay, decodes on the model ranks rather than on a separate decoder GPU, and runs the K/V cache update and decoding sequentially.

Parallel control preparation and decoding. The runtime specializes the parallelism in Sec. 4.2 with two rank axes. Context parallelism partitions the view-major query sequence into contiguous ranges and all-gathers keys, values, and predicted flow, leaving each rank with the full cache and chunk. Optional tensor parallelism shards layers by key/value head group with an all-reduce per layer. The launcher sets tensor parallelism to 1. Cameras are assigned round-robin across ranks for drawing, encoding, and decoding, with decoded pixels collected on rank 0. During prefill, the control tracks are encoded camera by camera across ranks. A map GPU per model rank can draw and encode the next chunk's controls concurrently with denoising, introducing a 1-chunk control delay. The model ranks still execute the control-token pass that writes the control keys and values (Fig. 22).

D.4.1. Runtime Performance

Matched runtime comparisons. Which optimizations reduce streaming latency under a fixed GPU allocation? We compare the execution choices in App. D.4, varying compilation, attention backend, and the division of GPUs between model and control processing.

Measurement protocol. Each measurement series runs its configurations back to back on the same GPUs to control for node-to-node variability. Most repeated configurations varied by up to 8%, with wider ranges for the single-camera slow mode, the 4-camera map layout, and the 7-camera configurations on 32 GPUs. We interpret differences against the observed repeat range and do not attribute changes below 8% to an optimization.

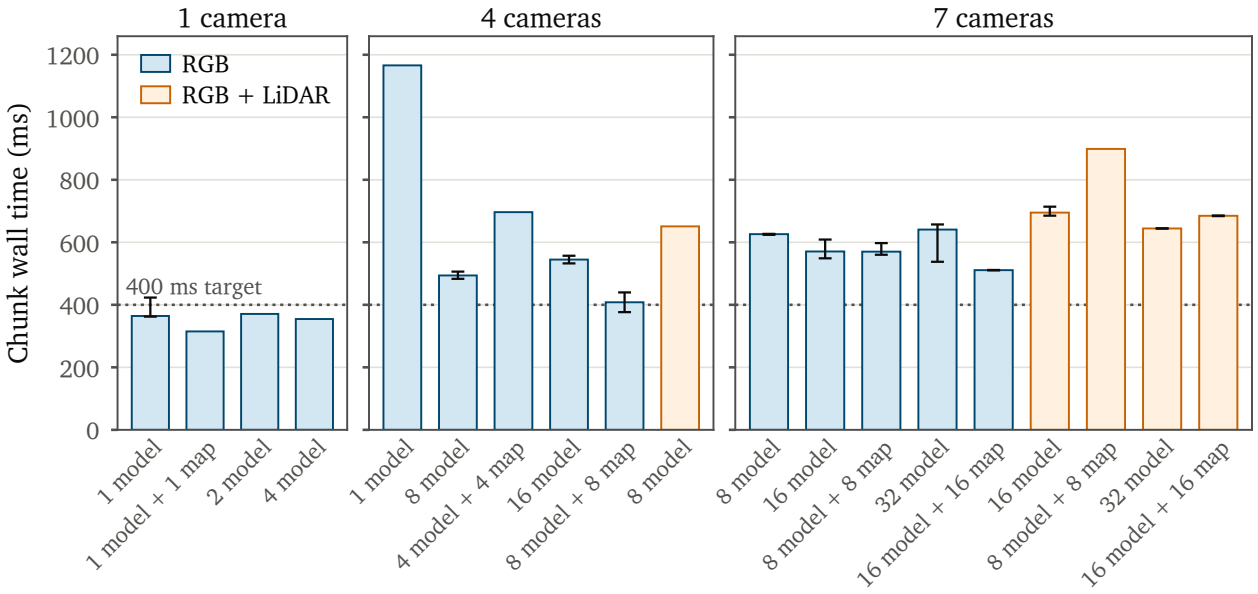


Figure D.4: **Cosmos3-Nano-Sim-Auto chunk wall time by GPU count and layout on GB200.** Shipped flags, from the same-allocation ladders of Tab. D.7; bars give the median of the run medians of a configuration and whiskers their range where the configuration was repeated. Labels give the GPU layout as model ranks plus map GPUs, where map GPUs draw and encode controls on separate GPUs, so the GPU count of a configuration is their sum; blue bars are RGB and orange bars RGB + LiDAR.

Single GPU. A single GB200 exceeds the 30-FPS video rate at the median for 1 camera, while 4 cameras require more computation than it can sustain at that rate (Tab. D.7). For 1 camera, adding model ranks provides little benefit within the repeat range, whereas assigning a second GPU to control preparation reduces latency more clearly (Fig. D.4). A slower mode appeared in some repeats on one node but did not recur on another, motivating comparisons on the same allocation.

Multi-GPU. Control prefetching helps when enough model ranks remain to process the sensor tokens efficiently. On 16 GPUs, splitting ranks equally between generation and control preparation improves 4-camera RGB latency, while the 7-camera layouts overlap within the repeat range (Fig. D.4). With only 8 GPUs, diverting half the ranks to controls instead slows generation. On 32 GPUs, the split layout improves 7-camera latency over the 16-GPU split layout, whereas adding model ranks alone does not produce a monotonic speedup. An eager 7-camera comparison also found that tensor parallelism of 2 lengthens chunks by 28% relative to 1, supporting the launcher's setting.

Joint LiDAR generation changes this allocation trade-off by adding sensor tokens and control processing. For 7 cameras with LiDAR, dedicating all 16 GPUs to model ranks outperforms an equal model/map split, while the tested 32-GPU layouts offer no improvement beyond the repeat range. The runtime therefore retains all GPUs as model ranks for LiDAR sessions. Among the LiDAR-specific options, range culling has the largest measured effect: disabling it increases chunk latency by 41%. These results favor selecting GPU roles for the sensor workload rather than increasing parallelism uniformly.

Measured attention variants. Maskless attention improves complete-chunk latency across the tested eager and compiled configurations. We compare the flash scaled dot-product attention backend with fused group merging against block-sparse FlexAttention and dense masked cuDNN, placing each variant between 2 default

Table D.8: **Chunk latency quantiles for selected Cosmos3-Nano-Sim-Auto configurations on GB200.** The p50 and p95 columns report the median across runs of each run's corresponding chunk-latency quantile. Chunk counts sum steady-state observations after discarding 10 warm-up chunks per run, excluding Nsight runs. All GPUs serve as model ranks, with compilation and parallel decoding enabled. Wall time covers the generation call on the primary rank and includes output visualization in joint RGB–LiDAR runs.

Views	GPUs	Mode	Chunk wall time (ms)		Chunks	Runs
			p50	p95		
1	1	RGB	433	453	700	5
1	1	RGB + LiDAR	1,535	1,690	140	1
1	2	RGB + LiDAR	927	1,029	140	1
4	8	RGB	503	535	280	2
4	8	RGB + LiDAR	655	693	140	1
7	16	RGB	598	627	280	2
7	16	RGB + LiDAR	742	825	140	1

runs on the same GPUs. The comparison covers both attention execution and group merging, since the masked paths replace the separate groups with a single pass. For 1 camera, the eager masked paths increase latency by 33% and 44%, respectively, while dense masking increases eager 7-camera latency by 143% on 16 model ranks. The gap is largest with compiled denoising on 8 model ranks and 8 map GPUs, where FlexAttention and dense masking increase latency by 195% and 262%. Compilation also reduces the value of a separate fused RMSNorm kernel: its 12–15% gains in eager execution fall within the repeat range once denoising is compiled. The runtime uses maskless attention with fused group merging and retains the checkpoint's RMSNorm.

Stage and startup timing. Compilation trades faster steady-state generation for longer startup, even with a warm compiler cache. Process-to-first-frame time is 42 s for 1 camera on 1 GPU and 74 s for 7 cameras on 8 model ranks + 8 map GPUs, compared with 21 and 37 s for eager execution on 1 GPU and 16 model ranks, respectively. First launches are more expensive: the 4-camera configurations on 1 and 16 GPUs take 146 and 168 s, compared with 38–55 and 64 s on later launches, consistent with compilation for new shapes. During prefill, the transformer runs eagerly while the control encoder executes its first compiled call. It takes 13.3 s for 1 camera with compilation versus 3.4 s without it, and 19.4 s for 7 cameras on 16 GPUs.

After denoising is accelerated, control preparation and clean-history updates remain substantial costs (Fig. 22). Each costs approximately 2–4 compiled denoising steps, making them useful targets for further optimization. Parallel decoding is comparatively inexpensive at 8–10 ms per chunk, versus 26 ms when 1 GPU decodes 4 cameras. Approximately 7 ms lies outside the four measured stages. Peak device memory is reported with the corresponding operating points in Tab. D.7.

Latency variability. To characterize latency variation and small-GPU joint generation, Tab. D.8 supplements the matched comparisons with a separate series of FlashDreams 0.2 RC1 measurements on GB200. The runs use the residential scene and the resolution, chunk length, and sampling steps in Sec. 7.4.2. Configurations were repeated across allocations, with single-camera RGB run medians spanning 358–510 ms across nodes. This variation motivates measuring optimization gains back to back on the same GPUs.

Kernel profiling and scaling bottlenecks. Nsight traces of companion runs help explain why adding model ranks does not always reduce latency. For 1 camera on 1 GPU, the GPU is busy for 43% of the traced step, with attention and matrix multiplications dominating its active time. The substantial time outside GPU kernels points to host-side launch and bookkeeping as optimization targets. With 7 cameras on 16 GPUs, GPU activity rises to 72%, and collectives account for 17% of GPU time. For joint RGB–LiDAR on 16 model ranks, collectives

account for 54% of GPU time, highlighting communication as a larger cost for the additional sensor workload.

The single-camera joint profile on 2 GPUs also exposes dependencies beyond transformer execution. The second rank rasterizes, encodes, and decodes LiDAR sweeps concurrently with the primary rank, which waits for control tokens and decoded sweeps when needed. These dependencies complement the stage-level costs in Fig. 22 and motivate coordinating sensor processing with generation. Tracing adds 6% to step wall time for 7 RGB cameras on 16 GPUs and 42% for 1 camera on 1 GPU.

D.5. Inference and Benchmark Settings

D.5.1. Inference Settings

This section specifies the inference and prompt configurations used for the evaluations in Sec. 7.5.

Across both benchmarks, Cosmos3-Nano-Sim-Auto uses a rolling K/V cache of 2 chunks: the initial chunk, retained as an attention sink, and the most recent completed chunk. The one-hour rollout in Fig. 27 uses the same cache configuration.

Driving-Sim-Eval. Cosmos3-Nano-Transfer-Auto and Cosmos3-Nano-Sim-Auto jointly generate RGB and LiDAR, with 297 RGB frames per camera at 720p and 30 FPS across 7 non-fisheye views. Both use timestamped per-camera captions produced by the same captioner as their training data (Sec. 7). Cosmos3-Nano-Transfer-Auto uses guidance 3, while Cosmos3-Nano-Sim-Auto uses 4 denoising steps without classifier-free guidance (CFG). The baseline Cosmos-Transfer2.5 also generates 297 frames per camera at 720p and 30 FPS using the same 7 views, with guidance 3 and autoregressive generation in its native 29-frame chunks. Dense, untimestamped paragraphs yield its highest generation quality, with one caption per camera covering the first 10 seconds of the clip. Boxes D.3 and D.4 illustrate the timestamped and untimestamped caption formats, respectively.

Traffic-Scenario-Eval. Cosmos3-Nano-Transfer-Auto uses guidance 1.5 and jointly refined, environment-enhanced, timestamped per-camera captions. Cosmos3-Nano-Sim-Auto uses the iteration-1,800 checkpoint with 4 denoising steps, no CFG, and timestamped per-camera captions. Both generate at 30 FPS. The baseline Cosmos-Transfer2.5 uses guidance 3 and its native 10-FPS setting, which gave its best results in our tests. We tested all caption variants for the baseline and selected the original benchmark captions, which yielded the highest generation quality.

For Cosmos3-Nano-Transfer-Auto evaluation on Traffic-Scenario-Eval, we refine the timestamped per-camera captions to express the requested weather, time of day, road condition, and region through concrete visual details. We refine all seven captions jointly using their WSM controls, preserving actor appearances, trajectories, and timestamps while keeping environmental descriptions consistent across views. These practices are summarized in App. D.2.3 and provided as a prompt upsampling skill in Cosmos Framework. The negative prompt used for this evaluation is reproduced below.

Negative prompt

```
The driving-camera video has unstable shading, with faint, dirty-looking blotches drifting across the sky
and other smooth surfaces. These patches appear and disappear between frames while local brightness
fluctuates independently of scene motion and lighting. Surface textures crawl, fine details shimmer, and
object edges wobble. Dense fog or heavy rain and road spray severely obscure nearby vehicles and lane
markings, while persistent water films on the lens conceal large portions of the scene. Motion is
excessively smeared, leaving long trails and ghosted outlines that erase recognizable detail even at
moderate speeds. The image remains unnaturally soft, vehicles deform, road geometry drifts, and exposure
changes abruptly without corresponding changes in illumination. The scene has a computer-generated,
game-rendered appearance, with plastic-looking materials, unnaturally saturated colors, and artificially
exaggerated contrast. People and objects morph, merge, split, or change identity discontinuously between
frames. Human limbs distort, fuse, or detach. People or objects abruptly appear or disappear in
unobstructed areas without entering or leaving the camera view or passing behind an occluder.
```

Weather-prompt study. For the study in App. D.6.2, each rollout contains 93 RGB frames per camera at 480p and 30 FPS and 31 LiDAR sweeps at 10 Hz, using 35 UniPC steps, guidance 6, shift 10, and a fixed negative prompt.

D.5.2. Evaluation Metric Definitions

This section gives the definitions and scoring procedures for the metrics summarized in Sec. 7.5.1.

RGB video quality metrics. We measure overall fidelity, temporal consistency, and multi-view consistency. Single-view metrics are reported as the average across the 7 non-fisheye views. Fisheyes are excluded for comparison against Cosmos-Transfer2.5.

Frame FID (Heusel et al., 2017), I3D FVD (Unterthiner et al., 2019), StyleGAN-V FVD (Skorokhodov et al., 2022), and the DOVER (Wu et al., 2023) technical score cover overall fidelity.

Adjacent cosine mean (ACM) measures temporal visual consistency similarly to VBench (Huang et al., 2024). We extract CLIP ViT-B/32 (Radford et al., 2021) features $\mathbf{f}_t$ for frames $t = 0, \ldots, T-1$, then average feature similarities to the preceding and first frames:

$$\mathrm{ACM} = \frac{1}{T-1} \sum_{t=1}^{T-1} \frac{1}{2} \big[\cos^+(\mathbf{f}_t, \mathbf{f}_{t-1}) + \cos^+(\mathbf{f}_t, \mathbf{f}_0) \big], \tag{D.11}$$

where $\cos^+(\mathbf{a}, \mathbf{b}) = \max\big(0, \cos(\mathbf{a}, \mathbf{b})\big)$ is cosine similarity clamped at zero.

We measure pixelwise multi-view consistency primarily by cross-view Sampson error (CSE) (Sampson, 1982). For each pair of overlapping views (v, w) at time t, we extract keypoints with SuperPoint (DeTone et al., 2018) and match them with LightGlue (Lindenberger et al., 2023) to obtain point correspondences $M^t_{v,w} = \{(\mathbf{x}, \mathbf{x}')\}$. Undistorted keypoints are used to fit a fundamental matrix F with RANSAC and the eight-point algorithm. Total cross-view Sampson error for a single view pair (v, w) is calculated as the average square-root Sampson error over all frames T.

$$\mathrm{CSE}_{v,w} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|M^t_{v,w}|} \sum_{(\mathbf{x},\mathbf{x}') \in M^t_{v,w}} d_F(\mathbf{x}, \mathbf{x}'), \tag{D.12}$$

where $d_F(\mathbf{x}, \mathbf{x}')$ is the Sampson distance (Sampson, 1982), the square root of the Sampson error of the correspondence under F. We report CSE-2 and CSE-5, which are the average $\mathrm{CSE}_{v,w}$ over 2 and 5 camera pairs, respectively. CSE-2 involves the (front, front-left) and (front, front-right) cameras, while CSE-5 extends that with the (front, front telephoto), (rear-left, rear telephoto), and (rear-right, rear telephoto) pairs.

Temporal Sampson error (TSE) applies the same Sampson distance within one camera stream to measure camera consistency across time. For each pair of consecutive frames in a view, it matches keypoints, fits a fundamental matrix to the matches, and averages d_F over the matches. The result is similarly averaged over frame pairs and then over the 7 views. TSE requires care because a small baseline between frames can make F ill-posed, and the scene may change between frames.

LiDAR metrics. LiDAR evaluation covers distribution fidelity, temporal consistency, and object faithfulness to WSM controls. Real scans are projected to the generated range-view representation (Sec. 7.1) for comparable point-density patterns.

Following LiDARGen (Zyrianov et al., 2022), bird's-eye-view (BEV) histograms count points in 2 m grid cells covering ± 100 m, using sensor ranges of 5–100 m. Global BEV JSD measures Jensen–Shannon distance (Lin, 1991) between generated and real histograms summed over all frames. Local BEV JSD instead sums within each clip and averages the distances across clips, retaining scene-specific differences that dataset-level aggregation can obscure. BEV MMD^2 (squared maximum mean discrepancy) (Gretton et al., 2012) compares normalized frame histograms within each clip using a Gaussian kernel with $\sigma = 0.5$, then averages over clips.

For feature-space comparisons, we average-pool Utonia (Zhang et al., 2026c) point-cloud features into one vector per frame. Across all frames, we compute Fréchet Utonia distance (FUD), analogous to FID (Heusel et al., 2017), and unbiased Utonia MMD^2 with a cubic polynomial kernel, analogous to kernel inception distance (Bińkowski et al., 2018).

Chamfer temporal consistency (CTC) measures Chamfer distance between consecutive scans aligned using ground-truth ego poses. Vehicle CTC applies the same measure to tracked vehicle crops in object-local coordinates. Both are reported as ratios to the corresponding measures on real LiDAR, with an ideal value of 1.

Object support recall is the fraction of labeled cuboids with at least 30 real points that also contain at least 10 generated points, covering vehicles, people, and animals. Detection F1 evaluates a pre-trained PointPillars (Lang et al., 2019) detector on generated LiDAR, matching same-class detections to ground-truth objects at planar center distances below 3 m.

Traffic-object adherence. On Traffic-Scenario-Eval, we measure how well the generated objects follow the WSM controls with AutoGrader, which uses the object correspondence checks of Cosmos Evaluator (NVIDIA, 2026c). For each WSM object, the checks project its cuboid into each camera and compare it with a semantic segmentation of the generated frame, giving a per-frame correspondence score. We transform the correspondence scores into detection counts with a threshold of 0.5: an object whose mean correspondence score is at least 0.5 counts as detected, and otherwise as missing. Segmented objects that match no WSM object count as hallucinated. We report F1 for vehicles, pedestrians, and cyclists, computed from these counts summed over all cameras and scenarios.

Human evaluation rubric. Video-quality review covers the full frame, including the sky and distant objects, and checks textures, colors, lighting, and geometric stability. Examples of minor artifacts allowed at score 2 include brief flicker, slight softness, and small lighting inconsistencies. Fused vehicles, floating objects, or intersecting bodies receive score 1 even when visible only briefly. Scene-element questions assess the presence, placement, appearance, and stability of the specified lanes, signals, signs, pedestrians, and bicyclists. Cross-view questions rate consistency of the same objects' identity, geometry, location, and motion: 3 for agreement, 2 for minor discrepancies, and 1 for major contradictions.

D.6. Supplementary Results and Ablations

We present ablations that inform our final attention design and training recipe, followed by studies of joint sensor generation and weather conditioning.

D.6.1. Ablation Study

Training-stage quality comparison. Which stages recover generation quality after converting the bidirectional model into a causal, few-step generator? We compare the bidirectional teacher, stage 1 TF, stage 2 TF-dCM, stage 3 SF-DMD short, and stage 3 SF-DMD long checkpoints (Sec. 7.3) on the same 50-scene validation set (Tab. D.9). All evaluations use seven cameras, 720p RGB, 10-second clips, and no rolling KV cache. The bidirectional teacher and stage 1 TF use 35 UniPC steps with guidance 3, while stage 2 TF-dCM and both stage 3 SF-DMD checkpoints use 4 SDE steps with no guidance. Step distillation in stage 2 TF-dCM reduces visual fidelity and geometric consistency relative to stage 1 TF. Stage 3 SF-DMD recovers much of the lost generation quality, and its long-sequence phase further improves FID, FVD, temporal consistency, and cross-view consistency. These results support retaining both SF-DMD phases in the final training recipe.

Cross-view temporal visibility. Which cross-view temporal window best preserves scene consistency? We compare three cross-view visibility rules from the same bidirectional starting point using the same training budget. A permits the closest other-view latent frame, B permits the inclusive preceding 0.4 seconds, and C adds the current jointly denoised noisy chunk to B. Although C may attend to future noisy frames within the chunk, it does not expose future chunks or future clean targets. The rule applies in both training and

Table D.9: **Training-stage quality comparison.** All checkpoints are evaluated on the same 50-sample validation set. Stage 3 SF-DMD improves FID, FVD, and geometric consistency over stage 2 TF-dCM with the same 4-step sampler. Bold and underlined values mark the best and second-best scores among the causal models, with the bidirectional teacher as a reference. Arrows indicate the preferred score direction.

	Video quality			Temporal consistency	Cross-view alignment	
Checkpoint	**Frame FID ↓**	**FVD I3D ↓**	**DOVER tech. ↑**	**ACM ↑**	**CSE-2 ↓**	**CSE-5 ↓**
Bidirectional teacher	40.61	352.6	7.472	0.8988	2.867	2.273
Stage 1 TF	57.08	551.8	<u>7.579</u>	0.8807	4.732	3.669
Stage 2 TF-dCM	75.34	767.3	7.486	0.8638	7.547	4.562
Stage 3 SF-DMD short	<u>48.93</u>	<u>506.9</u>	**7.847**	<u>0.8902</u>	<u>2.971</u>	<u>2.377</u>
Stage 3 SF-DMD long	**45.69**	**435.2**	7.366	**0.8957**	**2.795**	**2.302**

Table D.10: **Cross-view temporal visibility.** All variants are evaluated on the same 10-sample validation subset. Allowing the past 0.4 seconds achieves the best result by substantially reducing cross-view error. Adding the current noisy chunk provides no further improvement here. Arrows indicate score directions, with the best and second-best values bold and underlined.

	Video quality			Temporal consistency	Cross-view alignment		
Visibility	**Frame FID ↓**	**FVD I3D ↓**	**DOVER tech. ↑**	**ACM ↑**	**CSE-2 ↓**	**CSE-5 ↓**	**Pose (rad) ↓**
A: Nearest-time slice	**98.74**	<u>1244.0</u>	**8.385**	**0.8815**	10.585	6.793	<u>0.194</u>
B: Past 0.4 s	<u>102.61</u>	**1196.7**	<u>8.312</u>	<u>0.8770</u>	**5.743**	**4.685**	**0.188**
C: B + current chunk	109.99	1334.2	8.134	0.8732	<u>7.854</u>	<u>6.539</u>	0.252

generation. Tab. D.10 scores the full 10-second outputs on a subset of 10 validation scenes. Pose denotes rotation error in radians, and DOVER denotes technical quality. The preceding 0.4-second window yields the best overall result, substantially reducing cross-view error. Adding the current noisy chunk offers no further benefit. We therefore use the preceding 0.4-second window for cross-view attention.

Restricting cross-view attention. Our cross-view attention allows each camera stream to attend to all other selected views. A possible way to accelerate generation is to restrict these connections to neighboring cameras or cameras with overlapping fields of view, using the camera rig's spatial arrangement and FOVs. Tab. D.11 compares this restricted scheme with dense all-view attention in stage 1 TF on the same validation set. We observe no speedup with this scheme in our implementation, consistent with overhead from fragmented attention masks. Despite slightly improved video quality and temporal consistency on par with dense all-view attention, the neighboring and overlapping-view scheme yields substantially worse cross-view consistency. We therefore retain dense all-view attention.

Deduplicating same-view keys. Does duplicated same-view evidence impair cross-view consistency? When the same-stream and cross-sensor groups share keys, those keys contribute twice to the joint softmax. Removing the query's own stream from the cross-sensor group improved cross-view consistency in our experiments, supporting the disjoint grouping used in the final model (App. D.1.2).

Memory and long rollouts. How much cached history is needed to sustain coherent long rollouts? We vary the cache size C and visual attention sink size S, measured in chunks of three latent frames, for our final model on an evaluation set of 94 videos that are around one minute long (Fig. D.5). We use time-aligned source captions for each camera view. Our model switches captions at their time boundaries without resetting the visual KV

Table D.11: **Cross-view attention schemes in stage 1 TF.** Both schemes are evaluated on the same 50-sample validation set. The neighboring and overlapping-view scheme substantially degrades cross-view consistency despite small improvements in visual quality. Arrows indicate the preferred score direction. Bold and underlined values mark the better and second score when their difference is at least 1%.

	Video quality			Temporal consistency	Cross-view alignment	
Attention scheme	**Frame FID ↓**	**FVD I3D ↓**	**DOVER tech. ↑**	**ACM ↑**	**CSE-2 ↓**	**CSE-5 ↓**
Dense all-view	<u>61.28</u>	<u>546.9</u>	<u>6.777</u>	0.8786	**5.503**	**4.046**
Neighboring + overlapping	**56.17**	**511.3**	**7.331**	0.8833	<u>6.823</u>	<u>4.755</u>

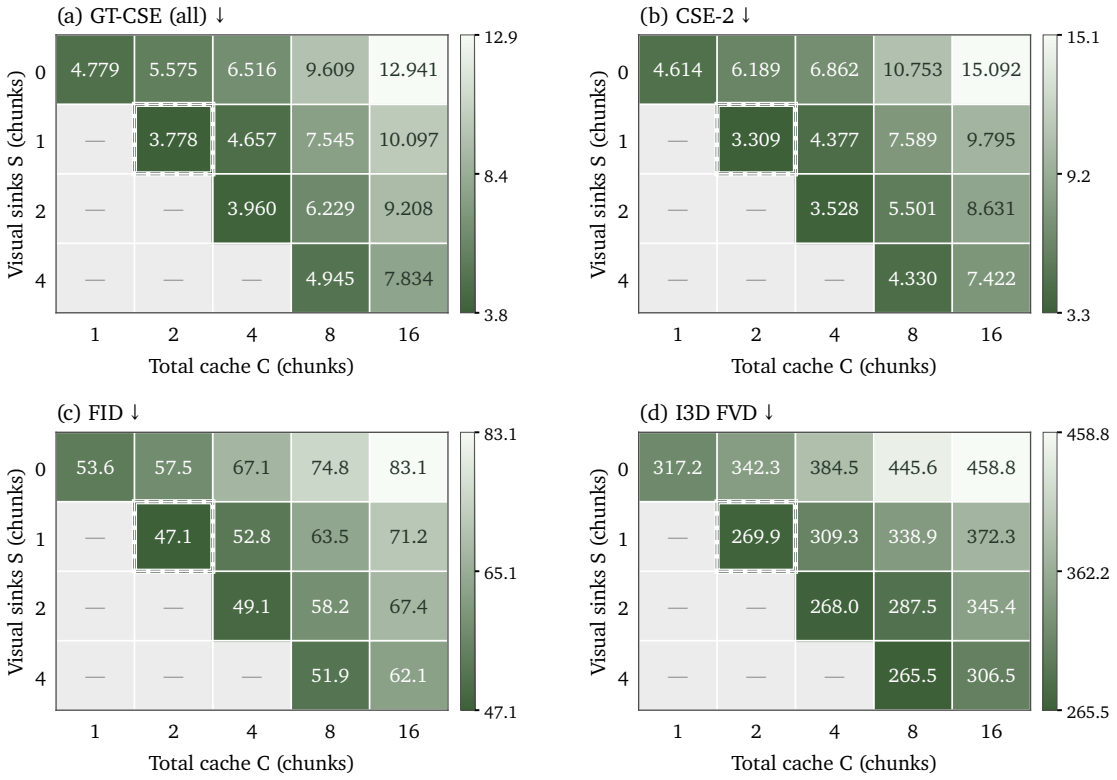


Figure D.5: **Cache and sink sizes for minute-long rollouts.** The outlined $C = 2, S = 1$ setting achieves the best trade-off overall. Darker green is better within each panel.

cache, while retaining four initial text tokens per view as language sinks. The total cache size includes the visual sinks. The $C = 2, S = 1$ setting achieves the best trade-off overall, motivating the two-chunk cache with one attention-sink chunk used in our evaluation. At a fixed one-chunk sink, retaining more history worsens both distributional quality and cross-view consistency. This result supports assigning the cache to an initial appearance anchor and recent motion, even though training includes sequences of up to 34 chunks (Tab. D.5). Training horizon and inference cache size serve different purposes: a long self-forcing rollout exposes the student to successive predictions conditioned on its own outputs, while the cache selects the observations directly available to each prediction.

The one-hour experiment in Fig. 27 uses this compact cache while cycling the scene controls and captions. The most recent generated chunk becomes context for the next, so its appearance and any prediction errors can propagate across successive cache replacements. Stable scores over the hour characterize this continuing

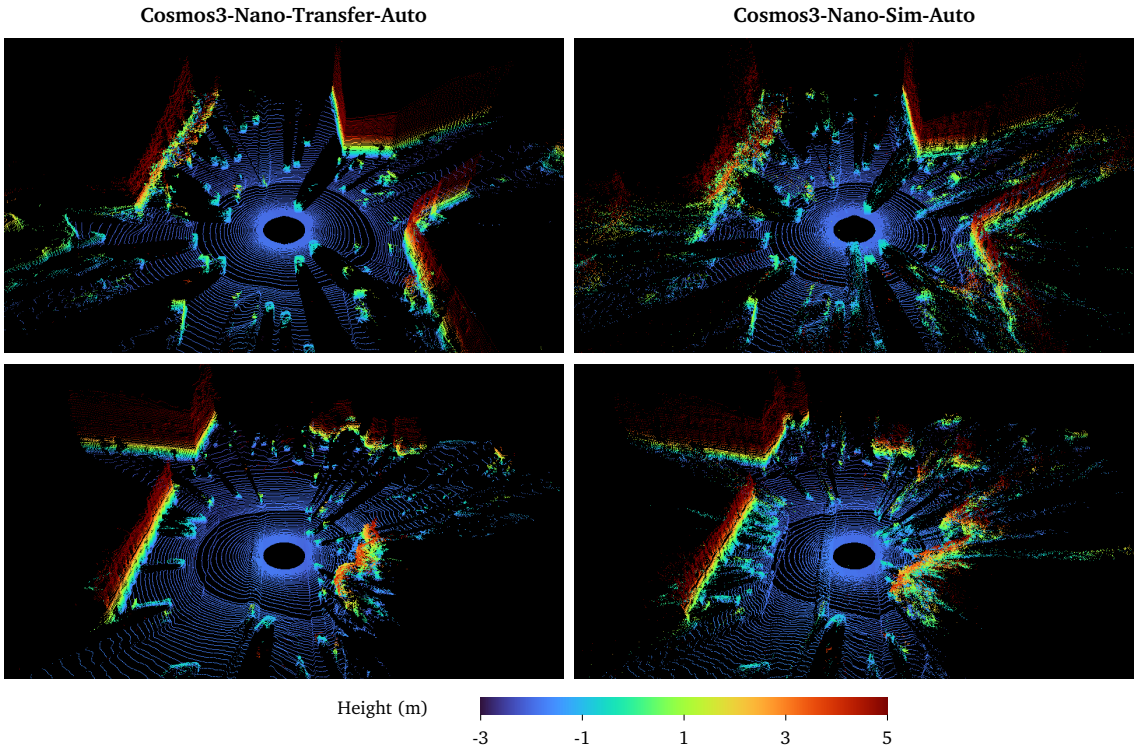


Figure D.6: **LiDAR geometry with Cosmos3-Nano-Transfer-Auto and Cosmos3-Nano-Sim-Auto.** Scenes in Fig. 26 show Transfer (left) and Sim (right) at 4.9 s with matched viewpoints and a shared height scale.

generation process with an initial visual anchor and recurring structural controls. For simulator-driven use, the evolving geometry and actor motion are supplied by the external scenario state through WSM controls, while cached generated observations maintain visual continuity.

D.6.2. LiDAR Generation Case Study

LiDAR quality comparison between Transfer-Auto and Sim-Auto. Fig. D.6 compares the two models on the same scenes and time point as the ground-truth comparison in Fig. 26. Both models receive the first RGB frame from each of 11 cameras and no measured LiDAR sweep. Cosmos3-Nano-Transfer-Auto uses 35 denoising steps and guidance 1.5, while Cosmos3-Nano-Sim-Auto uses 4 steps and guidance 1. The four-step causal model preserves the main road and roadside geometry, but its returns are more scattered and object boundaries less sharply defined than those of the bidirectional model. This contrast matches the LiDAR fidelity gap in Tab. 16. Improving multimodal causal distillation to preserve fine LiDAR geometry remains a focus of future work.

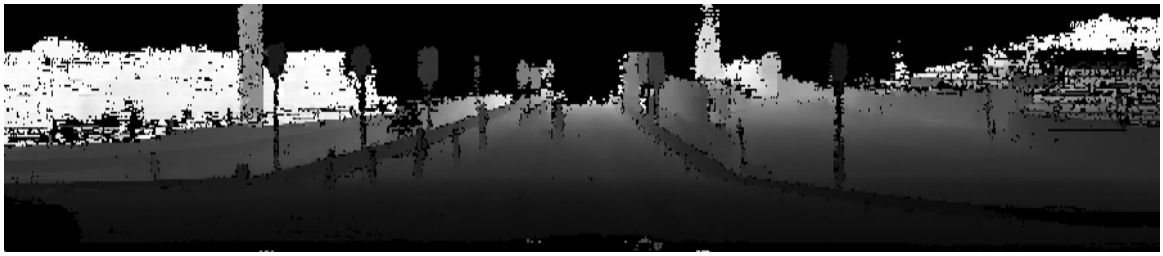
Weather-prompt effects on LiDAR. To test whether weather prompts affect LiDAR as well as RGB, we vary only the weather sentence in all 11 camera captions for one highway scene, keeping the WSM and HD-map controls fixed. Cosmos3-Nano-Transfer-Auto generates 30 joint rollouts across 10 weather conditions and 3 shared seeds (2025, 2026, and 2027). We use the inference settings documented in App. D.5.1. Valid-return fraction measures the proportion of nonzero pixels in the 128×1800 range view, averaged over each run's 31 sweeps. Tab. D.12 reports its mean and standard deviation across seeds, together with mean return intensity.

Rain reduces the mean valid-return fraction from 0.587 to 0.434 (26%), while drizzle, stormy weather, fog, and hurricanes reduce it by 18–19%. All three seeds under these five conditions produce fewer returns than the sparsest sunny run, whereas the remaining conditions overlap the sunny seed range. Rain reduces near-road returns, while fog suppresses distant returns beyond 50 m (Fig. D.7). Mean intensity decreases under wet-weather prompts and increases under fog, snow, and sandstorm prompts.

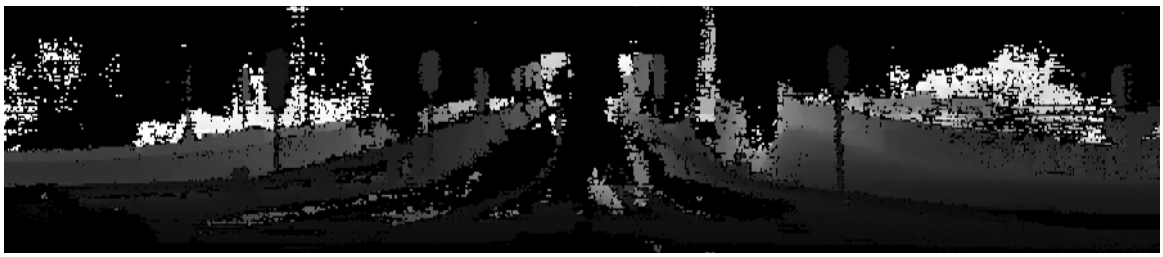
Table D.12: **Weather prompts change generated LiDAR returns.** Cosmos3-Nano-Transfer-Auto receives the same scene controls under all 10 weather conditions. Valid-return fractions are averaged over 31 sweeps per run and reported as mean ± standard deviation across 3 seeds. Ratios compare mean valid-return fractions with sunny conditions.

Weather	Valid-return fraction	Ratio to sunny	Mean intensity
Sunny	0.587 ± 0.040	1.00	0.128
Cloudy	0.558 ± 0.032	0.95	0.131
Overcast	0.549 ± 0.016	0.94	0.124
Rain	0.434 ± 0.018	0.74	0.078
Drizzle	0.480 ± 0.019	0.82	0.097
Snow	0.543 ± 0.028	0.93	0.157
Stormy	0.478 ± 0.023	0.82	0.078
Fog	0.478 ± 0.005	0.82	0.169
Hurricanes	0.476 ± 0.023	0.81	0.081
Sandstorms	0.541 ± 0.022	0.92	0.158

(a) Sunny



(b) Rain



(c) Fog

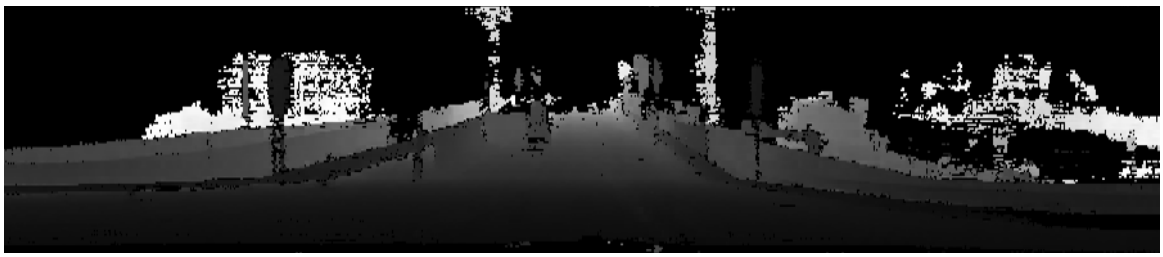


Figure D.7: **LiDAR range views under different weather prompts.** The front $\pm 90^\circ$ of sweep 15 is shown for the same scene controls and random seed (2025). Brighter pixels indicate greater range on a shared 0–100 m scale, and black pixels denote missing returns. Compared with sunny conditions, rain produces gaps in road-surface returns, while fog suppresses distant returns and preserves much of the nearby road.

For pixels valid in both a weather run and its same-seed sunny reference, mean absolute range differences are 2.8–5.1 m, below the 5.9–6.7 m differences between sunny seeds. In this scene, weather prompts therefore

change return coverage and intensity while shared-return geometry remains comparatively stable. This response suggests a path toward more realistic joint RGB–LiDAR simulation, with environmental prompts shaping both appearance and sensor returns.

References

[1] Krish Agarwal, Zhuoming Chen, Cheng Luo, Yongqi Chen, Haizhong Zheng, Xun Huang, Atri Rudra, and Beidi Chen. MonarchRT: Efficient Attention for Real-Time Video Generation. In *NeurIPS*, 2026. URL https://www.infini-ai-lab.cmu.edu/. To appear. 63

[2] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. 61

[3] Sherwin Bahmani, Tianchang Shen, Jiawei Ren, Jiahui Huang, Yifeng Jiang, Haithem Turki, Andrea Tagliasacchi, David B. Lindell, Zan Gojcic, Sanja Fidler, et al. Lyra: Generative 3D scene reconstruction via video diffusion model self-distillation. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/85e446f011bd3b6ee382ef2a7e47f20e-Abstract-Conference.html. 61

[4] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. 6, 32

[5] Aarti Basant, Amlan Kar, Despoina Paschalidou, Fangyin Wei, Francesco Ferroni, Guillermo Garcia Cobo, Haithem Turki, Huan Ling, Jaewoo Seo, James Lucas, et al. NVIDIA OmniDreams: Real-Time Generative World Model for Closed-Loop Autonomous Vehicle Simulation. *arXiv preprint arXiv:2606.03159*, 2026. URL https://arxiv.org/abs/2606.03159. 4, 44, 47, 62, 63

[6] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1801.01401. 103

[7] Ollin Boer Bohan. TAEHV: Tiny AutoEncoder for Hunyuan Video. https://github.com/madebyollin/taehv, 2025. 20

[8] 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 *ICML*, 2024. 61

[9] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *IROS*, 2025. 22, 27, 28, 35, 66

[10] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the Kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6299–6308, July 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Carreira_Quo_Vadis_Action_CVPR_2017_paper.html. 54

[11] Henry Che, Xuanchi Ren, Sanja Fidler, Shenlong Wang, Laura Leal-Taixé, and Shengyu Huang. Into the LidarVerse: A real-time unified LiDAR world model across tasks and sensors. *arXiv preprint arXiv:2610.19201*, 2026. URL https://arxiv.org/abs/2610.19201. 43, 58, 85, 87

[12] Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. In *NeurIPS*, 2024. doi: 10.52202/079017-0759. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2aee1c4159e48407d68fe16ae8e6e49e-Abstract-Conference.html. 62

[13] Sili Chen, Hengkai Guo, Shengnan Zhu, Feihu Zhang, Zilong Huang, Jiashi Feng, and Bingyi Kang. Video Depth Anything: Consistent depth estimation for super-long videos. In *CVPR*, 2025. 32, 72, 73

[14] Yukang Chen, Luozhou Wang, Wei Huang, Shuai Yang, Bohan Zhang, Yicheng Xiao, Ruihang Chu, Weian Mao, Qixin Hu, Shaoteng Liu, et al. LongLive-2.0: An NVFP4 Parallel Infrastructure for Long Video Generation. *arXiv preprint arXiv:2605.18739*, 2026a. URL https://arxiv.org/abs/2605.18739. 63, 70

[15] Zhifei Chen, Luozhou Wang, Guibao Shen, Dongyu Yan, Shuai Yang, Tianshuo Xu, Yihua Du, Wei Wang, Tianyi Gui, Lianghua Huang, and Yingcong Chen. ReWorld: An Interactive World Model with Long-Horizon Memory. *arXiv preprint arXiv:2608.23565*, 2026b. doi: 10.48550/arXiv.2608.23565. URL https://arxiv.org/abs/2608.23565. 63

[16] Ke Cheng, Hanqiao Ye, Lei Shi, Yahui Liu, Yunhan Shen, Jingtao Dong, Zhenke Wang, Wenxuan Ao, Weixiang Xu, Kaining Huang, and Shuhan Shen. M^{4}World: A Multi-view Multimodal Driving World Model for Interactive Object Manipulation and Minute-long Streaming. *arXiv preprint arXiv:2607.14005*, 2026. URL https://arxiv.org/abs/2607.14005. 62

[17] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025. 5

[18] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *CVPR Workshops*, 2018. 102

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. URL https://arxiv.org/abs/2010.11929. 6

[20] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552. 73

[21] Yao Feng, Chendong Xiang, Xinyi Mao, Hengkai Tan, Zuyue Zhang, Shuhe Huang, Kaiwen Zheng, Haitian Liu, Hang Su, and Jun Zhu. Vidarc: Embodied video diffusion model for closed-loop control. *arXiv preprint arXiv:2512.17661*, 2025. URL https://arxiv.org/abs/2512.17661. 62

[22] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017. doi: 10.1109/ICRA.2017.7989324. URL https://doi.org/10.1109/ICRA.2017.7989324. 61

[23] Ruiyuan Gao, Kai Chen, Enze Xie, Lanqing Hong, Zhenguo Li, Dit-Yan Yeung, and Qiang Xu. MagicDrive: Street View Generation with Diverse 3D Geometry Control. In *ICLR*, 2024. URL https://openreview.net/forum?id=sBQwvucduK. 61

[24] Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. DreamDojo: A generalist robot world model from large-scale human videos. In *ICML*, 2026a. URL https://proceedings.mlr.press/v306/gao26q.html. 4, 27, 28, 61

[25] Zelin Gao, Qiuyu Wang, Jiapeng Zhu, Jingye Chen, Zichen Liu, Qingyan Bai, Jiahao Wang, Yufeng Yuan, Hanlin Wang, Yichong Lu, Ka Leong Cheng, Haojie Zhang, Jian Gao, Tianrui Feng, Yuzheng Liu, Yao Yao, Yinghao Xu, Xing Zhu, Yujun Shen, and Hao Ouyang. Infinite Worlds with Versatile Interactions. *arXiv preprint arXiv:2607.07534*, 2026b. URL https://arxiv.org/abs/2607.07534. 26, 27, 61

[26] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-Exo4D: Understanding skilled human activity from first- and third-person perspectives. In *CVPR*, 2024. 35

[27] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 13(25):723–773, 2012. URL https://www.jmlr.org/papers/v13/gretton12a.html. 54, 102

[28] Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4D World Action Modeling from Video Priors with Asynchronous Denoising. *arXiv preprint arXiv:2604.26694*, 2026a. doi: 10.48550/arXiv.2604.26694. URL https://arxiv.org/abs/2604.26694. 62

[29] Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-World: A controllable generative world model for robot manipulation. In *ICLR*, 2026b. 62

[30] Ahmet H. Güzel, Jenny Seidenschwarz, Benjamin Graham, Jonathan Sadeghi, Jeffrey Hawke, and Ilija Bogunovic. PROWL: Prioritized Regret-Driven Optimization for World Model Learning. *arXiv preprint arXiv:2605.18803*, 2026. doi: 10.48550/arXiv.2605.18803. URL https://arxiv.org/abs/2605.18803. 62

[31] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. 61

[32] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *ICML*, 2019. 61

[33] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020. 61

[34] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *CVPR*, 2023. 85

[35] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. CameraCtrl: Enabling camera control for video diffusion models. In *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f98fd73d59d8494489ea970747b91fe4-Abstract-Conference.html. 61

[36] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In 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/8a1d694707eb0fefe65871369074926d-Paper.pdf. 54, 102, 103

[37] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI. 14, 68

[38] Boni Hu, Xiong Wei, Haoming Huang, Yong Huang, Chenbo Wang, Yi Yang, Jiancheng Wang, Ruicheng Zhu, Zhimin Yang, Guanglai Liu, Qiaowan Jin, Dongzhuo Wang, Haiwei Kuang, Jiajun Fan, Yue Wu, Jiaxin Wei, Hao Sun, Feihong Yan, Yuyao Zhou, Wei Bi, Kaixuan Wang, Zichao Guo, and Xiaozhi Chen. ZYT-World: A Real-Time Controllable World Model for Closed-Loop Autonomous-Driving Simulation. *arXiv preprint arXiv:2609.21712*, 2026. URL https://arxiv.org/abs/2609.21712. 61

[39] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. URL https://arxiv.org/abs/2106.09685. 32, 68

[40] Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, et al. ViPE: Video pose engine for 3D geometric perception. *arXiv preprint arXiv:2508.10934*, 2025a. 22, 26, 67

[41] Junchao Huang, Ziyang Ye, Xinting Hu, Tianyu He, Guiyu Zhang, Shaoshuai Shi, Jiang Bian, and Li Jiang. LIVE: Long-horizon Interactive Video World Modeling. In *ICML*, 2026a. URL https://proceedings.mlr.press/v306/huang26bt.html. 62

[42] Lianghua Huang, Zhi-Fan Wu, Yupeng Shi, Wei Wang, Mengyang Feng, Junjie He, Chen-Wei Xie, Yu Liu, Jingren Zhou, Ang Wang, Bang Zhang, Baole Ai, Chen Liang, Cheng Yu, Chongyang Zhong, Jinwei Qi, Kai Zhu, Pandeng Li, Peng Zhang, Wenyuan Zhang, Xinhua Cheng, Yitong Huang, Yun Zheng, Yuxiang Bao, Yuzheng Wang, and Zoubin Bi. Wan-Streamer v0.2: Higher Resolution, Same Latency. *arXiv preprint arXiv:2607.04443*, 2026b. URL https://arxiv.org/abs/2607.04443. 63

[43] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self Forcing: Bridging the Train-Test Gap in Autoregressive Video Diffusion. In *NeurIPS*, 2025b. doi: 10.52202/085713-5576. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f4823f831af67a3ef15e41a85434422a-Abstract-Conference.html. 4, 14, 15, 60, 62

[44] Ze Huang, Jiahui Zhang, Hairuo Liu, Chenxi Zhang, Ran Cheng, and Li Zhang. Learning Transferable Dynamics Priors from Action to World Modeling. *arXiv preprint arXiv:2606.29501*, 2026c. doi: 10.48550/arXiv.2606.29501. URL https://arxiv.org/abs/2606.29501. 61

[45] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. VBench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024. 102

[46] I2RT Robotics. YAM arm series. https://doc.i2rt.com/products/yam, n.d. Accessed: 2026-10-19. 22, 66

[47] InSpatio Team, Donghui Shen, Guofeng Zhang, Haomin Liu, Haoyu Ji, Hujun Bao, Hongjia Zhai, Jialin Liu, Jing Guo, Nan Wang, Siji Pan, Weihong Pan, Weijian Xie, Xianbin Liu, Xiaojun Xiang, Xiaoyu Zhang, Xinyu Chen, Yifu Wang, Yipeng Chen, Zhenzhou Fan, Zhewen Le, Zhichao Ye, and Ziqiang Zhao. INSPATIO-WORLD: A Real-Time 4D World Simulator via Spatiotemporal Autoregressive Modeling. *arXiv preprint arXiv:2604.07209*, 2026. doi: 10.48550/arXiv.2604.07209. URL https://arxiv.org/abs/2604.07209. 61

[48] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed Ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023. 20, 63

[49] Yuxin Jiang, Shengcong Chen, Siyuan Huang, Liliang Chen, Pengfei Zhou, Yue Liao, Xindong He, Chiming Liu, Hongsheng Li, Maoqing Yao, and Guanghui Ren. EnerVerse-AC: Envisioning embodied environments with action condition. In *NeurIPS Workshop on Embodied World Models for Decision Making*, 2025. URL https://openreview.net/forum?id=zIiV66Dimq. 62

[50] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12697–12705, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Lang_PointPillars_Fast_Encoders_for_Object_Detection_From_Point_Clouds_CVPR_2019_paper.html. 55, 103

[51] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *CoRL*, 2023. 22, 66

[52] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi: 10.1109/18.61115. URL https://doi.org/10.1109/18.61115. 102

[53] Shanchuan Lin, Ceyuan Yang, Hao He, Jianwen Jiang, Yuxi Ren, Xin Xia, Yang Zhao, Xuefeng Xiao, and Lu Jiang. Autoregressive adversarial post-training for real-time interactive video generation. *Advances in Neural Information Processing Systems*, 38:41061–41086, 2026. 62

[54] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Lindenberger_LightGlue_Local_Feature_Matching_at_Light_Speed_ICCV_2023_paper.html. 36, 102

[55] Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling Forcing: Autoregressive Long Video Diffusion in Real Time. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/935151cc6cb5d8b6816133b75233775a-Abstract-Conference.html. 62

[56] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023. URL https://arxiv.org/abs/2209.03003. 6, 11, 31, 43

[57] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. URL https://arxiv.org/abs/1711.05101. 51

[58] Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7e9c2053258b1bdd32ff2654802cd594-Abstract-Conference.html. 62

[59] Mecka AI. Mecka: The data engine for physical AI. https://www.mecka.ai/, n.d. Accessed: 2026-10-19. 22, 66

[60] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831. 36, 76

[61] Jisu Nam, Yicong Hong, Chun-Hao Paul Huang, Feng Liu, JoungBin Lee, Jiyoung Kim, Siyoon Jin, Yunsung Lee, Jaeyoon Jung, Suhwan Choi, Seungryong Kim, and Yang Zhou. WorldCam: Interactive Autoregressive 3D Gaming Worlds with Camera Pose as a Unifying Geometric Representation. *arXiv preprint arXiv:2603.16871*, 2026. doi: 10.48550/arXiv.2603.16871. URL https://arxiv.org/abs/2603.16871. 61

[62] Vinh Nguyen, Michael Carilli, Sukru Burc Eryilmaz, Vartika Singh, Michelle Lin, Natalia Gimelshein, Alban Desmaison, and Edward Yang. Accelerating PyTorch with CUDA graphs. https://pytorch.org/blog/accelerating-pytorch-with-cuda-graphs/, October 2021. October 26, 2021. Accessed October 17, 2026. 19

[63] NVIDIA. World simulation with video foundation models for physical AI. *arXiv preprint arXiv:2511.00062*, 2025a. URL https://arxiv.org/abs/2511.00062. 44, 45, 54, 61

[64] NVIDIA. Cosmos-Transfer1: Conditional world generation with adaptive multimodal control. *arXiv preprint arXiv:2503.14492*, 2025b. 35, 38, 61

[65] NVIDIA. Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575*, 2025c. 4, 60, 61

[66] NVIDIA. Cosmos 3: Omnimodal World Models for Physical AI. *arXiv preprint arXiv:2606.02800*, 2026a. URL https://arxiv.org/abs/2606.02800. 4, 5, 6, 9, 20, 21, 31, 48, 50, 55, 60, 61, 66, 87

[67] NVIDIA. LidarVerse: Public NVIDIA PAI tokenizer configuration and model card. https://huggingface.co/nvidia/lidarverse, 2026b. Tokenizer configuration: https://huggingface.co/nvidia/lidarverse/blob/main/lidarverse-tokenizer-pai/config.json. 87, 88

[68] NVIDIA. Cosmos Evaluator. https://github.com/NVIDIA/cosmos-evaluator, 2026c. Release 26.10. Accessed: 2026-10-20. 55, 103

[69] NVIDIA. NVIDIA Isaac GR00T N1.7. https://huggingface.co/nvidia/GR00T-N1.7-3B, 2026d. Model card. Accessed: 2026-10-20. 38, 76

[70] NVIDIA. NVIDIA Isaac Sim: Robotics simulation and synthetic data generation. https://developer.nvidia.com/isaac/sim, 2026e. Accessed: 2026-05-09. 36, 76

[71] NVIDIA. Omniverse RTX Renderer. https://docs.omniverse.nvidia.com/materials-and-rendering/latest/rtx-renderer.html, 2026f. Accessed: 2026-10-20. 36

[72] Amine Ouasfi, Runjia Li, Junlin Han, Eric Marchand, Philip H. S. Torr, and Adnane Boukhayma. PartiCam: Camera Controlled Video Generation with Reward Guidance. *arXiv preprint arXiv:2609.39504*, 2026. doi: 10.48550/arXiv.2609.39504. URL https://arxiv.org/abs/2609.39504. 61

[73] PyTorch Contributors. AOTInductor: Ahead-of-time compilation for PyTorch exported models. https://docs.pytorch.org/docs/2.12/user_guide/torch_compiler/torch.compiler_aot_inductor.html, 2026a. PyTorch 2.12 documentation. Accessed 2026-05-25. 19

[74] PyTorch Contributors. `torch.nn.attention.varlen`: Variable-length attention using flash attention. https://docs.pytorch.org/docs/2.12/nn.attention.varlen.html, 2026b. PyTorch 2.12 documentation. Accessed 2026-05-25. 51

[75] Runjia Qian, Zile Wang, Jihai Zhang, Kai Zou, Wei Yu, Jiaxing Li, Zexiang Liu, Yaokun Li, Fei Kang, Kaichen Huang, Mengyin An, Haobo Zhang, Biao Jiang, Jiahua Wang, Haofeng Sun, Yang Liu, and Yangguang Li. Matrix-Game 3.5: Enhancing Real-Time Streaming Interactive World Models with Patch Memory. *arXiv preprint arXiv:2608.29910*, 2026. doi: 10.48550/arXiv.2608.29910. URL https://arxiv.org/abs/2608.29910. 61, 63

[76] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 102

[77] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. SAM 2: Segment anything in images and videos. In *ICLR*, 2025. 32

[78] Xuanchi Ren, Yifan Lu, Tianshi Cao, Ruiyuan Gao, Shengyu Huang, Amirmojtaba Sabour, Tianchang Shen, Tobias Pfaff, Jay Zhangjie Wu, Runjian Chen, et al. Cosmos-Drive-Dreams: Scalable synthetic driving data generation with world foundation models. *arXiv preprint arXiv:2506.09042*, 2025a. 85

[79] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. GEN3C: 3D-informed world-consistent video generation with precise camera control. In *CVPR*, 2025b. 61

[80] Team Robbyant, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, et al. Advancing open-source world models. *arXiv preprint arXiv:2601.20540*, 2026. 61

[81] Paul D. Sampson. Fitting conic sections to "very scattered" data: An iterative refinement of the Bookstein algorithm. *Computer Graphics and Image Processing*, 18(1):97–108, 1982. doi: 10.1016/0146-664X(82)90101-0. 36, 54, 102

[82] Georgy Savva, Oscar Michel, Daohan Lu, Suppakit Waiwitlikhit, Timothy Meehan, Dhairya Mishra, Srivats Poddar, Jack Lu, and Saining Xie. Solaris: Building a Multiplayer Video World Model in Minecraft. *arXiv preprint arXiv:2602.22208*, 2026. doi: 10.48550/arXiv.2602.22208. URL https://arxiv.org/abs/2602.22208. 61

[83] Tianchang Shen, Sherwin Bahmani, Kai He, Sangeetha Grama Srinivasan, Tianshi Cao, Jiawei Ren, Ruilong Li, Zian Wang, Nicholas Sharp, Zan Gojcic, et al. Lyra 2.0: Explorable generative 3D worlds. Accepted to SIGGRAPH Asia, 2026. URL https://nv-tlabs.github.io/Project-Lyra/. 61

[84] Joonghyuk Shin, Zhengqi Li, Richard Zhang, Jun-Yan Zhu, Jaesik Park, Eli Shechtman, and Xun Huang. MotionStream: Real-Time Video Generation with Interactive Motion Controls. In *ICLR*, 2026. URL https://openreview.net/forum?id=v1DKz5Vxr7. Preprint first released in November 2025, arXiv:2511.01266. 63

[85] Prajwal Singh, Arjun Badola, Seema Kumari, Hajime Nagahara, and Shanmuganathan Raman. Stabilizing Camera-Controlled Novel View Synthesis at Inference Time. *arXiv preprint arXiv:2609.03639*, 2026. doi: 10.48550/arXiv.2609.03639. URL https://arxiv.org/abs/2609.03639. 61

[86] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. StyleGAN-V: A continuous video generator with the price, image quality and perks of StyleGAN2. In *CVPR*, 2022. 54, 102

[87] Chenxi Song, Yanming Yang, and Chi Zhang. World in World: Explore the World with World Models. *arXiv preprint arXiv:2609.11548*, 2026. doi: 10.48550/arXiv.2609.11548. URL https://arxiv.org/abs/2609.11548. 61

[88] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency Models. In *ICML*, 2023. URL https://proceedings.mlr.press/v202/song23a.html. 13, 62

[89] Team HY-World, Chenjie Cao, Xuhui Zuo, Zhenwei Wang, Yisu Zhang, Junta Wu, Zhenyang Liu, Yuning Gong, Yang Liu, Bo Yuan, Chao Zhang, Coopers Li, Dongyuan Guo, Fan Yang, Haiyu Zhang, Hang Cao, Jianchen Zhu, Jiaxin Lin, Jie Xiao, Jihong Zhang, Junlin Yu, Lei Wang, Lifu Wang, Lilin Wang, Linus, Minghui Chen, Peng He, Penghao Zhao, Qi Chen, Rui Chen, Rui Shao, Sicong Liu, Wangchen Qin, Xiaochuan Niu, Xiang Yuan, Yi Sun, Yifei Tang, Yifu Sun, Yihang Lian, Yonghao Tan, Yuhong Liu, Yuyang Yin, Zhiyuan Min, Tengfei Wang, and Chunchao Guo. HY-World 2.0: A Multi-Modal World Model for Reconstructing, Generating, and Simulating 3D Worlds. *arXiv preprint arXiv:2604.14268*, 2026. doi: 10.48550/arXiv.2604.14268. URL https://arxiv.org/abs/2604.14268. 61

[90] Team Odyssey. Starchild-1: A real-time multimodal world model. https://starchild.odyssey.ml/starchild-1.pdf, 2026. Technical report. Accessed October 6, 2026. 63

[91] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991. doi: 10.1109/34.88573. URL https://web.stanford.edu/class/cs273/refs/umeyama.pdf. 26, 67

[92] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. FVD: A new metric for video generation. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2019. URL https://openreview.net/forum?id=rylgEULtdN. 54, 102

[93] 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. 6, 12, 20, 34

[94] Haoyuan Wang, Yabo Chen, Haibin Huang, Chi Zhang, and Xuelong Li. Directing the World: Fast Autoregressive Video Generation with Compositional Human-Camera Control. *arXiv preprint arXiv:2606.27964*, 2026a. doi: 10.48550/arXiv.2606.27964. URL https://arxiv.org/abs/2606.27964. 61

[95] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861. URL https://www.cns.nyu.edu/pub/lcv/wang03-reprint.pdf. 27, 36, 67

[96] Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, et al. Matrix-Game 3.0: Real-Time and Streaming Interactive World Model with Long-Horizon Memory. *arXiv preprint arXiv:2604.08995*, 2026b. URL https://arxiv.org/abs/2604.08995. 61

[97] Waymo. The Waymo world model: A new frontier for autonomous driving simulation. https://waymo.com/blog/2026/02/the-waymo-world-model-a-new-frontier-for-autonomous-driving-simulation/, 2026. 62

[98] World Labs. Announcing the World API. https://www.worldlabs.ai/blog/announcing-the-world-api, 2026. January 21, 2026. Accessed October 6, 2026. 61

[99] World Labs Team. Atlas: A World Model for Spatial Intelligence. https://www.worldlabs.ai/blog/atlas, 2026. September 1, 2026. Accessed October 6, 2026. 61

[100] Haoning Wu, Erli Zhang, Liang Liao, Chaofeng Chen, Jingwen Hou, Annan Wang, Wenxiu Sun, Qiong Yan, and Weisi Lin. Exploring video quality assessment on user generated contents from aesthetic and technical perspectives. In *ICCV*, 2023. 35, 41, 54, 102

[101] 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. doi: 10.52202/079017-2173. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/7dbb5bfab324e3b86af9bd0df15498dd-Abstract-Conference.html. 61

[102] Ruiqi Wu, Xuanhua He, Meng Cheng, Tianyu Yang, Yong Zhang, Zhuoliang Kang, Xunliang Cai, Xiaoming Wei, Chunle Guo, Chongyi Li, and Ming-Ming Cheng. Infinite-World: Scaling Interactive World Models to 1000-Frame Horizons via Pose-Free Hierarchical Memory. In *ICML*, 2026. URL https://proceedings.mlr.press/v306/wu26v.html. 63

[103] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *ICLR*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF. 18, 63

[104] Zhexiao Xiong, Yizhi Song, Hao Kang, Qing Yan, Liming Jiang, Jenson Yang, Zhoujie Fu, Stathi Fotiadis, Angtian Wang, Zichuan Liu, Bo Liu, Yiding Yang, Xin Lu, and Nathan Jacobs. ActWorld: From Explorable to Interactive World Model via Action-Aware Memory. In *NeurIPS*, 2026. URL https://interactwm.github.io/ActWorld/. To appear. 61, 63

[105] Dejia Xu, Weili Nie, Chao Liu, Sifei Liu, Jan Kautz, Zhangyang Wang, and Arash Vahdat. CamCo: Camera-controllable 3D-consistent image-to-video generation. *arXiv preprint arXiv:2406.02509*, 2024. 61

[106] Jiacong Xu, Hanwen Jiang, Zhixin Shu, Kalyan Sunkavalli, Vishal M. Patel, and Yiqun Mei. Wonder: Video World Model Done Better. *arXiv preprint arXiv:2607.26037*, 2026a. doi: 10.48550/arXiv.2607.26037. URL https://arxiv.org/abs/2607.26037. 63

[107] Youcan Xu, Jiaxin Shi, Zhen Wang, Wensong Song, Feifei Shao, Chen Liang, Jun Xiao, and Long Chen. RealCam: Real-Time Novel-View Video Generation with Interactive Camera Control. *arXiv preprint arXiv:2605.06051*, 2026b. doi: 10.48550/arXiv.2605.06051. URL https://arxiv.org/abs/2605.06051. 63

[108] Jiazhi Yang, Shenyuan Gao, Yihang Qiu, Li Chen, Tianyu Li, Bo Dai, Kashyap Chitta, Penghao Wu, Jia Zeng, Ping Luo, et al. Generalized predictive model for autonomous driving. In *CVPR*, 2024a. 35, 73

[109] Sherry Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *ICLR*, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/c4d66eae503694424123b93ac0fbaf17-Abstract-Conference.html. 61

[110] Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, et al. LongLive: Real-time Interactive Long Video Generation. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/91a1610c6ed9e02d33f826b46f472b92-Abstract-Conference.html. 63, 70

[111] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. 62

[112] Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. FlashInfer: Efficient and Customizable Attention Engine for LLM Inference Serving. In *MLSys*, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/dbf02b21d77409a2db30e56866a8ab3a-Abstract-Conference.html. 19, 63, 90

[113] Jung Yi, Minjae Kim, Paul Hyunbin Cho, Wooseok Jang, Sangdoo Yun, and Seungryong Kim. WorldKV: Efficient World Memory with World Retrieval and Compression. *arXiv preprint arXiv:2605.22718*, 2026. URL https://arxiv.org/abs/2605.22718. 63

[114] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. *NeurIPS*, 2024. 14, 15, 16, 62

[115] Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From Slow Bidirectional to Fast Autoregressive Video Diffusion Models. In *CVPR*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yin_From_Slow_Bidirectional_to_Fast_Autoregressive_Video_Diffusion_Models_CVPR_2025_paper.html. 62

[116] Jiwen Yu, Jianhong Bai, Yiran Qin, Quande Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Xihui Liu. Context as Memory: Scene-Consistent Interactive Long Video Generation with Memory Retrieval. In *SIGGRAPH Asia*, 2025. doi: 10.1145/3757377.3763833. URL https://doi.org/10.1145/3757377.3763833. 63

[117] Ted Zadouri, Markus Hoehnerbach, Jay Shah, Vijay Thakkar, and Tri Dao. FlashAttention-4: Algorithm and Kernel Pipelining Co-Design for Asymmetric Hardware Scaling. In *MLSys*, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/ae8b0b5838ba510daff1198474e7b984-Abstract-Conference.html. 51, 89

[118] Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame Context Packing and Drift Prevention in Next-Frame-Prediction Video Diffusion Models. In *NeurIPS*, 2025a. doi: 10.52202/085713-1024. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/2bde8fef08f7ebe42b584266cbcfc909-Abstract-Conference.html. 62

[119] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html. 27, 67

[120] Xiangcheng Zhang, Runhan Huang, and Yilun Du. World Action Planner: Generalizable Robot Decision-Making with Action-Conditioned World Models. *arXiv preprint arXiv:2607.27599*, 2026a. doi: 10.48550/arXiv.2607.27599. URL https://arxiv.org/abs/2607.27599. 62

[121] Yifu Zhang, Hao Yang, Yuqi Zhang, Yifei Hu, Fengda Zhu, Chuang Lin, Xiaofeng Mei, Yi Jiang, Zehuan Yuan, and Bingyue Peng. Waver: Wave your way to lifelike video generation. *arXiv preprint arXiv:2508.15761*, 2025b. 66

[122] Yisu Zhang, Chenjie Cao, Tengfei Wang, Xuhui Zuo, Junta Wu, Jianke Zhu, and Chunchao Guo. WorldStereo: Bridging Camera-Guided Video Generation and Scene Reconstruction via 3D Geometric Memories. In *CVPR*, 2026b. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zhang_WorldStereo_Bridging_Camera-Guided_Video_Generation_and_Scene_Reconstruction_via_3D_CVPR_2026_paper.html. 61

[123] Yujia Zhang, Xiaoyang Wu, Yunhan Yang, Xianzhe Fan, Han Li, Yuechen Zhang, Zehao Huang, Naiyan Wang, and Hengshuang Zhao. Utonia: Toward One Encoder for All Point Clouds. In *ICML*, 2026c. doi: 10.48550/arXiv.2603.03283. URL https://arxiv.org/abs/2603.03283. 54, 103

[124] Min Zhao, Hongzhou Zhu, Kaiwen Zheng, Zihan Zhou, Bokai Yan, Xinyuan Li, Xiao Yang, Chongxuan Li, and Jun Zhu. Causal Forcing++: Scalable Few-Step Autoregressive Diffusion Distillation for Real-Time Interactive Video Generation. *arXiv preprint arXiv:2605.15141*, 2026. URL https://arxiv.org/abs/2605.15141. 62

[125] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch FSDP: experiences on scaling fully sharded data parallel. *PVLDB*, 2023. 51, 67, 96

[126] Jia Zheng, Teli Ma, Yudong Fan, Zifan Wang, Shuo Yang, and Junwei Liang. MotionWAM: Towards Foundation World Action Models for Real-Time Humanoid Loco-Manipulation. *arXiv preprint arXiv:2606.09215*, 2026a. doi: 10.48550/arXiv.2606.09215. URL https://arxiv.org/abs/2606.09215. 62

[127] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *ICLR*, 2026b. 6

[128] Kaiwen Zheng, Guande He, Min Zhao, Jintao Zhang, Huayu Chen, Jianfei Chen, Chen-Hsuan Lin, Ming-Yu Liu, Jun Zhu, and Qianli Ma. Causal-rCM: A Unified Teacher Forcing and Self Forcing Open Recipe for Autoregressive Diffusion Distillation in Streaming Video Generation and Interactive World Models. *arXiv preprint arXiv:2606.25473*, 2026c. URL https://arxiv.org/abs/2606.25473. 4, 11, 15, 16, 17, 51, 60, 62, 63

[129] Kaiwen Zheng, Yuji Wang, Qianli Ma, Huayu Chen, Jintao Zhang, Yogesh Balaji, Jianfei Chen, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Large scale diffusion distillation via score-regularized continuous-time consistency. In *ICLR*, 2026d. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0534abc9e6db91683d82186ef0d68202-Abstract-Conference.html. 62

[130] Zirui Zheng, Jiaqian Yu, Xiongfeng Peng, Jun Shi, Mingyi Li, Chao Zhang, Weiming Li, Dong Wang, Huchuan Lu, and Xu Jia. Mem-World: Memory-Augmented Action-Conditioned World Models for Persistent Robot Manipulation. *arXiv preprint arXiv:2606.18960*, 2026e. doi: 10.48550/arXiv.2606.18960. URL https://arxiv.org/abs/2606.18960. 63

[131] Fengzhe Zhou, Jiannan Huang, Jialuo Li, Deva Ramanan, and Humphrey Shi. PAI-Bench: A comprehensive benchmark for physical AI. In *CVPR*, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zhou_PAI-Bench_A_Comprehensive_Benchmark_For_Physical_AI_CVPR_2026_paper.html. 25, 27, 36, 67

[132] Xiaoyu Zhou, Dingwei Xian, Zhenyu Wang, Yajiao Xiong, Yongtao Wang, and Ming-Hsuan Yang. ChronoWorld: Camera-Controlled Consistent 4D World Generation via Spatiotemporal Cues and Geometric Reflections. *arXiv preprint arXiv:2610.06687*, 2026b. doi: 10.48550/arXiv.2610.06687. URL https://arxiv.org/abs/2610.06687. 61

[133] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019. 21

[134] Haoyi Zhu, Haozhe Liu, Yuyang Zhao, Tian Ye, Junsong Chen, Jincheng Yu, Tong He, Song Han, and Enze Xie. SANA-WM: Efficient Minute-Scale World Modeling with Hybrid Linear Diffusion Transformer. *arXiv preprint arXiv:2605.15178*, 2026a. URL https://arxiv.org/abs/2605.15178. 25, 26, 27, 61, 68

[135] Ziyu Zhu, Zhaoxu Sun, Fan Jiang, Mengchao Wang, Junfeng Ni, Ting-Bing Xu, Zhicheng Liu, and Mu Xu. LongForcing: Scaling Temporal Manifold Matching for Long-Horizon World Rollout. *arXiv preprint arXiv:2610.18289*, 2026b. URL https://arxiv.org/abs/2610.18289. 62

[136] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic LiDAR point clouds. In *European Conference on Computer Vision (ECCV)*, volume 13683 of *Lecture Notes in Computer Science*, pages 17–35. Springer, 2022. doi: 10.1007/978-3-031-20050-2_2. URL https://doi.org/10.1007/978-3-031-20050-2_2. 102