A Mesh to Remember: Persistent Appearance for Interactive Video Worlds

Joonghyuk Shin[1,2*] Alex Trevithick[1] Katarína Tóthová[1] Liyuan Zhu[1,3*] Shengyu Huang[1]
Jaesik Park[2] Shalini De Mello[1] Laura Leal-Taixé[1] Riccardo de Lutio[1†] Haithem Turki[1†]

[1]NVIDIA [2]Seoul National University [3]Stanford University

[*]Work done during an internship at NVIDIA [†]Equal supervision

https://research.nvidia.com/labs/sil/projects/m2r/

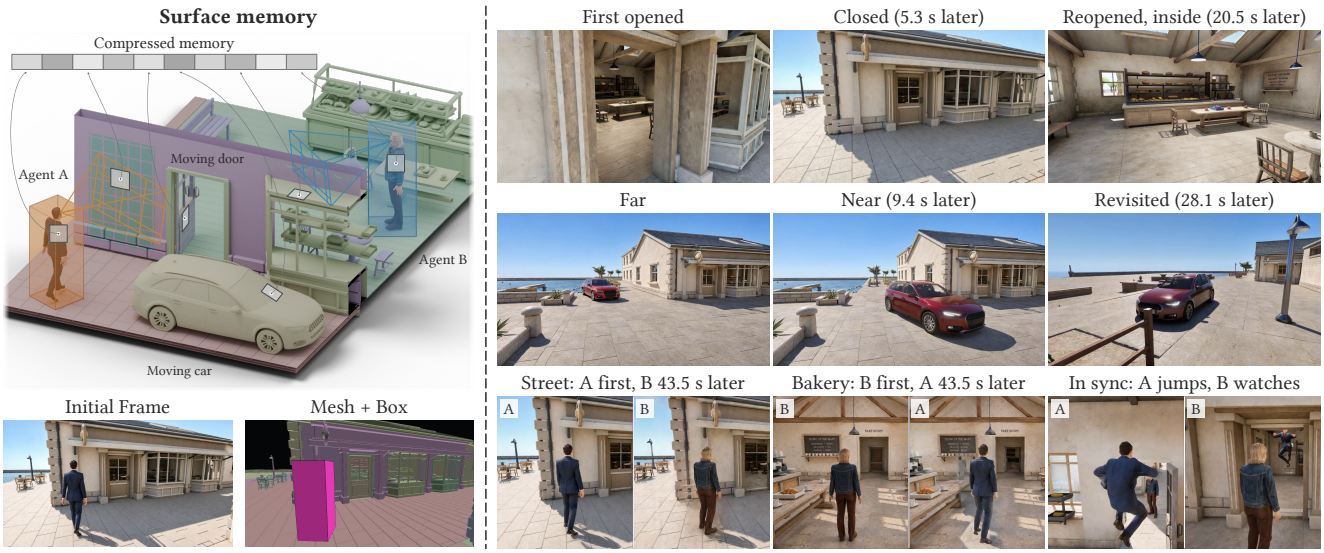

Figure 1. Given an initial frame and a coarse, untextured mesh from a coding agent or a 3D scan, M2R renders photorealistic video for every agent in real time. It stores the features it generates on the mesh surfaces they depict, in a memory that all agents share (**left**). Appearance thus persists behind a reopened door, on a moving car, and across agents, long after leaving the model's context (**right**).

Abstract

Video world models synthesize photorealistic observations in response to user actions but struggle to remain consistent over long interactions. Coding agents and text-to-3D models instead generate explicit 3D worlds whose geometry and state persist by construction, but whose appearance falls far short of photorealism. Recent methods combine the strengths of both by rendering such worlds into coarse proxies that guide a video model. However, since these proxies only specify where things are, the renderer must invent what they look like as the world is explored, and anything it forgets is reinvented when a surface reappears. Our key insight is that a mesh identifies which surface each pixel shows, across frames and views, even as objects move, allowing it to serve as an address for what the renderer generates. We propose M2R (Mesh to Remember), a few-step causal video renderer that stores its own features on the surfaces they depict, in each object's frame, and retrieves them by rasterizing the mesh whenever those surfaces come back into view. Appearance thus travels with objects wherever they move, persists across revisits, and is shared by every agent that observes it, while memory grows with the surfaces seen instead of the number of frames. Trained only on synthetic data, M2R transfers zero-shot to real indoor scans and to worlds authored by coding agents. It outperforms state-of-the-art methods in consistency across static, dynamic, and multi-agent benchmarks, more than doubling object permanence and nearly doubling cross-agent consistency, while rendering 720p video in real time (40 FPS).

1. Introduction

Interactive virtual worlds are essential for applications in gaming, embodied AI, and closed-loop simulation. These use cases require both photorealistic rendering and persistent world state: a door opened by one agent should remain open when another walks by, and the room behind it should look the same each time it is opened (Fig. 1). In recent years, two paradigms have emerged that promise to build such worlds automatically: video world models and 3D world generators.

Video world models synthesize observations in response to user actions, from game worlds [4, 27, 80] to photorealistic scenes [1, 27, 63], and increasingly for multiple players [20, 30, 34, 47, 70, 87]. Despite this progress, these models must infer both how the world evolves and how it looks from their own past observations, and errors in either can accumulate over long interactions. Memory mechanisms that retrieve past frames or their cached features, or maintain 3D caches, extend the history a model can draw upon and improve consistency when regions are revisited [71, 76, 89, 93, 102, 106]. However, they tie memory to where content was observed, which no longer holds once the world changes. When an object is carried across a room, its remembered appearance stays where the object used to be, and the model must reinvent it.

3D world generators instead create the world itself, from explorable scenes that text-to-3D models produce given a single prompt [37, 78] to entire dynamic worlds that coding agents write as programs, with articulated furniture [83] and rules for how objects move [13]. As these worlds are explicit, their geometry and state persist by construction, and every agent observes the same world. However, they fall short of photorealism, as even the most sophisticated real-time game renderings remain distinguishable from real imagery [69].

Instead of treating these paradigms as standalone alternatives, recent methods combine their complementary strengths: the explicit world maintains persistent geometry and state, while a video model renders coarse proxies of it (geometry or semantic layouts) into realistic observations [13, 15, 21, 36, 44, 55]. Since these proxies only specify what is where, the renderer must invent the world's appearance as it is explored, and keep it consistent as agents revisit regions, move objects, and encounter surfaces that others have already seen. A natural place to keep this appearance is the explicit world itself. Existing methods paint generated colors back onto its geometry [21, 28, 57, 68], but colors are a lossy copy of what the renderer generated and must be re-rendered and re-encoded at every step, which is slow and accumulates errors (Tab. 4).

We instead treat the mesh as an address. Our key insight is that a mesh identifies the surface behind every pixel in every view, even after that surface has moved. We therefore store the renderer's own features on the surfaces they depict as they are generated, and attend to them whenever those surfaces reappear. This memory has three benefits. First, as memory is indexed by surface instead of time, storage follows scene coverage and observations remain addressable long after leaving the model's context. Second, when an object moves or articulates, its remembered appearance travels with it, whereas memory tied to camera poses or world positions stays behind. Third, all agents read from and write to the same surfaces, allowing each to render what only others have seen.

We realize this insight in M2R (Mesh to Remember), a real-time video world model that renders worlds of coarse, untextured meshes into photorealistic video from a single initial frame. As users act and the world updates, M2R conditions at every step on renderings of the current meshes and agent boxes from each agent's viewpoint, reusing stored appearance on observed surfaces and synthesizing newly revealed regions and view-dependent effects. Agents are remembered the same way: M2R generates each agent's body and motion from its bounding box, and the box carries that appearance with it. We fine-tune a pretrained video diffusion model only on synthetic multi-agent episodes rendered in Unreal Engine with Isekai3D [39] and adapt it into a few-step causal renderer with rollout-based training [35]. As this renderer remembers nothing outside its short context, we train it to write its own features to the surface memory during rollouts and to read them back on palindromic paths that revisit earlier views, so that it learns to use memory that carries its own errors. In summary, our contributions are:

1. We design a causal video renderer that conditions on renderings of scene geometry and agent bounding boxes, and reads and updates surface memory indexed by the mesh in each object's frame, so that remembered appearance follows objects as they move.
2. We bound this memory by the observed surface area, compress it 18-fold with transform coding inspired by language-model KV caches [74], and pair it with a fast video autoencoder, allowing M2R to render at 40 FPS and keep 8,000-frame rollouts within 45 GiB of GPU memory.
3. Although trained only on synthetic data, M2R transfers zero-shot to real indoor scans and to articulated worlds authored by coding agents, and outperforms state-of-the-art methods on revisit, interaction, and multi-agent benchmarks.

2. Related Work

Interactive video world models. Interactive video world models [1, 4, 27, 80] are progressing from a single player to several [20, 87], who must all observe the same evolving world. Most multi-agent models keep this shared world inside the network, generating all players' views jointly [30, 34, 47, 58, 70]. Joint generation grows costly as players are added [34, 47], and the shared world can drift

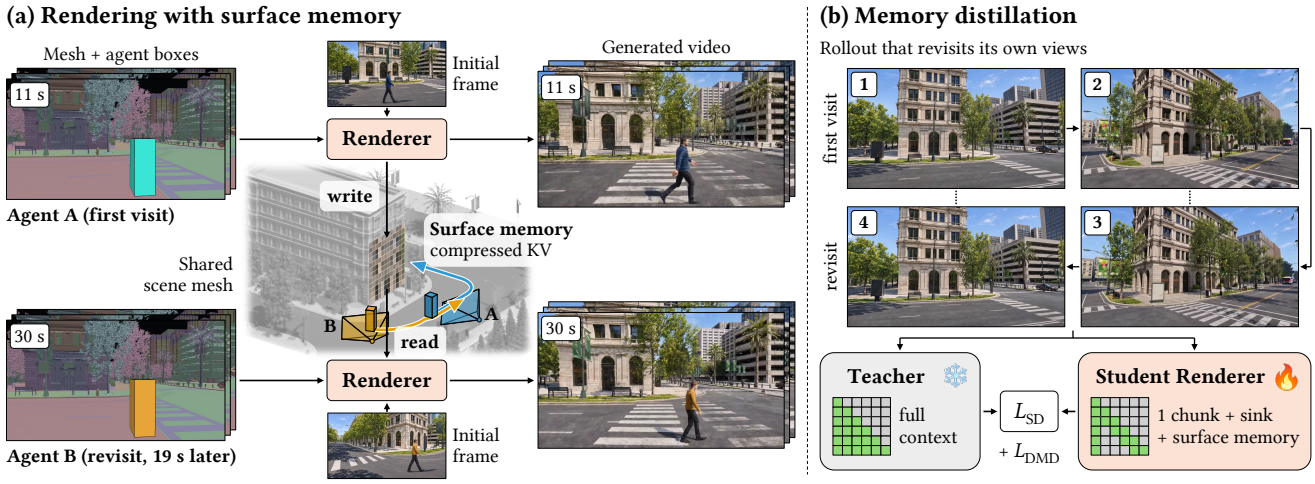

Figure 2. **Method overview.** M2R renders each agent's view from the initial frame and renderings of a shared scene mesh. As agent A explores, it stores generated keys and values in a surface memory, indexed by the mesh surfaces they depict. When agent B encounters the same surfaces, it retrieves and renders them as agent A saw them (**a**). We train our renderer on rollouts that revisit earlier views. With a one-chunk window, it must render each revisit from the memory it wrote, supervised by a frozen teacher that sees the full past (**b**).

once players leave each other's view [70]. Other methods render each view from an explicit state, such as an editable level map [65], an authoritative game state [7], an entity table [113], or a shared point cloud [31]. While this keeps geometry and state consistent across agents, it does not retain the appearance that each view generates. To share this appearance, recent methods exchange generated frames between agents [96, 100, 116]. M2R instead stores it directly on the mesh surfaces that every agent observes.

Generative rendering of explicit geometry. Generative models increasingly render explicit geometry into realistic footage, from G-buffers, depth, and segmentation maps [43, 62] to renderings of animated meshes, low-poly games, and semantic block worlds [8, 11, 24, 26], some in real time [61, 107]. Executable worlds apply them to scenes written as programs, often by coding agents, with their geometry, state, and rules [13, 15, 36, 44, 55, 108]. However, as geometry alone does not determine materials, colors, or identities [15], the renderer must remember the appearance it invents. Offline methods store this appearance in the 3D representation itself, in mesh textures [28, 68], in the UV space of animated meshes [6], or by distilling refined views into captured scenes [17, 88]. They require the animation or target views in advance, which an interactive world cannot provide. Closest to our renderer, the concurrent AgentGarten [15] adapts the same backbone into a few-step causal renderer of code-defined worlds. While it keeps appearance only in a reference frame and its recent context, and can lose it after long occlusions, M2R stores it on the mesh itself and thus remains consistent over long rollouts (Sec. 5.3).

Memory for long video generation. Most memories retrieve past frames by camera pose or view overlap [76, 85, 93, 105] or return the cached keys and values of past chunks [14, 29, 102, 111]. Unless bounded [14, 18], these memories grow with the rollout, reaching an estimated 300 GiB of CPU memory per thousand frames for the concurrent MosaiChunk's image-to-video backbone [111]. They also leave the network to infer how the retrieved content corresponds to the current view. Geometric memories make this correspondence explicit by storing observations in world-fixed point clouds, surfels, or per-frame geometry [32, 42, 57, 67, 71, 89], lifting features or latent patches into world space [66, 84, 106, 114], or reprojecting tokens with engine depth [56]. However, this correspondence only holds for static content, as moving objects are filtered out [66, 84, 89] or left to the generator [106], and even TourPhysics [110], which routes the generator's own attention values with simulator geometry, excludes them from long-term memory. Memories that do handle motion replay it from an input video [67, 115], predict it while objects are out of sight [12, 19, 72], or remember objects by identity [18, 82, 94], but none track which of an object's surfaces is visible. Closest to our memory, the concurrent CoDeR [21] registers generated colors on whitebox surfaces, where later views and other agents observe them. As it renders 124 frames at a time, however, the effect of an action becomes visible to any agent only after a full clip is rendered. It also learns to read colors registered from its training targets instead of its own outputs, which lets its errors accumulate over long rollouts. M2R instead keeps the renderer's own keys and values, which it writes after every chunk and learns to read back from its own rollouts. Stored in mesh surface cells in each object's frame, they follow objects as they move, grow only with the surfaces seen, and can be read by any agent in real time.

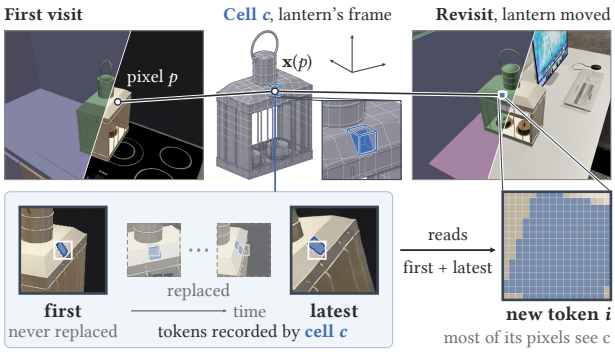

Figure 3. **Surface memory.** We assign each pixel p to the cell c that contains its surface point $\mathbf{x}(p)$ in the object's coordinate frame (**top**). After each chunk, we record the token that best covers each visible cell, keeping only the first and latest (**bottom left**). When we revisit the lantern after it has moved, each new token i reads the past tokens that most of its pixels point to (**bottom right**).

3. Preliminaries

Video diffusion models. We build upon video diffusion models, which typically operate in the latent space of a video autoencoder. Flow matching [46] learns to transport Gaussian noise into these latents. Given clean latents $\mathbf{z}$, noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, and flow time $\tau \in [0, 1]$, we construct $\mathbf{z}_\tau = (1-\tau)\boldsymbol{\epsilon} + \tau\mathbf{z}$ and train a velocity field v_θ by minimizing $\mathbb{E}_{\mathbf{z},\boldsymbol{\epsilon},\tau}\|v_\theta(\mathbf{z}_\tau, \tau; \mathbf{c}) - (\mathbf{z} - \boldsymbol{\epsilon})\|_2^2$, where $\mathbf{c}$ denotes conditioning signals such as actions or renderings of scene geometry. At inference, integrating the field from $\tau = 0$ to $\tau = 1$ produces clean latents.

Autoregressive video generation. As bidirectional video models denoise all frames of a clip jointly, their memory cost grows with the number of frames. To solve this, long-horizon interactive world models instead generate video autoregressively, one latent chunk at a time. Each chunk attends to the chunks before it [104], whose keys and values are kept in a KV cache. To keep this cache bounded, only a sliding window of recent chunks is typically retained [35, 73], at the cost of reinventing any content that reappears after leaving the window. As autoregressive models condition on their own outputs at inference, Self Forcing [35] trains them on their own rollouts, supervised by distribution matching distillation (DMD) [103] from a bidirectional teacher.

4. Method

Given a coarse, untextured mesh, a caption of the scene, and an initial first- or third-person frame for each agent, our goal is to render consistent, photorealistic views in real time as agents explore and change the world. We do so via a few-step causal video renderer that is conditioned on renderings of the mesh and stores generated features in a surface memory shared by all agents (Fig. 2). As the mesh identifies the surface behind every pixel, the renderer can retrieve these features whenever the surface they depict reappears, even after it has moved.

4.1. Geometry Conditioning

To specify what is visible in each chunk, we rasterize the current mesh and agent boxes from each agent's camera into a proxy video, with flat shading and one arbitrary color per asset to separate objects. We encode it into the backbone's latent space and add it to the embedded noisy latents through a zero-initialized patch embedding, which keeps the sequence length unchanged.

4.2. Surface Memory

Most autoregressive video models remember between 24 and 72 frames in a key-value cache [35, 73, 99]. Instead, we pair a much smaller temporal cache (8 frames) with a surface memory shared across agents, which indexes keys and values by the mesh surfaces they depict. Once observed surfaces are evicted from the temporal cache, they remain retrievable by any agent that later sees them, and the memory grows with the surfaces observed instead of with time (Fig. 7b).

Cells. To index the memory, we divide the surfaces of the mesh and agent boxes into small cells. We define cells in the coordinates of the object they lie on, which lets a cell follow its surface as the object moves or articulates, with no update to the memory (Fig. 4). Rasterizing this geometry into a view gives the object $o(p)$ and the surface point $\mathbf{x}(p)$ behind each pixel p, and we assign the pixel to the cell that holds this point (Fig. 3, top),

$$c(p) = \Big(o(p),\ \big\lfloor \mathbf{T}_{o(p)}(t)\,\mathbf{x}(p)/s \big\rfloor\Big), \tag{1}$$

where $\mathbf{T}_o(t)$ maps world coordinates into the coordinates of object o at time t and s is the cell size. As cells do not depend on any camera, all agents address a surface through the same cell. In practice, we also split cells by the direction the surface faces and keep them at four sizes, so that surfaces seen from afar still fill the coarser cells (Sec. D).

Writing. As the renderer operates on tokens that each span a patch of pixels across several frames, a single token often covers multiple cells. We therefore store each token's keys and values once and let cells point to it. After generating chunk n, we associate each visible cell c with the token that covers most of its pixels, $j = \arg\max_{j'} |\{p \in \Omega_{j'} : c(p) = c\}|$, where $\Omega_{j'}$ denotes the pixels of token j', and record the observation (n, j) (Fig. 3, bottom left). We keep two observations per cell: the first, which we never replace [95] to prevent errors from accumulating across revisits, and the latest, which reflects the surface's current state.

Storage. Storing the keys and values of every token is expensive [91, 102], amounting to about 54 GiB per minute

Table 1. **Static environments.** We report Orbia scores [54] (0–100, higher is better) on ScanNet++ and Replica, with the best, second, and third results highlighted. Throughout the tables, † and § mark methods that generate at 480p and 540p, while all others generate at 720p. M2R outperforms all baselines on every metric except visual quality, and is the only method to render 720p video in real time.

Method	ScanNet++ [101]						Replica [75]						Efficiency				
	Control Align.	Visual Quality	Input Pres.	Content Persist.	3D Consist.	Temp. Stab.	Control Align.	Visual Quality	Input Pres.	Content Persist.	3D Consist.	Temp. Stab.	# Params.	Lat. (s)↓	FPS↑	MP/s↑	Peak (GiB)↓
SolarWM H3 [33]	83.01	71.72	71.46	80.34	74.96	87.22	84.26	73.65	74.23	84.09	76.46	88.99	33B	9.8	1.73	1.79	113.6
Matrix-Game 3.5 [66]	62.72	69.52	71.18	54.58	54.15	75.89	68.67	72.21	69.77	66.23	56.25	79.07	5B	4.5	2.67	2.41	63.6
GEN3C [67]	85.48	67.21	80.68	85.46	72.57	86.93	95.82	69.64	88.88	92.06	75.42	90.83	7B	147.5	0.81	0.73	50.9
Lyra 2.0† [71]	92.21	71.71	82.26	81.92	75.18	89.88	93.33	72.05	83.57	86.26	76.21	90.43	14B	170.2	0.47	0.19	72.6
Lyra 2.0 (DMD)† [71]	92.11	71.60	82.42	85.36	75.23	90.11	94.18	72.85	82.64	86.53	76.73	91.30	14B	23.7	3.38	1.35	73.0
WorldDirector† [82]	78.62	67.60	79.48	73.75	70.14	83.43	85.10	68.84	83.86	84.84	75.90	86.60	14B	289.6	0.28	0.11	67.9
Code World Model [13]	83.64	71.93	82.40	80.14	74.41	89.03	85.89	71.14	83.82	76.81	77.23	88.68	33B	535.7	0.17	0.18	75.7
Code Video Model§ [44]	67.60	73.54	74.84	70.24	74.98	85.26	71.99	71.21	78.62	69.89	75.22	85.00	33B	166.0	0.75	0.39	146.3
AgentGarten† [15]	88.92	73.66	87.15	89.49	75.02	88.62	93.20	73.63	88.44	91.72	78.89	91.10	8B	0.5	34.1	13.6	39.2
M2R	95.63	72.52	88.54	89.88	76.60	91.37	97.80	72.91	92.27	92.16	79.69	92.34	8B	0.19	41.1	37.0	33.6

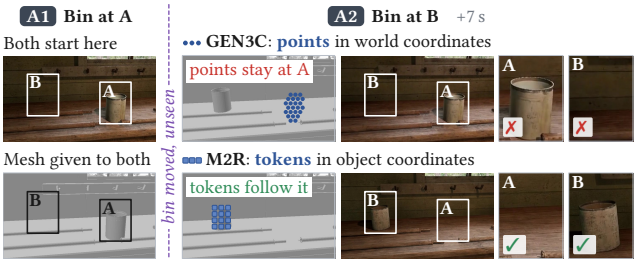

Figure 4. **Moving objects.** A bin is seen at A (A1) and then moved to B while the camera looks away. GEN3C [67] indexes its 3D cache by world position, which leaves the bin's appearance at A and nothing at B (**top**), and thus still renders the bin at A and none at B when the camera looks back (A2). As we define cells in each object's frame, the bin keeps its cells (**bottom**), and M2R renders it at B and no longer at A.

of 720p video at 30 FPS. To limit this cost, we discard tokens as they leave the temporal cache unless a cell still points to them. We further compress the remaining tokens following key-value compression for language models [9, 74]. We project each token's keys and values across all layers onto a principal component basis that we fit once on the renderer's own rollouts, and store the result at 8-bit precision. This shrinks each token 18-fold without degrading consistency (Sec. D and Tab. A1). The temporal cache, which also keeps the initial chunk as an attention sink [50, 92, 99], stays uncompressed.

Reading. For each token i of a new chunk, we look up the first and latest observations of the cells that it covers. Since a token can cover multiple cells, we select the past token that most of its pixels $p \in \Omega_i$ point to, choosing first its chunk n_i and then the token j_i within that chunk,

$$n_i = \arg\max_n \left|\{\, p \in \Omega_i : (n, \cdot) \in \mathcal{O}(p) \,\}\right|,$$
$$j_i = \arg\max_j \left|\{\, p \in \Omega_i : (n_i, j) \in \mathcal{O}(p) \,\}\right|, \quad (2)$$

where $\mathcal{O}(p)$ is the first observation of the cell at pixel p. We repeat this selection with the latest observations and retrieve a set $\mathcal{R}_i$ of at most two past tokens (Fig. 3, bottom right).

Memory conditioning. Each token attends to the tokens retrieved for its chunk in addition to the temporal cache. We assign each token in $\mathcal{R}_i$ the spatial position of token i, as in the re-positioned frame and patch memories of recent world models [66, 76, 97, 106] (Sec. D and Fig. A3). As each token retrieves at most two past tokens, the cost of rendering stays independent of the length of the interaction.

Multiple agents. With several agents, each agent keeps its own temporal cache, while all agents read from and write to the same surface memory. Each agent can thus render surfaces that only others have seen (Tab. 3), even though we train with a single view at a time (Sec. D).

4.3. Optimization

Causal renderer. We start from a bidirectional image-to-video model and fine-tune it with flow matching on videos paired with mesh renderings and keyboard controls (Secs. C and E), so that it learns to follow the geometry. For real-time generation, we make it causal and few-step following Causal Forcing++ [112], and then train it on its own rollouts with Self Forcing (Sec. 3) to limit how errors compound over long videos. The resulting renderer generates arbitrarily long videos with a few sampling steps per chunk, but remembers nothing outside its temporal cache.

Memory distillation. Although the causal renderer already benefits from the memory without any training (Sec. B and Tab. A1), it has little incentive to rely on it. We therefore apply Self Forcing a second time, except that the renderer now reads from and writes to its memory on rollouts that revisit earlier views (Fig. 2b). We construct these rollouts as palindromes, in which the camera goes back and forth up to four times, such that each return can only be rendered from



Figure 5. **Static environments.** We show a ScanNet++ rollout that returns to the input view (**A**) and revisits a generated view (**B**). Lyra 2.0 and SolarWM H3 fail to preserve the input view (**bottom rows**), and AgentGarten alters the monitor by the window (**right insets**). Although trained only on synthetic data, M2R is the only method to preserve both views.

memory (as the temporal cache holds a single chunk). To supervise each return, we use a frozen copy of the causal renderer that attends to the entire rollout (and thus still sees the first visit) as a teacher [48, 111], and minimize the difference between their clean-latent predictions $\hat{\mathbf{z}}^T$ and $\hat{\mathbf{z}}^S$ at every chunk:

$$\mathcal{L}_{\text{SD}} = \mathbb{E}_{n,\tau}\left[\left\|\hat{\mathbf{z}}^S_{n,\tau} - \text{sg}\big(\hat{\mathbf{z}}^T_{n,\tau}\big)\right\|_2^2\right], \tag{3}$$

where sg stops gradients. Our final loss is $\mathcal{L} = \mathcal{L}_{\text{DMD}} + \lambda\,\mathcal{L}_{\text{SD}}$, with $\lambda = 0.1$ (Sec. E and Fig. A5).

5. Experiments

We evaluate M2R in the three situations in which an interactive world model must remember appearance: a camera revisits a region, an object moves, and several agents see the same surfaces.

5.1. Implementation

We use Cosmos3-Nano [60] as the base model and train on 64 GPUs with paired mesh and RGB videos from Isekai3D [39]. At inference time, we generate eight-frame chunks (two latent frames) with two-step denoising, and measure all timings with a single GB200. To speed up encoding and decoding, we replace the base model's VAE with FastVAE, a transformer autoencoder that we train from scratch for the same latent space.

5.2. Static Environments

Datasets. We evaluate on 46 scenes of ScanNet++ [101], real indoor captures with laser-scanned meshes, and on the 18 scenes of Replica [75], whose dense meshes provide ground truth from any viewpoint. We follow the revisit protocol of Orbia [54], in which each 22-second trajectory starts at an input view, discovers a region absent from it, explores elsewhere, and then revisits both. We use Orbia's trajectories on ScanNet++ and generate four per scene on Replica (Sec. G.1).

Table 2. **Dynamic environments.** We report control alignment (Ctrl.), visual quality (Qual.), object permanence (Perm.), revisit (Rev.), and moved objects (0–100, higher is better). M2R more than doubles the best baseline's object permanence, and outperforms all baselines on every metric except visual quality.

	HSSD [40]					MutableWorlds				
Method	Ctrl.	Qual.	Perm.	Rev.	Moved	Ctrl.	Qual.	Perm.	Rev.	Moved
SolarWM H3	50.6	61.2	14.0	33.1	31.7	70.3	72.0	19.2	41.6	25.6
Matrix-Game 3.5	35.3	62.0	24.2	53.1	28.0	51.3	68.8	23.3	60.1	25.7
GEN3C	42.4	54.8	26.8	37.0	38.2	44.1	57.9	25.6	39.1	28.4
Lyra 2.0†	40.5	62.5	31.9	43.9	41.7	56.1	70.9	24.0	40.6	31.5
Lyra 2.0 (DMD)†	45.5	57.4	27.1	55.8	38.4	59.7	70.6	28.3	58.2	26.7
WorldDirector†	37.2	53.3	24.7	46.3	49.3	37.1	57.7	38.1	50.6	33.9
Code World Model	48.0	64.9	19.6	46.4	58.1	65.8	70.6	25.1	52.7	42.4
Code Video Model§	34.3	63.1	16.9	37.4	41.0	29.2	74.8	15.0	49.2	35.7
AgentGarten†	72.3	72.0	34.7	59.3	52.6	84.3	78.1	43.4	73.7	38.1
M2R	78.2	69.7	91.5	85.8	86.0	86.8	76.2	94.1	91.5	83.1

Baselines. We compare M2R to three groups of baselines (Tab. 1), namely a video world model that takes no geometry [33], methods that store what they generate in a 3D cache [66, 67, 71], and renderers that condition on proxy videos of the geometry [13, 15, 44, 82]. For fairness, we provide the ground-truth mesh to all methods that take geometry.

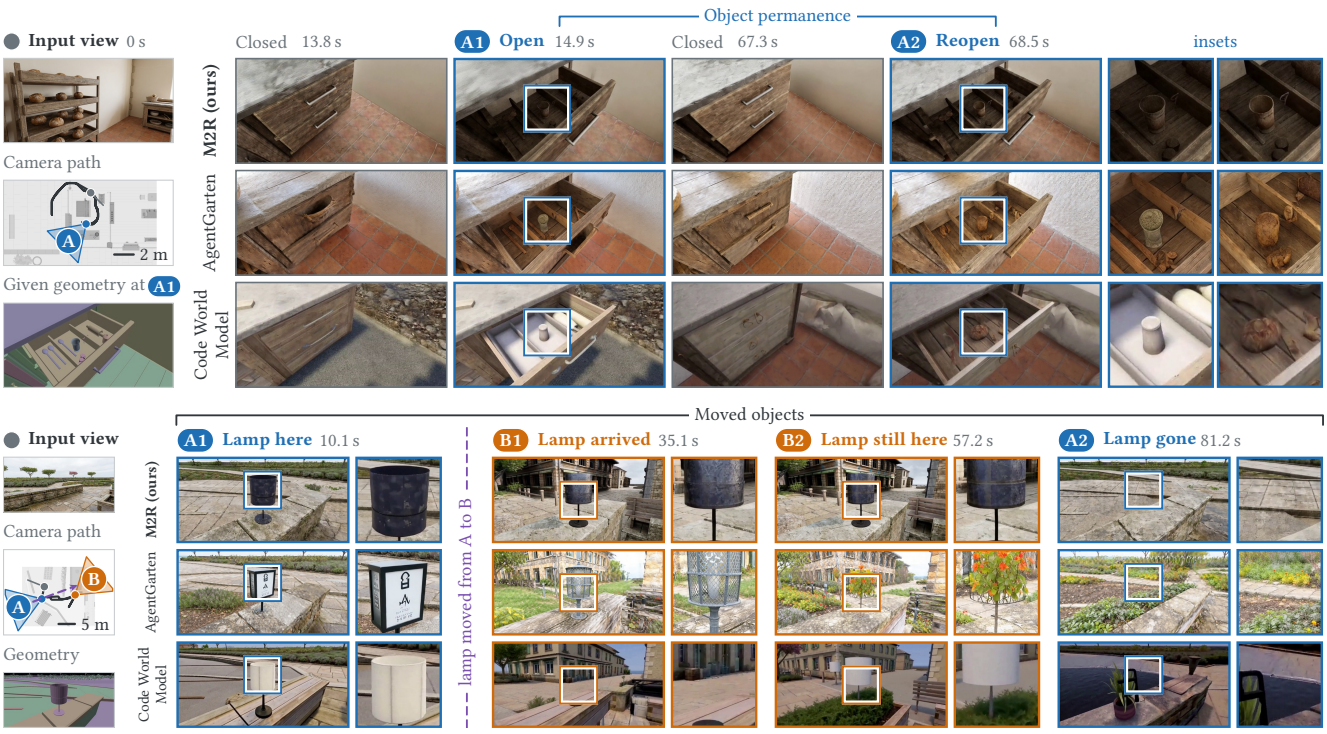

Figure 6. **Dynamic environments.** We show two rollouts in MutableWorlds in which a drawer is reopened (**top**) and a lamp is moved while out of view (**bottom**). AgentGarten and Code World Model replace the cup in the drawer with a loaf (**top insets**) and fail to preserve the lamp across visits. M2R is the only method to preserve appearance when drawers reopen and when objects move.

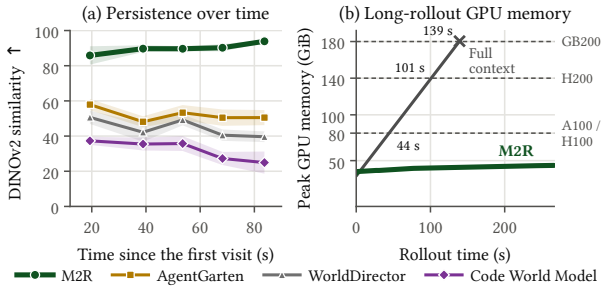

Figure 7. **Persistence and memory over time.** We plot DINOv2 similarity between the first and later openings of drawers and doors on HSSD (**a**) and peak GPU memory over an 8,001-frame rollout (**b**). Unlike the baselines, our memory does not fade with time. M2R stays below 45 GiB, whereas attending to every past chunk runs out of memory after 139 s.

Metrics. We report the six scores of Orbia [54] using its released evaluator: control alignment, visual quality, 3D consistency, temporal stability, and two memory scores. *Input preservation* compares later views of the surfaces visible in the input with ground truth, and *content persistence* compares each revisit of the discovered region with its first generated view. We also measure latency (from an input to the first frame it affects), frame rate, throughput in megapixels per second (MP/s), and peak GPU memory.

Table 3. **Multi-agent environments.** We render two or four agents that share a world in MutableWorlds, each from its own third-person camera. M2R is the only method to share appearance between agents, and outperforms all baselines by a wide margin.

Method	Two agents		Four agents	
	Cross-Agent Consist.	Agent Persist.	Cross-Agent Consist.	Agent Persist.
SolarWM H3 [33]	27.10	1.52	25.38	0.36
Matrix-Game 3.5 [66]	27.65	0.00	31.81	0.11
GEN3C [67]	26.82	4.68	24.80	1.80
Lyra 2.0† [71]	21.70	15.19	22.20	13.96
Lyra 2.0 (DMD)† [71]	26.05	4.95	23.33	2.39
WorldDirector† [82]	26.07	52.76	27.29	36.98
Code World Model [13]	36.51	43.08	37.45	31.41
Code Video Model§ [44]	32.05	12.62	25.38	22.94
AgentGarten† [15]	42.60	22.43	42.16	29.74
M2R	79.99	87.25	77.12	73.69

Results. We present quantitative results in Tab. 1 and qualitative examples in Fig. 5. Lyra 2.0 and SolarWM H3 lose the input view, and AgentGarten alters revisited objects. Although trained only on synthetic data, M2R outperforms all baselines on every metric except visual quality, where it performs competitively with the concurrent AgentGarten [15] and Code Video Model [44]. It is also the only method to render 720p video in real time, with 2.7× the throughput of

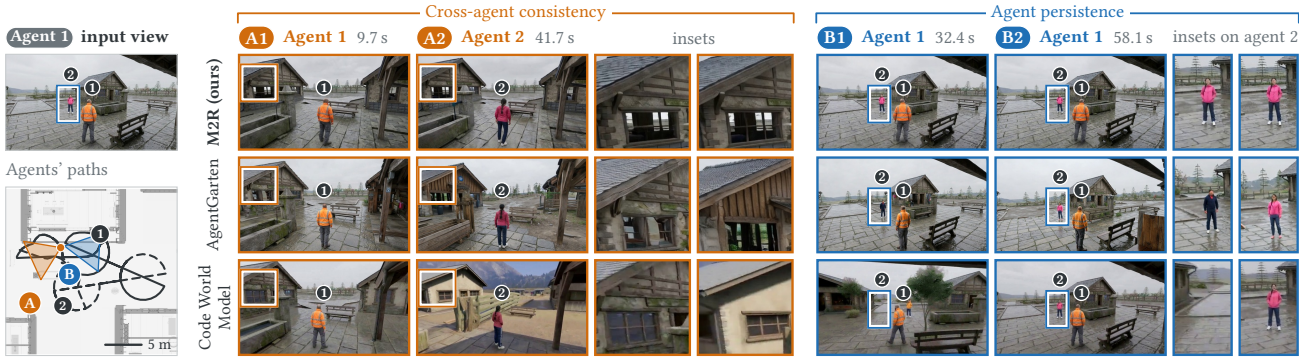

Figure 8. **Multiple agents.** We show a rollout in MutableWorlds in which two agents look from the same viewpoint (**A**) and meet twice (**B**). AgentGarten and Code World Model show agent 2 a different building (**A2**) and fail to preserve agent 2 across meetings (**B**). M2R is the only method to share appearance between agents.

Table 4. **Diagnostics.** We evaluate content persistence on ScanNet++, revisit on HSSD, and peak GPU memory over a 357-frame rollout. All trained variants, including M2R, use 16 GPUs for 300 iterations. Storing the renderer's key-value (K/V) features instead of colors is crucial to rendering revisits consistently, and compression shrinks the additional K/V store from 3.70 to 0.21 GiB (18×).

Memory	Compr.	Revisiting Rollouts	Self-Distill.	Fast VAE	Content Persist.	Revisit	Peak (GiB)↓	FPS↑
None	–	–	–	✓	84.45	74.33	31.7	51.1
RGB	–	✓	✓	✓	83.59	79.56	32.9	19.8
K/V	✗	✓	✓	✓	89.74	85.73	38.2	42.9
K/V	✓	✗	✓	✓	89.09	84.47	33.6	41.1
K/V	✓	✓	✗	✓	89.53	83.73	33.6	41.1
K/V	✓	✓	✓	✗	89.76	86.52	42.7	19.2
M2R	✓	✓	✓	✓	89.83	86.78	33.6	41.1

AgentGarten and over 15× that of every other baseline.

5.3. Dynamic and Multi-Agent Environments

Datasets and baselines. We evaluate on 45 trajectories in five articulated HSSD homes [10, 40] and on 52 in MutableWorlds, eight worlds that we build with a coding agent. The camera returns to rooms it has seen, reopens drawers and doors with objects inside, and revisits objects that have moved. We compare to the same methods as in Sec. 5.2.

Metrics. We report control alignment and visual quality as in Sec. 5.2. We measure memory by comparing a method's frames at each revisit with its own frames from the same camera poses on the first visit. *Revisit* compares the static parts of a view, and *object permanence* compares the contents of each reopened drawer or door. *Moved objects* checks that an object appears only at its current location, and that its appearance has not changed. We compute these scores from DINOv2 features [64] and ratings by Gemini 3.7 Flash [25], as detailed in Sec. G.2.

Results. We present quantitative results in Tab. 2 and qualitative examples in Fig. 6. M2R outperforms all baselines on every metric except visual quality, where it is second only to the concurrent AgentGarten. However, AgentGarten keeps appearance only in a reference frame and its recent context, and often shows new contents when a drawer or door is reopened (Fig. 6). We more than double its object permanence (91.5 vs. 34.7 on HSSD and 94.1 vs. 43.4 in MutableWorlds). Our memory does not fade with the time between visits (Fig. 7a). GEN3C and Lyra 2.0 index their 3D caches by world position, which leaves a moved object's appearance where it was first seen (Fig. 4). M2R is the only method to preserve appearance when drawers reopen and when objects move.

Multiple agents. In MutableWorlds, we also render 30 two-agent and 23 four-agent episodes, in which each agent later sees what another agent saw first. *Cross-agent consistency* averages revisit, object permanence, and moved objects when the first visit comes from another agent's video, and *agent persistence* compares another agent's appearance across two separate meetings. The baselines generate each agent's video separately, and thus give a new appearance to what another agent has already seen. We nearly double the best cross-agent consistency among them (80.0 vs. 42.6 with two agents and 77.1 vs. 42.2 with four) and raise the best agent persistence from 52.8 to 87.3 with two agents and from 37.0 to 73.7 with four (Tab. 3 and Fig. 8). M2R is the only method to share appearance between agents.

5.4. Diagnostics

We ablate what the memory stores, its compression, revisiting rollouts, the self-distillation loss, and FastVAE (Tab. 4). Storing the renderer's keys and values instead of colors is crucial, as colors on the same surface cells are less consistent (79.6 vs. 86.8 revisit) and must be re-rendered and re-encoded for every chunk, which halves the frame rate. Retrieving latent frames by view frustum instead renders at 15 FPS, and is less consistent on revisits and moved objects in dynamic scenes (Sec. B, Tab. A3, and Fig. A1). Compression shrinks the additional K/V store 18-fold at no loss and

allows rollouts of over 8,000 frames within 45 GiB (Fig. 7b). Revisiting rollouts and the self-distillation loss each improve revisit, and FastVAE doubles the frame rate.

6. Conclusion

We present M2R, a real-time video renderer that stores what it generates on the mesh it renders. As it addresses memory by surface, M2R more than doubles the best baseline's object permanence and nearly doubles cross-agent consistency, with memory that grows only with the surfaces seen, while rendering 720p video at 40 FPS. Although a coarse mesh suffices, M2R assumes that it is accurate, and handling noisy geometry reconstructed from images [5] would be a valuable next step. We discuss limitations in Sec. H.

Acknowledgments

We thank Rafał Karp, Gene Liu, and the Isekai3D team [39] for developing and supporting the Isekai3D synthetic data generation platform.

References

[1] Philip J. Ball, Jakob Bauer, Frank Belletti, Bethanie Brownfield, Ariel Ephrat, Shlomi Fruchter, Agrim Gupta, Kristian Holsheimer, Aleksander Holynski, Jiri Hron, et al. Genie 3: A new frontier for world models. https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/, 2025. Google DeepMind blog post. 2

[2] Aarti Basant, Amlan Kar, Despoina Paschalidou, Fangyin Wei, Francesco Ferroni, Guillermo Garcia Cobo, Haithem Turki, Huan Ling, Jaewoo Seo, James Lucas, Jay Zhangjie Wu, Jialiang Wang, Jonathan Lorraine, Jun Gao, Kai He, Katarína Tóthová, Kevin Xie, Michał Tyszkiewicz, Qi Wu, Riccardo de Lutio, Ruilong Li, Sanja Fidler, Seung Wook Kim, Tianchang Shen, Tianshi Cao, Tobias Pfaff, William Lew, Xindi Wu, Xuanchi Ren, Yifan Lu, Yuxuan Zhang, Zan Gojcic, and Zian Wang. NVIDIA OmniDreams: Real-time generative world model for closed-loop autonomous vehicle simulation. arXiv preprint arXiv:2606.03159, 2026. 26

[3] Ollin Boer Bohan. TAEHV: Tiny autoencoder for Hunyuan video. https://github.com/madebyollin/taehv, 2025. 21

[4] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *ICML*, 2024. 2

[5] Alessandro Burzio, Tobias Fischer, Sven Elflein, Qunjie Zhou, Riccardo de Lutio, Jiawei Ren, Jiahui Huang, Shengyu Huang, Marc Pollefeys, Laura Leal-Taixé, Zan Gojcic, and Haithem Turki. Déjà View: Looping transformers for multi-view 3D reconstruction. In *NeurIPS*, 2026. 9

[6] Shengqu Cai, Duygu Ceylan, Matheus Gadelha, Chun-Hao Paul Huang, Tuanfeng Yang Wang, and Gordon Wetzstein. Generative rendering: Controllable 4D-guided video generation with 2D diffusion models. In *CVPR*, 2024. 3

[7] Ziqi Cai, Siqi Yang, Yimu Wang, Zixian Gao, Yunheng Liu, Shuchen Weng, Erwin Wu, Kaipeng Zhang, and Boxin Shi. MASS: Multiplayer world models with authoritative shared state. arXiv preprint arXiv:2608.06257, 2026. 3

[8] Wei Cao, Hao Zhang, Vikram Voleti, Yuqun Wu, Mallikarjun B R, Shimon Vainer, Mark Boss, and Yaoyao Liu. 4Director: Controlling video world models with rigid 3D geometry. arXiv preprint arXiv:2610.02160, 2026. 3

[9] Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Hung-Yueh Chiang, Yash Akhauri, Xilai Dai, Huiqiang Jiang, Yucheng Li, Luis Ceze, Kai-Chiang Wu, and Mohamed S. Abdelfattah. xKV: Cross-layer KV-cache compression via aligned singular vector extraction. In *ICML*, 2026. 5

[10] Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, Siddharth Patki, Ishita Prasad, Xavier Puig, Akshara Rai, Ram Ramrakhya, Daniel Tran, Joanne Truong, John M. Turner, Eric Undersander, and Tsung-Yen Yang. PARTNR: A benchmark for planning and reasoning in embodied multi-agent tasks. In *ICLR*, 2025. 8, 24

[11] Junhao Chen, Mingjin Chen, Henghaofan Zhang, Minglin Chen, Liaoyuan Fan, Boran Zhang, Saining Zhang, Mingze Sun, Hao Zhao, Ruqi Huang, Zhihao Li, and Yufei Wang. Video models as native 4D renderers: World-grounded conditioning from animated mesh. arXiv preprint arXiv:2608.00094, 2026. 3

[12] Kaijin Chen, Dingkang Liang, Xin Zhou, Yikang Ding, Xiaoqiang Liu, Pengfei Wan, and Xiang Bai. Out of sight but not out of mind: Hybrid memory for dynamic video world models. arXiv preprint arXiv:2603.25716, 2026. 3

[13] Yiwen Chen, Guosheng Lin, and Chi Zhang. Code World Model: Coding agent as world brain. arXiv preprint arXiv:2608.25927, 2026. 2, 3, 5, 6, 7, 15, 23, 24

[14] 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, 2026. 3

[15] Jiawei Chi, Shangchen Miao, Zhiyuan Shi, Kailu Wu, Hanyang Wang, Weiliang Chen, Qiyu Dai, Chengyang Li, Jinshan Ren, Jun Gao, Mingsheng Long, Yueqi Duan, Jiangran Lyu, Jialong Wu, and Fangfu Liu. AgentGarten: Code worlds for evolving agents. arXiv preprint arXiv:2610.12374, 2026. 2, 3, 5, 6, 7, 15, 23, 24

[16] LightX2V Contributors. Lightx2v: Light video generation inference framework. https://github.com/ModelTC/lightx2v, 2025. 21

[17] Riccardo de Lutio, Tobias Fischer, Yen-Yu Chang, Yuxuan Zhang, Jay Zhangjie Wu, Xuanchi Ren, Tianchang Shen,

Katarína Tóthová, Zan Gojcic, and Haithem Turki. ArtiFixer: Enhancing and extending 3D reconstruction with auto-regressive diffusion models. In *SIGGRAPH*, 2026. 3

[18] Weijia Dou, Hui Li, Jiahao Cui, Lei Zhou, Jingdong Wang, and Siyu Zhu. SlotMemory: Object-centric KV memory for streaming long-video generation. arXiv preprint arXiv:2605.31033, 2026. 3

[19] Zicheng Duan, Jiatong Xia, Zeyu Zhang, Wenbo Zhang, Gengze Zhou, Chenhui Gou, Yefei He, Feng Chen, Xinyu Zhang, and Lingqiao Liu. LiveWorld: Simulating out-of-sight dynamics in generative video world models. arXiv preprint arXiv:2603.07145, 2026. 3

[20] Enigma Labs. Introducing Multiverse: The first AI multiplayer world model. https://www.enigma.inc/posts/introducing-multiverse, 2025. Blog post; code at https://github.com/EnigmaLabsAI/multiverse. 2

[21] Zixun Fang, Yawen Shao, Kai Zhu, Jie Xiao, Shihan Chen, Yu Liu, Xueyang Fu, Yang Cao, Wei Zhai, and Zheng-Jun Zha. Code plans, diffusion renders: Open-ended generative world modeling. arXiv preprint arXiv:2609.26458, 2026. 2, 3

[22] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997. 16

[23] Songwei Ge, Aniruddha Mahapatra, Gaurav Parmar, Jun-Yan Zhu, and Jia-Bin Huang. On the content bias in fréchet video distance. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7277–7288. IEEE, 2024. 23

[24] Gonzalo Gomez-Nogales, Yicong Hong, Chongjian Ge, Peiye Zhuang, Marc Comino-Trinidad, Dan Casas, and Yi Zhou. Coarse-to-real: Generative rendering for populated dynamic scenes. In *NeurIPS*, 2026. 3

[25] Google DeepMind. Gemini 3.7 Flash model card. Model card, 2026. 8, 25

[26] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. GANcraft: Unsupervised 3D neural rendering of Minecraft worlds. In *ICCV*, 2021. 3

[27] Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, Baixin Xu, Hao-Xiang Guo, Kaixiong Gong, Size Wu, Wei Li, Xuchen Song, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-Game 2.0: An open-source, real-time, and streaming interactive world model. arXiv preprint arXiv:2508.13009, 2025. 2

[28] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2Room: Extracting textured 3D meshes from 2D text-to-image models. In *ICCV*, 2023. 2, 3

[29] Yicong Hong, Yiqun Mei, Chongjian Ge, Yiran Xu, Yang Zhou, Sai Bi, Yannick Hold-Geoffroy, Mike Roberts, Matthew Fisher, Eli Shechtman, Kalyan Sunkavalli, Feng Liu, Zhengqi Li, and Hao Tan. RELIC: Interactive video world model with long-horizon memory. arXiv preprint arXiv:2512.04040, 2025. 3

[30] Anthony Hu, Václav Volhejn, Adrien Ramanana Rahary, Chris Mulder, Aditya Makkar, Alyx Liao, Amélie Royer, Manu Orsini, Adam Jelley, Eloi Alonso, Florian Laurent, Fredrik Norén, James Swingos, Jan Hünermann, Kent Rollins, Lucas Hosseini, Matthieu Le Cauchois, Maxim Peter, Pim de Witte, Tim Brown, Vincent Micheli, Moritz Böhle, Gabriel de Marmiesse, Viktoriia Sharmanska, Lucia Specia, Michael Black, and Patrick Pérez. Multiplayer interactive world models with representation autoencoders. arXiv preprint arXiv:2607.05352, 2026. 2

[31] Teng Hu, Mingchun Lu, Yating Wang, Jiangning Zhang, Jinkun Hao, Ye Pan, Ran Yi, Lizhuang Ma, and Dacheng Tao. MetaWorld: Scaling multi-agent video world model from single-view video data. arXiv preprint arXiv:2606.02753, 2026. 3

[32] Junchao Huang, Xinting Hu, Boyao Han, Shaoshuai Shi, Zhuotao Tian, Tianyu He, and Li Jiang. Memory Forcing: Spatio-temporal memory for consistent scene generation on Minecraft. arXiv preprint arXiv:2510.03198, 2025. 3

[33] Junchao Huang, Guian Fang, Shengju Qian, Xianghao Kong, Zhuoran Zhao, Wei Huang, Yihua Du, Zixin Zhang, Justin Cui, Yuchao Gu, Yukang Chen, Xinting Hu, Tianyu He, Shaoshuai Shi, Zhuotao Tian, Xin Wang, Mike Zheng Shou, and Li Jiang. SolarWM: Open data and scalable training for long-horizon video world models. arXiv preprint arXiv:2609.02886, 2026. 5, 6, 7, 23, 24

[34] Kaiyi Huang, Wangbo Yu, Yukun Huang, Jiwen Yu, Kunhao Liu, Donglin Yang, Wenbo Hu, Ying Shan, and Xihui Liu. Multics: Generative multi-player world model with hierarchical communication. arXiv preprint arXiv:2610.17201, 2026. 2

[35] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self Forcing: Bridging the train-test gap in autoregressive video diffusion. In *NeurIPS*, 2025. 2, 4

[36] Zheng-Hui Huang, Guixu Lin, Jiacheng Lin, Yi-Chuan Huang, Ruihan Yu, Muyao Niu, Siqi Yang, Yu-Lun Liu, Yung-Yu Chuang, Kaipeng Zhang, and Zhixiang Wang. Programmable world model. arXiv preprint arXiv:2609.10540, 2026. 2, 3

[37] HunyuanWorld Team, Zhenwei Wang, Yuhao Liu, Junta Wu, Zixiao Gu, Haoyuan Wang, Xuhui Zuo, Tianyu Huang, Wenhuan Li, Sheng Zhang, Yihang Lian, Yulin Tsai, et al. HunyuanWorld 1.0: Generating immersive, explorable, and interactive 3D worlds from words or pixels. arXiv preprint arXiv:2507.21809, 2025. 2

[38] Minguk Kang and Suha Kwak. FlashDecoder: Real-time latent-to-pixel streaming decoder with transformers. arXiv preprint arXiv:2607.14898, 2026. 21, 22

[39] Rafal Karp, Gene Liu, Shih-Hsin Li, Eric Courtemanche, Shengze Wang, Josef Spjut, David Pankratz, Mischa Kolbe, Michael Stengel, Sven Elflein, et al. Isekai3d sdg: A platform for diverse synthetic gameplay visual data. *arXiv preprint arXiv:2611.03626*, 2026. 2, 6, 9, 19, 22, 26

[40] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X. Chang, and Manolis Savva. Habitat synthetic scenes dataset (HSSD-200): An analysis of 3D scene scale and realism tradeoffs for ObjectGoal navigation. In *CVPR*, 2024. 6, 8, 24

[41] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (ToG)*, 39(6):1–14, 2020. 16

[42] Runjia Li, Philip Torr, Andrea Vedaldi, and Tomas Jakab. VMem: Consistent interactive video scene generation with surfel-indexed view memory. In *ICCV*, 2025. 3

[43] Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Chih-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, and Zian Wang. DiffusionRenderer: Neural inverse and forward rendering with video diffusion models. In *CVPR*, 2025. 3

[44] Zhiyang Liang, Jiawei Zhang, Guocheng Qian, Dongdong Chen, and Ziyu Wan. Code Video Model. Preprint, 2026. 2, 3, 5, 6, 7, 23, 24

[45] Bin Lin, Yunyang Ge, Xinhua Cheng, Zongjian Li, Bin Zhu, Shaodong Wang, Xianyi He, Yang Ye, Shenghai Yuan, Liuhan Chen, et al. Open-sora plan: Open-source large video generation model. *arXiv preprint arXiv:2412.00131*, 2024. 22

[46] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023. 4

[47] Fangfu Liu, Kai He, Tianchang Shen, Tianshi Cao, Sanja Fidler, Yueqi Duan, Jun Gao, Igor Gilitschenski, Zian Wang, and Xuanchi Ren. Gamma-World: Generative multi-agent world modeling beyond two players. arXiv preprint arXiv:2605.28816, 2026. 2

[48] Hongyu Liu, Chun Wang, Feng Gao, Xuanhua He, Yue Ma, Ziyu Wan, Yong Zhang, Xiaoming Wei, and Qifeng Chen. OPSD-V: On-policy self-distillation for post-training few-step autoregressive video generators. arXiv preprint arXiv:2607.08766, 2026. 6

[49] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025. 22

[50] Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling Forcing: Autoregressive long video diffusion in real time. In *ICLR*, 2026. 5, 19

[51] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pages 9992–10002. Ieee, 2021. 21

[52] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 11966–11976. IEEE, 2022. 21

[53] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 20, 22

[54] Dongyue Lu, Guian Fang, Hur Lim, Yonggang Wu, Yanrong Wang, Hongshan Chen, Shengju Qian, Yue Liao, Wei Chow, Lingdong Kong, Tianxin Huang, Yu Yang, Hongji Yang, Junchao Huang, Ao Liang, Yihao Wang, Zihan Wang, Rong Li, Hanlin Chen, Xin Wang, Wei Tsang Ooi, Mike Zheng Shou, and Shuicheng Yan. Orbia: Do generated videos form a consistent 3D world? arXiv preprint arXiv:2610.20693, 2026. 5, 6, 7, 22, 24

[55] Yanzuo Lu, Tian Ye, Yitong Li, Shuchen Xue, Haozhe Liu, Song Han, and Enze Xie. StreamRender-H3: Interactive coding worlds with real-time streaming diffusion render. https://nvlabs.github.io/Sana/Sol-Engine/StreamRender-H3/, 2026. Project report; accessed 27 October 2026. 2, 3

[56] Wenchao Ma, Changran Liu, Sharon X. Huang, and Haomiao Jiang. Closing the loop: Training-free revisit consistency for autoregressive generative rendering. arXiv preprint arXiv:2607.21848, 2026. 3

[57] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *ECCV*, 2020. 2, 3

[58] Sicheng Mo, Yuheng Li, Ziyang Leng, Krishna Kumar Singh, and Bolei Zhou. Streaming multi-agent autoregressive diffusion model with world state registers. arXiv preprint arXiv:2607.21594, 2026. 2

[59] Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. Openvid-1m: A large-scale high-quality dataset for text-to-video generation. In *International conference on learning representations*, pages 1045–1064, 2025. 22

[60] NVIDIA. Cosmos 3: Omnimodal world models for physical AI. arXiv preprint arXiv:2606.02800, 2026. 6, 19, 26

[61] NVIDIA. DLSS 5: Real-time generative rendering. Technical report, NVIDIA, 2026. 3

[62] NVIDIA, Hassan Abu Alhaija, Jose Alvarez, Maciej Bala, Tiffany Cai, Tianshi Cao, Liz Cha, Joshua Chen, Mike Chen, Francesco Ferroni, Sanja Fidler, Dieter Fox, Yunhao Ge, et al. Cosmos-Transfer1: Conditional world generation with adaptive multimodal control. arXiv preprint arXiv:2503.14492, 2025. 3

[63] NVIDIA, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical AI. arXiv preprint arXiv:2501.03575, 2025. 2

[64] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024. 8, 25

[65] Ryan Po, David Junhao Zhang, Amir Hertz, Gordon Wetzstein, Neal Wadhwa, and Nataniel Ruiz. MultiGen: Level-design for editable multiplayer worlds in diffusion game engines. arXiv preprint arXiv:2603.06679, 2026. 3

[66] 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. 3, 5, 6, 7, 24

[67] 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*, 2025. 3, 5, 6, 7, 24

[68] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. TEXTure: Text-guided texturing of 3D shapes. In *SIGGRAPH*, 2023. 2, 3

[69] Stephan R. Richter, Hassan Abu Alhaija, and Vladlen Koltun. Enhancing photorealism enhancement. *TPAMI*, 2023. 2

[70] 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. 2, 3

[71] Tianchang Shen, Sherwin Bahmani, Kai He, Sangeetha Grama Srinivasan, Tianshi Cao, Jiawei Ren, Ruilong Li, Zian Wang, Nicholas Sharp, Zan Gojcic, Sanja Fidler, Jiahui Huang, Huan Ling, Jun Gao, and Xuanchi Ren. Lyra 2.0: Explorable generative 3D worlds. arXiv preprint arXiv:2604.13036, 2026. Accepted to SIGGRAPH Asia 2026. 2, 3, 5, 6, 7, 23, 24

[72] Joonghyuk Shin, Yicong Hong, Jaesik Park, and Xun Huang. Can video world models track unobserved world states? arXiv preprint arXiv:2608.30692, 2026. 3

[73] 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. 4, 21

[74] Konrad Staniszewski and Adrian Łańcucki. KV cache transform coding for compact storage in LLM inference. In *ICLR*, 2026. 2, 5

[75] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797, 2019. 5, 6

[76] Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. WorldPlay: Towards long-term geometric consistency for real-time interactive world modeling. In *ICML*, 2026. 2, 3, 5

[77] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimír Vondruš, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *NeurIPS*, 2021. 24

[78] Team HY-World, Chenjie Cao, Xuhui Zuo, Zhenwei Wang, Yisu Zhang, Junta Wu, Zhenyang Liu, Yuning Gong, et al. HY-World 2.0: A multi-modal world model for reconstructing, generating, and simulating 3D worlds. arXiv preprint arXiv:2604.14268, 2026. 2

[79] Haithem Turki, Qi Wu, Xin Kang, Janick Martinez Esturo, Shengyu Huang, Ruilong Li, Zan Gojcic, and Riccardo de Lutio. SimULi: Real-time LiDAR and camera simulation with unscented transforms. In *ICLR*, 2026. 26

[80] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *ICLR*, 2025. 2

[81] 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. 21

[82] Hanlin Wang, Hao Ouyang, Qiuyu Wang, Wen Wang, Qingyan Bai, Ka Leong Cheng, Yue Yu, Yixuan Li, Yihao Meng, Zichen Liu, Yanhong Zeng, Yujun Shen, and Qifeng Chen. WorldDirector: Building controllable world simulators with persistent dynamic memory. arXiv preprint arXiv:2607.02517, 2026. 3, 5, 6, 7, 23, 24

[83] Puyi Wang, Yuhao Wang, Linjie Li, Zhengyuan Yang, Kevin Qinghong Lin, Yangguang Li, and Yu Cheng. SceneCode: Executable world programs for editable indoor scenes with articulated objects. arXiv preprint arXiv:2605.19587, 2026. 2

[84] Weijie Wang, Haoyu Zhao, Yifan Yang, Feng Chen, Zeyu Zhang, Yefei He, Zicheng Duan, Donny Y. Chen, Yuqing Yang, and Bohan Zhuang. Latent spatial memory for video world models. arXiv preprint arXiv:2606.09828, 2026. 3

[85] Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, Yidan Xietian, Jiangbo Pei, Liang Hu, Boyi Jiang, Hua Xue, Zidong Wang, Haofeng Sun, Wei Li, Wanli Ouyang, Xianglong He, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-Game 3.0: Real-time and streaming interactive world model with long-horizon memory. arXiv preprint arXiv:2604.08995, 2026. 3

[86] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-Rings watermarks: Invisible fingerprints for diffusion images. In *NeurIPS*, 2023. 26

[87] Haoyu Wu, Jiwen Yu, Yingtian Zou, and Xihui Liu. MultiWorld: Scalable multi-agent multi-view video world models. arXiv preprint arXiv:2604.18564, 2026. 2

[88] Jay Zhangjie Wu, Yuxuan Zhang, Haithem Turki, Xuanchi Ren, Jun Gao, Mike Zheng Shou, Sanja Fidler, Zan Gojcic, and Huan Ling. DIFIX3D+: Improving 3D reconstructions with single-step diffusion models. In *CVPR*, 2025. 3

[89] Tong Wu, Shuai Yang, Ryan Po, Yinghao Xu, Ziwei Liu, Dahua Lin, and Gordon Wetzstein. Video world models with long-term spatial memory. In *NeurIPS*, 2025. 2, 3

[90] Xindi Wu, Sven Elflein, James Lucas, Olga Russakovsky, Laura Leal-Taixé, Despoina Paschalidou, Jonathan Lorraine, and Aljoša Ošep. Addressable memory for video world models. arXiv preprint arXiv:2608.07408, 2026. Accepted to NeurIPS 2026. 19

[91] Haocheng Xi, Shuo Yang, Yilong Zhao, Muyang Li, Han Cai, Xingyang Li, Yujun Lin, Zhuoyang Zhang, Jintao Zhang, Xiuyu Li, Zhiying Xu, Jun Wu, Chenfeng Xu, Ion Stoica, Song Han, and Kurt Keutzer. Quant VideoGen: Auto-regressive long video generation via 2-bit KV-cache quantization. In *ICML*, 2026. 4

[92] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024. 5, 19

[93] Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. WorldMem: Long-term consistent world simulation with memory. In *NeurIPS*, 2025. 2, 3, 15

[94] 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. arXiv preprint arXiv:2606.17730, 2026. Accepted to NeurIPS 2026. 3

[95] Haiyang Xu, Zheng Ding, and Zhuowen Tu. RECAP-Forcing: Retaining content appearances for long video generation. arXiv preprint arXiv:2608.26671, 2026. 4

[96] Qianxun Xu, Xianfang Zeng, Xinyao Liao, Wei Cheng, Gang Yu, and Chi Zhang. ConsistWorld: Evidence routing for consistent multi-agent world models. arXiv preprint arXiv:2609.22641, 2026. 3

[97] Tian-Xing Xu, Zi-Xuan Wang, Guangyuan Wang, Li Hu, Zhongyi Zhang, Peng Zhang, Bang Zhang, and Song-Hai Zhang. UCM: Unified modeling of camera control and memory with time-aware positional encoding warping for world models. In *SIGGRAPH*, 2026. 5

[98] Zhucun Xue, Jiangning Zhang, Teng Hu, Haoyang He, Yinan Chen, Yuxuan Cai, Yabiao Wang, Chengjie Wang, Yong Liu, Xiangtai Li, et al. Ultravideo: High-quality uhd video dataset with comprehensive captions. *arXiv preprint arXiv:2506.13691*, 2025. 22, 23

[99] Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, Song Han, and Yukang Chen. LongLive: Real-time interactive long video generation. In *ICLR*, 2026. 4, 5

[100] Ziyang Ye, Junchao Huang, Evelyn Zhang, Zhihao Xie, Ruicheng Zhang, Boyao Han, Litao Ban, Ziye Wang, Xinting Hu, Shaoshuai Shi, Zhuotao Tian, and Li Jiang. WorldCast: Distributed multiplayer world models. arXiv preprint arXiv:2610.12412, 2026. 3

[101] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *ICCV*, 2023. 5, 6

[102] 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. 2, 3, 4

[103] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024. 4

[104] Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Frédo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *CVPR*, 2025. 4

[105] 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. 3, 15

[106] Wei Yu, Runjia Qian, Yumeng Li, Liquan Wang, Songheng Yin, Sri Siddarth Chakaravarthy P, Dennis Anthony, Yang Ye, Yidi Li, Weiwei Wan, and Animesh Garg. MosaicMem: Hybrid spatial memory for controllable video world models. arXiv preprint arXiv:2603.17117, 2026. 2, 3, 5

[107] Xiaoyu Zhan, Xinyu Wang, Xiaohong Zhang, Huanjie Zhu, Tengjiao Sun, Pengcheng Fang, Jiaxing Yu, Yanwen Guo, and Dongjie Fu. Magpie: Real-time world renderer for interactive games. arXiv preprint arXiv:2608.27168, 2026. 3

[108] Jihai Zhang, Runjia Qian, Kaiqin Kong, Biao Jiang, Puyi Wang, Siru Chen, Tianle Li, Guanjie Chen, Junyi Wu, Chubin Chen, Yuanhang Li, Kaichen Huang, Haobo Zhang, Yangguang Li, Wei Yu, Guibin Chen, Yu Cheng, and Yahui Zhou. Matrix-Game 4.0: Harnessing video models for real-time playable game worlds. arXiv preprint arXiv:2610.27999, 2026. 3

[109] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 23

[110] Xin Zhang, Yabo Chen, Zixuan Duan, Haibin Huang, Chi Zhang, Feng Xu, and Xuelong Li. TourPhysics: Bringing physics to world models for exploration and manipulation from a single image. arXiv preprint arXiv:2609.04911, 2026. 3

[111] Yiwen Zhang, Haocheng Xi, Michael Tian-Yue Liu, Alexei A. Efros, Hadar Averbuch-Elor, Qianqian Wang, and Haiwen Feng. MosaiChunk: Compositing spatio-temporal memory for autoregressive video generation. arXiv preprint arXiv:2610.02153, 2026. 3, 6

[112] 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. 5, 20

[113] Renjie Zhao, Yuxiang Wu, Mingyu Zhang, Jiaxin Li, Sisi Li, He Li, Yimin Sheng, Tianxi Tan, Zhenkai Zhang, Jiao Liang, Jianyi Zhu, and Yong-Lu Li. Population-scalable multi-agent world modeling. arXiv preprint arXiv:2608.08600v3, 2026. 3

[114] Siyuan Zhou, Yilun Du, Yuncong Yang, Lei Han, Peihao Chen, Dit-Yan Yeung, and Chuang Gan. Learning 3D persistent embodied world models. In *NeurIPS*, 2025. 3

[115] Hanxin Zhu, Jianqi Ma, Cong Wang, Zhihao Hu, Junyao Gao, Tianyu He, Junyan Lin, Jiayin Xiao, Jiateng Shou, Sen Liang, Weidong Zhang, Zhibo Chen, and Jun Zhang. Demiurge: Forging navigable 4D worlds via autoregressive video rerendering. arXiv preprint arXiv:2610.14685, 2026. 3

[116] Jiayi Zhu, Jianing Zhang, Yiying Yang, Wei Cheng, and Xiaoyun Yuan. ShareVerse: Collaborative video generation for shared world modeling. arXiv preprint arXiv:2603.02697, 2026. 3

A Mesh to Remember: Persistent Appearance for Interactive Video Worlds

Supplementary Material

Contents

A. Videos 15

B. Additional Diagnostics 15

C. Geometry Conditioning 16

D. Surface Memory 18

E. Training and Inference 19

F. FastVAE 21

G. Additional Details on Evaluation 22
- G.1. Static Environments 22
- G.2. Dynamic Environments 24
- G.3. Multi-Agent Environments 25

H. Limitations 26

I. Societal Impact 26

J. Additional Qualitative Results 26

A. Videos

We recommend watching our videos on our project page at https://research.nvidia.com/labs/sil/projects/m2r/. The page shows single-agent and multi-agent rollouts of M2R across diverse meshes, compares it to AgentGarten [15] and Code World Model [13] on reopened drawers, moved objects, and multiple agents, and includes an 8,001-frame rollout (4 min 27 s).

B. Additional Diagnostics

Memory without training. To test whether the causal renderer benefits from the memory without any training, we evaluate it before memory distillation (upper rows of Tab. A1). The memory already raises content persistence from 84.45 to 88.48, whereas widening the attention window from 8 to 64 frames only reaches 86.19 and renders slower (35.2 vs. 42.9 FPS). On HSSD, it also raises revisit from 74.33 to 81.98 (Tabs. A2 and 4).

Compression. We measure the impact of compression by varying the number of principal components kept per token and the precision at which we store them (Tab. A1). Content persistence improves with the number of components and saturates at 8,192. Storing these components at 8-bit precision yields a memory that is as consistent as the uncompressed one, both before and after memory distillation, with tokens that are 18 times smaller. Compression and decompression have a negligible impact on the frame rate.

Table A1. **Compression.** We vary the number of principal components kept per token and their precision on ScanNet++, before and after memory distillation (trained, 16 GPUs). Store reports the peak size of the 0.485 GiB temporal cache plus the stored tokens, which shrink 18× on their own (3.70 to 0.21 GiB). Our compressed memory is as consistent as uncompressed keys and values (Raw K/V) at a fraction of the storage.

Memory	Control Align.	Visual Quality	Content Persist.	Store (GiB)↓	FPS↑
None (8f window)	95.07	72.33	84.45	0.485	51.1
None (64f window)	94.42	71.43	86.19	2.177	35.2
Raw K/V	95.74	72.50	88.48	4.190	42.9
PCA 512	95.41	72.25	85.83	0.511	40.9
PCA 1k	95.57	72.45	87.65	0.537	40.7
PCA 2k	95.32	72.43	88.19	0.589	41.0
PCA 4k	95.48	72.40	88.65	0.691	40.7
PCA 8k	95.71	72.50	88.82	0.897	41.0
PCA 16k	95.22	72.52	88.84	1.309	40.6
PCA 8k, FP8	95.57	72.50	88.64	0.691	41.0
Raw K/V (trained)	95.90	72.62	89.74	4.190	42.9
M2R (trained)	95.86	72.68	89.83	0.691	41.1

Observations per cell. To validate retaining both the first and the latest observation of each cell, we compare to variants that retain only one of the two (Tab. A2). Retaining both is more consistent than either alone, as the first observation preserves the initial appearance of a surface and the latest its most recent one. Averaging the two instead of attending to each degrades consistency.

Memory structure. We ablate what the memory stores by comparing it to variants that instead store colors on the same surface cells or retrieve past latent frames by view frustum [93, 105]. We summarize our results in Tab. A3. In the color variants, each generated chunk writes RGB colors to the same surface cells. For each subsequent chunk, we re-render the stored colors (first, latest, or both) from the current camera, encode them with FastVAE, and pass them as retrieved tokens at $t-8$ under the same 16-GPU memory distillation. Storing keys and values is more consistent than storing colors on both content persistence and revisit, and renders faster, as colors are a lossy copy of what the renderer

Table A2. **Observations per cell.** We vary which observations each cell retains, before memory distillation. Retaining both the first and the latest observation is more consistent than retaining either alone or averaging the two. The frame rates differ within measurement noise.

Observations	ScanNet++ Content Persist.	HSSD Revisit	Store (GiB)↓	FPS↑
First only	84.66	78.90	0.625	40.8
Latest only	88.34	75.83	0.617	41.3
First + latest	88.64	81.98	0.691	41.0
Mean of first + latest	84.65	76.35	0.691	39.6

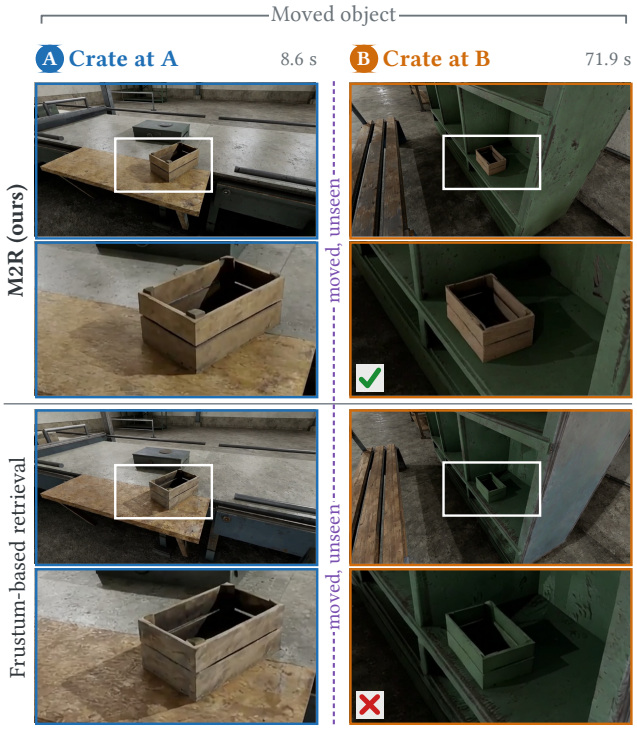

Figure A1. **Moved object under frustum retrieval.** A crate is seen at A and moved to B while the camera looks away. As the frames that show the crate at A share no view with B, frustum retrieval invents the crate's appearance there and draws it green. M2R keeps the crate's tokens in cells that move with the crate (Eq. (1)), and thus draws the wooden crate at B.

generated and must be re-rendered and re-encoded for every chunk. As a retrieved frame shows the scene as it was, retrieving past frames is less consistent in dynamic scenes (Fig. A1, 82.93 vs. 86.78 on revisit and 80.76 vs. 85.98 on moved objects), although it suffices on static scenes such as ScanNet++. On ScanNet++, every trajectory returns to its exact initial pose, and most intermediate revisits come from the same or nearby viewpoints. Our memory is both the fastest and the most consistent on revisits and moved objects in dynamic scenes.

Table A3. **Memory structure.** We compare our memory to variants that retrieve latent frames by view frustum or store colors (RGB) on the same surface cells, each after the same memory distillation (16 GPUs). Our memory is the fastest and the most consistent on revisits and moved objects in dynamic scenes.

Memory	ScanNet++ Content Persist.	HSSD Revisit	HSSD Moved Objects	FPS↑
Latent frames (frustum)	91.83	82.93	80.76	15.5
RGB, first	83.52	79.10	85.92	29.6
RGB, latest	85.06	78.56	84.55	29.0
RGB, first + latest	83.59	79.56	83.48	19.8
K/V (M2R)	89.83	86.78	85.98	41.1

C. Geometry Conditioning

Proxy rendering. At every video frame, we rasterize the current mesh from each agent's camera into a proxy video $\mathbf{R}$ (Fig. A2). We use flat shading under a fixed directional light and assign each asset one arbitrary color, which separates objects but carries no information about their appearance. Pixels without geometry, such as the sky, remain black. We encode $\mathbf{R}$ with the same autoencoder $\mathcal{E}$ as the video and add it to the video tokens through a zero-initialized projection W_m, such that each token i receives the geometry it should depict:

$$\mathbf{h}_i = W_v\,\mathcal{P}(\mathbf{z}_\tau)_i + W_m\,\mathcal{P}\big(\mathcal{E}(\mathbf{R})\big)_i, \tag{A1}$$

where $\mathbf{h}_i$ is the embedding of token i, $\mathcal{P}$ patchifies latents, and W_v is the pretrained projection of the noisy latents $\mathbf{z}_\tau$.

Online rasterization. At inference and during memory distillation, we rasterize the mesh on the GPU with nvdiffrast [41], in the same process as the renderer. Besides the proxy video, the rasterizer provides the depth, object, and surface normal behind each pixel, which the surface memory uses to assign pixels to cells (Sec. D). To limit the rasterizer's GPU memory during training, we cap the scene geometry at 4 GiB (about 70 million triangles) and simplify larger training scenes offline to 1% of the triangles of each asset with quadric error metrics [22].

Agents. We draw each agent into the proxy video as a bounding box (Fig. A2), from which the renderer generates the agent's body and motion. The hue of a box is random per episode and independent of the agent's identity, while its sides differ in brightness to show the agent's heading. All human characters share a box of $68 \times 72 \times 184$ cm, and each non-human character has a box that fits its asset. A third-person view shows the box of the agent that it follows and the boxes of all other agents in view, whereas a first-person view shows only the boxes of the other agents.

Keyboard controls. The renderer does not require keyboard controls, which serve as a hint for how to animate an

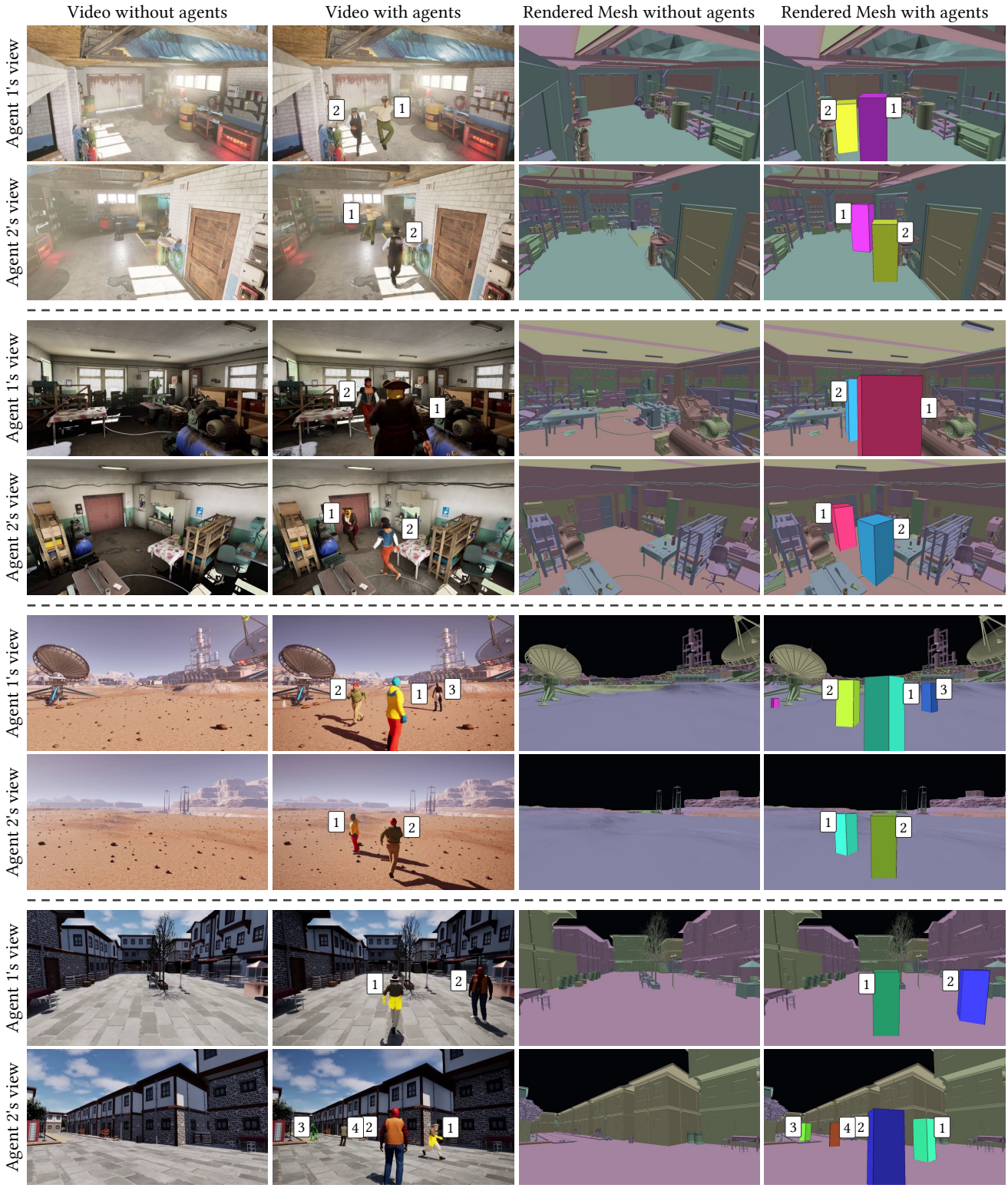

Figure A2. **Proxy video.** We show training videos (**left**) and their proxy videos (**right**) for four episodes, each rendered without and with agents. The proxy videos carry the geometry of each view but none of its appearance, and show each agent as a box (the numbers mark the same agent in both views).

Table A4. **Cells for every kind of pixel.** We list where the cells are defined and their sizes. Every pixel of a view thus maps to a cell, so that appearance persists on surfaces, on agents, and where the mesh has no geometry.

Pixel on	Cells defined in	Cell sizes
Mesh surface	Object coordinates, split by six surface directions	3, 12, 48 cm, 1.92 m
Agent box	Box coordinates	5, 20 cm
Hole	World coordinates, at an interpolated depth	12, 48 cm
Sky	Viewing direction, per 2 m cell of the camera position	8°, 32°

agent. During training, each view receives the keys held at each video frame by the agent that it follows (W, A, S, D, space, shift, and ctrl), or a flag that marks them as unknown if the clip has no agents. We embed the keys of the four video frames that each latent frame spans with a two-layer SiLU MLP whose last layer is zero-initialized, and add the embedding to every token of that latent frame. At evaluation, including episodes with agents, keys are unavailable, so instead of synthesizing them from the bounding boxes, we simply mark them as unknown, and the renderer still animates each agent naturally from its box alone.

D. Surface Memory

Cells. We extend the cells of Eq. (1) so that every pixel of a view maps to a cell, and what the renderer generates thus persists on distant surfaces, on thin structures, on agents, and where the mesh has no geometry (Tab. A4). As a pixel covers a larger patch of a surface the farther away the surface is, we keep cells at four sizes s_ℓ, from 3 cm to 1.92 m, and let each view write to those that match what its pixels cover. As a cubic cell can hold surfaces that face different ways, such as the two sides of a thin wall, we also split each cell by the direction $d(p)$ that its surface faces, and Eq. (1) becomes

$$c_\ell(p) = \Big(o(p),\ \big\lfloor \mathbf{T}_{o(p)}(t)\,\mathbf{x}(p)/s_\ell \big\rfloor,\ d(p)\Big), \tag{A2}$$

where $d(p)$ is one of six directions, the dominant axis and sign of the surface normal in the object's coordinates. An agent's box has cells of its own in the coordinates of the box, such that the agent carries its appearance with it (Tab. 3).

Pixels without a surface behind them have no cell under Eq. (A2). Where such a pixel is enclosed by geometry, as in the holes of scanned meshes (Fig. A10), we interpolate its depth from its surroundings and use the cells of the resulting point. Otherwise, as for the sky, we define its cells by viewing direction per 2 m camera-position cell, so they also remain bounded by the explored region of the scene.

Writing. Each view writes only to the cells whose size matches the scale at which it sees a surface (fine cells for a close-up and coarse cells for a distant view). We measure this scale by the footprint $f(p)$ of each pixel p, the width of the patch of surface that it covers. As a token j spans many pixels, it covers a larger patch of width $F(j)$. On mesh surfaces, a pixel p of token j writes to the cells of every size from $f(p)$ to $F(j)$,

$$\mathcal{W}(p) = \{\ell : f(p) \le s_\ell \le F(j)\}, \quad p \in \Omega_j. \tag{A3}$$

If no size lies in this range, the pixel ignores the upper bound and writes to the smallest $s_\ell \ge f(p)$. Agent boxes, holes, and the sky each have only two cell sizes (Tab. A4), and their pixels write to both. In practice, we rasterize the mesh at half the video resolution for efficiency, as cells and tokens are both far coarser than a pixel. A token then spans 16×16 pixels, and we set $F(j)$ to 16 times the median footprint of its pixels.

Storage. We compress each stored token 18-fold without degrading consistency (Tab. A1). The keys and values of a token form a vector $\mathbf{v}$ of $2 \times 36 \times 8 \times 128 = 73{,}728$ numbers, over 36 layers with 8 heads of 128 dimensions. We fit a principal component basis to these vectors once, on rollouts of the causal renderer before memory distillation. We then project each vector onto this basis and store its coefficients at 8-bit precision (FP8),

$$\mathbf{q} = \mathrm{FP8}\Big(\mathbf{\Lambda}^{-1}\,\mathbf{U}^\top \mathbf{S}^{-1}(\mathbf{v} - \boldsymbol{\mu})\Big), \tag{A4}$$

where $\boldsymbol{\mu}$ is the mean of $\mathbf{v}$, the diagonal matrix $\mathbf{S}$ holds its standard deviation per head, the columns of $\mathbf{U}$ are the 8,192 leading principal components, and the diagonal matrix $\mathbf{\Lambda}$ scales each coefficient to unit variance. When a token is read, we invert these steps, $\hat{\mathbf{v}} = \boldsymbol{\mu} + \mathbf{S}\,\mathbf{U}\mathbf{\Lambda}\,\mathbf{q}$. Each stored token takes 8,192 bytes instead of 147,456 at 16-bit precision. On the 46 ScanNet++ trajectories, the stored tokens thus take at most 0.21 GiB instead of 3.70 GiB. Over the 8,001-frame rollout of Fig. 7b, they take 5.95 GiB instead of 107 GiB. Even a rollout that kept every token it generates would add 1.7 GiB per 1,000 frames at 704×1280 instead of 30 GiB.

Reading. As in writing, each pixel reads at the scale at which it sees a surface. On mesh surfaces, a pixel p reads the finest cell that holds an observation and is at least as large as its footprint ($s_\ell \ge f(p)$). Without this bound, a distant view would read the fine cells of an earlier close-up instead of the cells at its own scale. For agent boxes, holes, and the sky, we omit this bound, and a pixel reads the finest of its two cells that holds an observation. Each token only retrieves past tokens from earlier visits to a surface. We thus exclude from the selection of Eq. (2) the observations made since the surface last came into view, as well as those whose tokens are still in the temporal cache, as the reading token attends to them directly. The selection with the latest observations also skips the chunk already selected with the

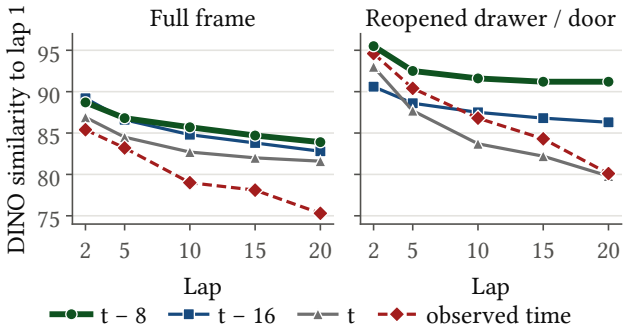

Figure A3. **Time position of retrieved tokens.** We compare four positions: 8 or 16 latents before the reading token ($t-8$, $t-16$), its own time (t), and the time at which the retrieved tokens were observed. For this figure, we change the position at inference only and plot an earlier renderer trained with retrieved tokens at t (our final renderer is trained with them at $t-8$). We plot how similar each lap is to the first, over the full frame (**left**) and inside reopened drawers and doors (**right**), for 20 laps of 13 HSSD loops (about 340 frames per lap, 7 minutes in total). Whereas Fig. 7a measures consistency on a single revisit at varying time intervals, here each lap is compared to the first across 20 repeated revisits of the same loop. Our choice ($t-8$) is the most consistent over long rollouts, while the observed time degrades the most, as its distance to the reading token keeps growing.

first observations, such that the two retrieved tokens come from different chunks.

Memory conditioning. In every layer, each token attends to the tokens retrieved for its chunk, both in the denoising steps and when the generated chunk is re-encoded for the temporal cache. We decompress the retrieved tokens into keys and values before attention, and never add them to this cache. As keys are stored without positional encoding [50, 90, 92], we apply the backbone's rotary encoding [60] when they are read. We give each retrieved token the spatial position of the reading token and a time that is a fixed 8 latents (32 frames) earlier, regardless of when it was observed. A token retrieved by token i at latent time t_i and spatial position (h_i, w_i) is thus placed at

$$(t_i - 8,\ h_i,\ w_i). \tag{A5}$$

With chunks of 2 latents and a cache of one chunk, the current chunk sits at $\{t-1, t\}$, the previous chunk at $\{t-3, t-2\}$, and the re-anchored initial chunk at $t-4$, such that the retrieved tokens at $\{t-9, t-8\}$ lie beyond the temporal cache. We compare this placement to alternatives in Fig. A3. Larger offsets (e.g., 16 or 32 latents) also work, while placing retrieved tokens at the time they were observed is less consistent over long rollouts.

Multiple agents. Several agents can share the surface memory although we train with a single view at a time, as the memory indexes tokens by the surfaces they depict rather than by the view that generated them (Tab. 3). At inference, all agents generate a chunk before any of them writes, and their observations then enter the memory together, ordered by agent index where several agents write to the same cell. The result thus does not depend on the order in which the agents run. Each agent runs on its own GPU. Each agent reads the observations of the other agents under the same rules as its own, including those from their first frames, which are not in its temporal cache. It skips the observations that another agent made in the previous chunk or since the surface last came into its own view. Two agents that look at the same surface would otherwise read each other's latest frames at every chunk, causing temporal inconsistencies.

Agents also keep their appearance from one meeting to the next (Tab. 3). In third-person videos, each agent starts from a first frame that shows it only from behind, so another view that first sees its front has to guess its appearance. As a written guess would stay in the memory for the rest of the video, an agent's box is only written by its own view and by the given first frames of other views, which are not generated. A token on the box in turn attends only to the tokens retrieved for itself, as those retrieved for the rest of its chunk would mix in the appearance of other agents. It also attends to reference tokens of that agent, taken from the agent's own view (its first frame and its latest chunk that shows the agent) and from the given first frames of other views that show it. We empirically place these references 48 latents before the reading token (16 or 32 also work), apart from the retrieved tokens at $t-8$, and add $\ln 4$ to their attention logits, which weights them four-fold so that the renderer draws unseen sides of an agent consistently with its observed views (Sec. H).

E. Training and Inference

Data. Although we fine-tune Cosmos3-Nano [60] (Fig. A4) only on synthetic episodes rendered in Unreal Engine with Isekai3D [39], M2R transfers zero-shot to every dataset that we evaluate on (Sec. G), including real indoor scans and worlds authored by coding agents. The training set has about 200,000 episodes of 60 s at 30 FPS and 720×1280, which span 223 environments. Each episode has one to eight agents drawn from 367 human and 34 non-human characters, and every agent is followed by its own camera, which gives about 850,000 single-view videos. We render non-human characters only in single-agent episodes. We render each video a second time along the same camera path without agents, and each render comes with its proxy video (Fig. A2). Until memory distillation, half of the training clips show agents and the other half do not. We train on one view at a time, on clips of 129 to 513 frames at 480×832 (480p) or 704×1280 (720p). Despite never training on two views of a world together or on more than 513 frames, M2R nearly doubles the best cross-agent

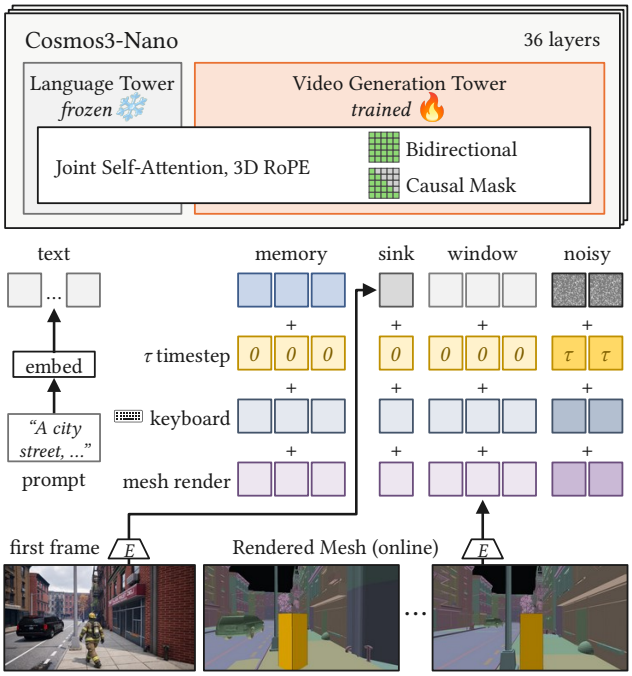

Figure A4. **Renderer architecture.** We build our renderer on Cosmos3-Nano and train only its video tower. Each video token receives the embeddings of its timestep, keyboard controls, and mesh render, and the text tokens are a fixed prefix. The same network runs with bidirectional attention in the first stage, with a causal mask in teacher forcing and Causal CD, and with an autoregressive key-value cache in Self Forcing and memory distillation. In these two stages, only the noisy chunk runs through the network, while the sink and window tokens (and the memory tokens in memory distillation) enter as a key-value cache that we commit from a clean pass. The cached keys of the sink and the window already carry their timestep and rotary position, whereas memory tokens are stored without rotary position, which we apply whenever they are read (Sec. D).

consistency among the baselines (Tab. 3) on episodes of two to four times that length (Sec. G.3).

Stages and hyperparameters. We train M2R in five stages on 64 GB200 GPUs (Tab. A5), starting each stage from the exponential moving average (EMA) of the weights of the previous one. We first train a bidirectional model for 23,000 iterations on clips of 129, 257, and then 513 frames at both resolutions, using AdamW [53] with a learning rate of 1×10^{-5}. We then make it causal with 2,000 iterations of teacher forcing at the same learning rate, and distill it into four steps with 500 iterations of Causal CD [112] at 2×10^{-6}. We next train it on its own rollouts with 2,000 iterations of Self Forcing, using learning rates of 2×10^{-6} for the generator and 4×10^{-7} for the fake score function. Throughout causal training, we draw each chunk's sliding window uniformly from one to 15 chunks or the entire past. Finally, we teach the renderer to use its memory with only

Table A5. **Training stages.** We train M2R in the order listed. Batch lists the clips per iteration at 480p / 720p. We draw both resolutions equally often for the first 19,000 iterations of the bidirectional stage, and 480p for one iteration in five for the rest of that stage and throughout the three causal stages.

Stage	Frames	Resolution	Iterations	Batch
Bidirectional	129	480p, 720p	13,000	576 / 256
	257	480p, 720p	2,000	256 / 128
	513	480p, 720p	8,000	256 / 128
Teacher forcing	513	480p, 720p	2,000	64
Causal CD [112]	513	480p, 720p	500	64
Self Forcing	513	480p, 720p	2,000	64
Memory distillation	513	720p	300	64

Algorithm A1 Memory distillation (one iteration)

Require: Renderer G_θ, its frozen copy $G_{\bar{\theta}}$, a clip of N chunks
Empty the surface memory $\mathcal{M}$ and pick the exit step k
for $n = 1, \dots, N$ **do**
  $\mathcal{R}_n \leftarrow \textsc{Read}(\mathcal{M})$ ▷ retrieved tokens
  $\hat{\mathbf{z}}_n^S \leftarrow G_\theta$ in k steps, given the cache and $\mathcal{R}_n$
  if the clip is a palindrome **then**
    $\hat{\mathbf{z}}_n^T \leftarrow G_{\bar{\theta}}$, given all earlier chunks
  end if
  Re-encode $\hat{\mathbf{z}}_n^S$, update the cache, $\textsc{Write}(\mathcal{M})$
end for
if generator iteration (one in six) **then**
  Update θ with $\mathcal{L}_{\text{DMD}} + \lambda \mathcal{L}_{\text{SD}}$
else
  Update the fake score function
end if

300 iterations of memory distillation on 513-frame clips at 704×1280, with $\lambda = 0.1$.

Memory distillation. In each iteration of memory distillation (Algorithm A1), the renderer rolls out all chunks of a clip with a temporal cache of one chunk, reading from and writing to the surface memory as at inference (Sec. D). In 70% of the iterations, the clip is a palindrome with one, two, or four round trips, which we build from agent-free clips only. In the rest, it is a clip with agents, played forward. We apply $\mathcal{L}_{\text{DMD}}$ to every clip, with a teacher and a fake score function that see the whole clip and have no memory. We apply $\mathcal{L}_{\text{SD}}$ to palindromes only and vary its weight λ in Fig. A5. In the ablation without revisiting rollouts (Tab. 4), we apply $\mathcal{L}_{\text{SD}}$ to forward agent-free clips using the same frozen teacher. A larger λ improves consistency but softens textures. Because the synthetic ground-truth renders in Fig. A5 are less vivid and less detailed than photorealistic video, we empirically choose $\lambda = 0.1$ for the best balance of visual quality and consistency across diverse scenes.

Sampling. At test time, M2R renders each eight-frame chunk in two steps, at noise levels 1 and 0.625, without classifier-free guidance. Although we distill the renderer into four steps, we find that two-step and four-step results differ

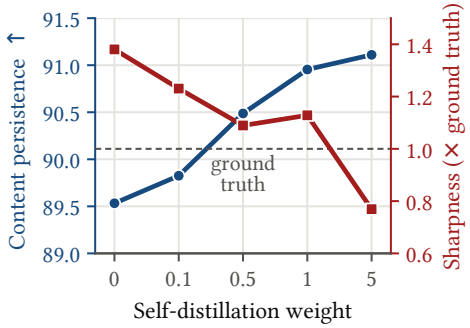

Figure A5. **Self-distillation weight.** We vary the weight λ of the self-distillation loss in memory distillation, training each model on 16 GPUs. We plot content persistence on ScanNet++ and sharpness relative to the ground truth on 513-frame revisiting trajectories in a world we build with a coding agent. A larger weight trades sharpness for consistency, and M2R uses 0.1.

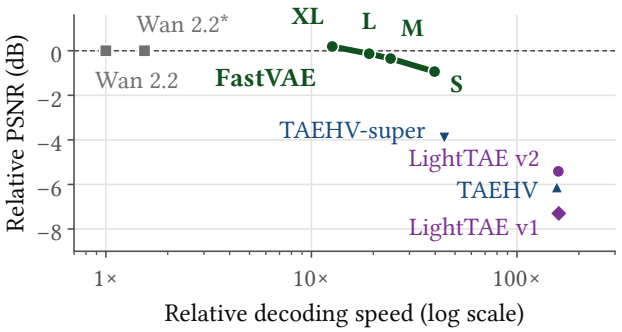

Figure A6. **Quality against speed.** We plot the PSNR of each decoder against its decoding speed, both relative to the Wan 2.2 decoder in eager mode and averaged over the three sizes of Tab. A7. The decoders run on the latents of the Wan 2.2 encoder, and we measure speed on a GB200. Wan 2.2 is the original decoder as it is commonly used, and Wan 2.2* is the same decoder optimized with `torch.compile` and CUDA graphs. All other decoders use the same optimization.

negligibly, as the mesh geometry and the memory provide strong signals. The renderer is trained on 30 FPS videos, as used in the 8,001-frame rollout of Fig. 7b, whereas our quantitative benchmarks follow Orbia's 16 FPS convention (Sec. G).

F. FastVAE

The Wan 2.2 VAE [81] of the base model provides high-quality reconstructions but is slow, taking 225 ms to decode an eight-frame chunk at 704×1280 on a GB200 (and still 143 ms when compiled), more than half of the 416 ms that M2R spends on the chunk when using it. Open-source tiny decoders for the same latents [3, 16, 73] are far faster but degrade PSNR by 4 to 7 dB (Fig. A6). FlashDecoder [38] instead replaces the decoder with a streaming transformer but keeps the original encoder, which M2R runs on the proxy video of every chunk (Sec. C). We thus build upon FlashDecoder and learn the encoder as well, allowing FastVAE to replace the entire Wan 2.2 VAE without changing its latent space (Fig. A7). FastVAE matches the quality of the original while running an order of magnitude faster (Tab. A7), more than doubling the frame rate of M2R (19.2 to 41.1 FPS) without degrading consistency (Tab. 4).

Architecture. As in FlashDecoder, our decoder is a streaming transformer that flattens each latent frame of size $H' \times W'$ into $H'W'$ tokens, where each latent frame attends to itself and the previous one through cached keys and values (Fig. A7). However, FlashDecoder attends over the whole frame, at a cost that grows quadratically with resolution. We instead use Swin Transformer blocks [51], which attend within 16×16 windows (shifted by 8 tokens in every other block) at a cost that grows only linearly. With full-frame attention, the same decoder takes 37% longer at 704×1280 and 99% longer at 1088×1920 on a GB200,

Table A6. **FastVAE decoder sizes.** N and d are the number and width of the Swin Transformer blocks at latent resolution, d_r is the width of the two blocks that refine the four frames of each latent, and d_up gives the widths of the two upsampling stages. All sizes use the same encoder architecture, 12 blocks of width 768 with 106M parameters.

Size	N	d	d_r	d_up	Params (M)
S	8	1024	256	128, 64	125.5
M	12	1024	512	256, 128	195.3
L	12	1536	512	256, 128	417.3
XL	16	1536	512	512, 256	559.9

with no noticeable gain in quality. To map tokens to pixels, FlashDecoder uses a single $16\times$ pixel shuffle, whereas we upsample progressively, by $2\times$ with Swin Transformer blocks, by $2\times$ with ConvNeXt blocks [52], and by $4\times$ with a pixel shuffle. We build the encoder from the same blocks over spatiotemporal patches of $4 \times 16 \times 16$ pixels, such that it encodes a chunk in 3.5 ms instead of the 63.5 ms of the Wan 2.2 encoder. The decoder comes in four sizes (Tab. A6), which trade speed for quality (Fig. A6). Throughout the paper, M2R uses FastVAE-XL, for both its scores and its frame rate.

Training. As FastVAE has to stay in the latent space of the Wan 2.2 VAE, we train its decoder and encoder one at a time against the frozen original, and then both end to end. The decoder first learns to reconstruct a clip $\mathbf{x}$ from the latents of the frozen Wan 2.2 encoder, at 224×224 and then at 480p to 1080p, with

$$\mathcal{L}_\mathrm{dec} = \|\hat{\mathbf{x}} - \mathbf{x}\|_1 + \lambda_2 \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 + \lambda_\mathrm{p} \,\mathrm{LPIPS}(\hat{\mathbf{x}}, \mathbf{x}), \quad \text{(A6)}$$

where $\lambda_2 = 10$ and $\lambda_\mathrm{p} = 0.5$. We drop the adversarial loss of FlashDecoder, which had a negligible effect on our

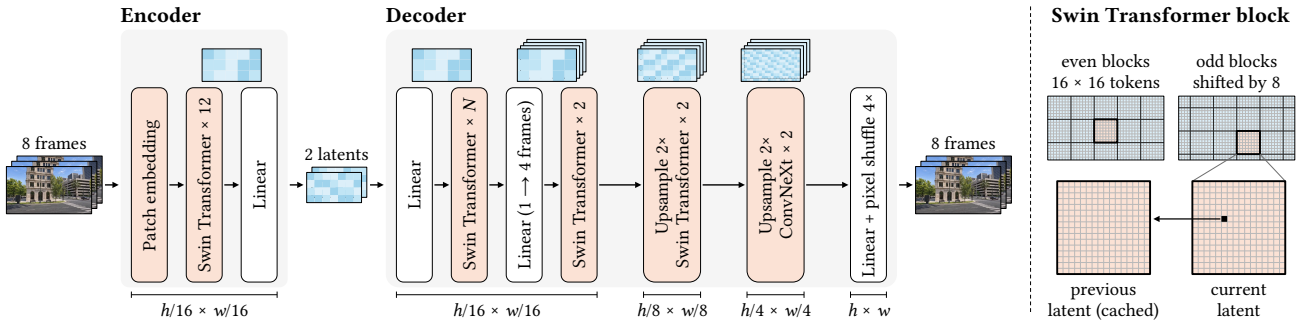

Figure A7. **FastVAE.** The encoder maps each chunk of eight frames to two latents in the latent space of the Wan 2.2 VAE, and the decoder maps them back to frames. Swin Transformer blocks attend inside shifted windows to the current and the previous latent, whose keys and values are cached, so both networks run chunk by chunk. Table A6 lists the number of blocks N and the widths of each decoder size.



Figure A8. **Reconstruction at 1080p.** We decode the Wan 2.2 latents of an UltraVideo clip [98] with two tiny decoders (**a**, **b**), the Wan 2.2 decoder (**c**), and FastVAE-XL (**d**), and show the region in the red box with the PSNR of each decoder on the clip. The tiny decoders lose the bricks and the window bars, while FastVAE-XL is as close to the ground truth (**e**) as the Wan 2.2 decoder. The layout is inspired by FlashDecoder [38].

decoder. The encoder then learns to regress the same latents with the decoder frozen, and a final end-to-end phase trains both networks to reconstruct the clip through the encoder and the decoder. To match the general-purpose latent space of the Wan 2.2 VAE, all phases train FastVAE on a mixture of publicly released web videos [45, 59] and synthetic episodes rendered with Isekai3D [39], whereas the renderer itself is fine-tuned only on synthetic data (Sec. E). We optimize weight matrices with Muon [49] and the remaining parameters with AdamW [53], on 64 A100 GPUs.

Speed and fidelity. Following FlashDecoder, we evaluate reconstruction on UltraVideo [98] (Tab. A7 and Fig. A6). All four sizes of FastVAE outperform the open-source tiny decoders in every metric and at every resolution, and the smallest (FastVAE-S) is already 3 dB above the best of them at a similar decoding time (5.7 vs. 5.1 ms). FastVAE-XL exceeds the PSNR of the Wan 2.2 decoder at every resolution while decoding 8× faster than its compiled version, and stays within 0.5 dB of the full Wan 2.2 VAE with its own encoder, which is 11× faster. Qualitatively, it preserves the bricks and window bars that the tiny decoders lose (Fig. A8). In M2R, FastVAE more than doubles the frame rate and lowers peak GPU memory from 42.7 to 33.6 GiB, while content persistence (89.83 vs. 89.76) and revisit (86.78 vs. 86.52) are preserved (Tab. 4).

G. Additional Details on Evaluation

G.1. Static Environments

Trajectories. On ScanNet++, we use the trajectories that Orbia [54] releases, one for each of 60 scenes, in which the camera leaves a view and returns to it. In 14 of them, the camera passes through a wall, behind which both the mesh render and the ground truth show the back of the wall or nothing (Fig. A9). We thus evaluate on the remaining 46

Table A7. **Reconstruction quality and speed.** We reconstruct 25-frame clips of the 1,145 UltraVideo videos [98] at three resolutions, with decoders on Wan 2.2 latents (**top**) and with each model's own encoder (**bottom**). We report PSNR, LPIPS with VGG features [109], rFVD (the content-debiased FVD [23] on the first 16 frames of each clip), and the time per eight-frame chunk at 704 × 1280 on a GB200 with `torch.compile` and CUDA graphs. FastVAE-XL stays within 0.5 dB of Wan 2.2 in PSNR at every resolution while encoding 11× and decoding 8× faster. Against Wan 2.2 in the common eager mode (63.5 ms to encode and 224.8 ms to decode), it is 18× and 13× faster.

		Params (M)		Time (ms)↓		480p (480 × 832)			720p (704 × 1280)			1080p (1088 × 1920)		
Encoder	Decoder	Enc.	Dec.	Enc.	Dec.	PSNR↑	LPIPS↓	rFVD↓	PSNR↑	LPIPS↓	rFVD↓	PSNR↑	LPIPS↓	rFVD↓
Wan 2.2	Wan 2.2	149.6	555.0	38.8	142.7	36.69	0.037	4.24	38.32	0.036	3.01	40.23	0.035	2.03
Wan 2.2	TAEHV	149.6	9.9	38.8	1.4	30.72	0.122	17.02	32.21	0.115	11.32	33.94	0.107	8.69
Wan 2.2	TAEHV-super	149.6	19.0	38.8	5.1	32.62	0.095	8.55	34.39	0.088	5.59	36.49	0.082	4.03
Wan 2.2	LightTAE v1	149.6	9.9	38.8	1.4	29.87	0.194	23.29	31.06	0.196	17.79	32.40	0.198	15.06
Wan 2.2	LightTAE v2	149.6	9.9	38.8	1.4	31.37	0.145	17.62	32.91	0.132	11.36	34.72	0.120	8.32
Wan 2.2	FastVAE-S	149.6	125.5	38.8	5.7	35.62	0.062	6.03	37.37	0.057	3.85	39.44	0.053	2.55
Wan 2.2	FastVAE-M	149.6	195.3	38.8	9.3	36.25	0.054	5.31	37.99	0.050	3.51	39.94	0.046	2.41
Wan 2.2	FastVAE-L	149.6	417.3	38.8	11.6	36.45	0.052	5.02	38.19	0.048	3.29	40.19	0.045	2.22
Wan 2.2	FastVAE-XL	149.6	559.9	38.8	17.4	36.77	0.048	4.77	38.50	0.045	3.15	40.53	0.041	2.06
TAEHV	TAEHV	1.5	9.9	1.0	1.4	30.25	0.140	21.68	31.64	0.131	14.52	33.25	0.123	11.12
TAEHV-super	TAEHV-super	1.5	19.0	1.0	5.1	31.85	0.124	13.32	33.36	0.116	8.82	35.11	0.108	6.83
LightTAE v1	LightTAE v1	1.5	9.9	1.0	1.4	28.09	0.251	56.02	29.04	0.248	46.95	30.05	0.245	40.02
LightTAE v2	LightTAE v2	1.5	9.9	1.0	1.4	28.74	0.215	52.16	29.76	0.201	40.32	30.86	0.188	33.14
FastVAE-S	FastVAE-S	106.2	125.5	3.5	5.7	35.42	0.074	6.12	37.17	0.067	4.05	39.24	0.060	2.65
FastVAE-M	FastVAE-M	106.2	195.3	3.5	9.3	35.87	0.067	5.69	37.62	0.061	3.69	39.61	0.055	2.59
FastVAE-L	FastVAE-L	106.2	417.3	3.5	11.6	36.02	0.065	5.45	37.77	0.059	3.59	39.82	0.054	2.39
FastVAE-XL	FastVAE-XL	106.2	559.9	3.5	17.4	36.24	0.062	5.09	37.99	0.056	3.30	40.05	0.051	2.24

and report all 60 in Tab. A9. Including these 14 trajectories lowers every geometry-conditioned method across all metrics except input preservation while leaving SolarWM H3 largely unchanged, yet M2R still ranks first in four of the six metrics (control alignment, input preservation, content persistence, and temporal stability). On Replica, which Orbia does not include, we generate four trajectories in each of the 18 scenes (72 in total) with Orbia's trajectory generator, and render the input frame and the ground-truth frames used for scoring from the textured mesh. On both datasets, each trajectory has 357 frames (22 s at 16 FPS) at 720 × 1280.

Mesh repair and scoring. On ScanNet++, we run every method that takes geometry on two versions of each mesh, the scanned one and one repaired by an LLM agent, as the scans have holes and artifacts (Fig. A10). For each trajectory and metric, we report the better of a method's two scores, for the baselines and M2R alike, such that no method is penalized by the defects of a scan or by its repair. We list the scores on each mesh separately, on the 46 trajectories, in Tab. A8. The repair raises the scores of most of these methods, by up to 2.7 points for GEN3C and by at most 0.7 for M2R. Even with the scanned meshes alone, M2R ranks first in control alignment, input preservation, content persistence, 3D consistency, and temporal stability.

Baselines. We run every baseline with its released sampling steps and context window. Lyra 2.0 [71] and WorldDirector [82] generate at 480×832 and Code Video Model [44] at 544 × 960, the only resolutions they support, and AgentGarten [15] at 480 × 832, the resolution it is trained at.

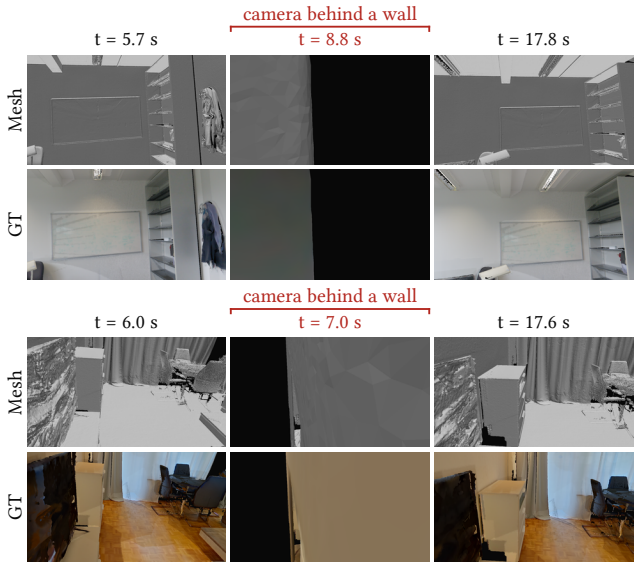

Figure A9. **Excluded trajectories.** We show two of the 14 ScanNet++ trajectories that we exclude, as the mesh render and the scan rendered with its colors (GT), before, while, and after the camera is behind a wall. Behind the wall, the view shows the back of the wall or nothing.

SolarWM [33] and Code World Model [13] generate at 768×1344, and the remaining methods at 704×1280. Methods that generate fixed-length blocks pad the last block with the final pose. As Code Video Model has no input to continue a video, we generate each trajectory in 124-frame windows and give each window the last frame of the previous one as

Table A8. **ScanNet++ on each mesh.** We report Orbia scores [54] on the 46 trajectories of Tab. 1 with the ground-truth mesh (**left**) and the LLM-repaired mesh (**right**), whereas Tab. 1 keeps the better of the two for each trajectory and score. SolarWM takes no geometry, so its two sides are the same videos, a number in parentheses counts the trajectories on which a score is undefined and counted as 0, and † and § mark methods that generate at 480p and 540p.

Method	ScanNet++, ground-truth mesh (46)						ScanNet++, LLM-repaired mesh (46)					
	Control Align.	Visual Quality	Input Pres.	Content Persist.	3D Consist.	Temp. Stab.	Control Align.	Visual Quality	Input Pres.	Content Persist.	3D Consist.	Temp. Stab.
SolarWM H3 [33]	83.01	71.72	71.46 (1)	80.34 (1)	74.96	87.22	83.01	71.72	71.46 (1)	80.34 (1)	74.96	87.22
Matrix-Game 3.5 [66]	56.93	68.49	70.01 (1)	49.27 (8)	49.31	72.77	57.05	68.88	70.51 (1)	47.19 (12)	50.20	72.64
GEN3C [67]	81.54	66.59	78.93 (2)	83.63	71.35	85.10	84.21	66.67	80.25 (2)	83.96	71.20	86.43
Lyra 2.0† [71]	91.40	71.20	81.27 (1)	78.27	74.48	89.15	91.00	71.37	81.01 (1)	79.89 (1)	74.08	89.29
Lyra 2.0 (DMD)† [71]	91.54	71.28	82.04 (1)	84.26	74.80	89.76	91.44	71.33	81.96 (1)	84.26	74.96	89.66
WorldDirector† [82]	76.52	66.74	77.14 (2)	71.71 (2)	69.08	81.99	76.15	66.97	78.80 (1)	68.72 (3)	67.26	81.95
Code World Model [13]	81.30	71.71	81.79 (1)	77.38 (2)	72.76	88.28	82.20	71.67	79.84 (2)	75.82 (2)	73.12	88.21
Code Video Model§ [44]	64.63	73.07	72.32 (5)	67.35 (5)	73.71	84.10	65.19	73.27	71.24 (6)	69.30 (4)	73.94	84.49
AgentGarten† [15]	87.78	73.17	86.82 (1)	87.07	74.15	87.74	88.34	73.57	86.94 (1)	89.00	74.60	88.49
M2R	94.96	72.01	88.34 (1)	88.68	76.11	90.91	95.38	72.44	88.44 (1)	89.35	76.36	91.28

Table A9. **All 60 ScanNet++ trajectories.** We report Orbia scores [54] on all 60 trajectories, including the 14 in which the camera passes through a wall, scored as in Tab. 1. Parentheses and markers are as in Tab. A8.

Method	ScanNet++, all trajectories (60)					
	Control Align.	Visual Quality	Input Pres.	Content Persist.	3D Consist.	Temp. Stab.
SolarWM H3 [33]	82.90	71.67	71.03 (1)	80.27 (1)	75.24	87.25
Matrix-Game 3.5 [66]	58.42	68.97	71.55 (1)	48.96 (11)	49.37	74.14
GEN3C [67]	84.55	66.60	82.02 (2)	84.89	71.16	85.72
Lyra 2.0† [71]	88.48	71.55	81.20 (1)	77.22	74.41	88.33
Lyra 2.0 (DMD)† [71]	89.47	70.85	81.58 (1)	79.57	73.54	87.91
WorldDirector† [82]	77.70	67.06	79.74 (1)	71.07 (2)	68.25	82.06
Code World Model [13]	78.77	71.44	82.32 (1)	75.39 (2)	71.75	86.93
Code Video Model§ [44]	66.00	73.26	75.06 (5)	67.14 (5)	73.37	84.13
AgentGarten† [15]	84.41	72.12	87.36 (1)	85.70 (1)	71.86	86.10
M2R	92.71	71.33	88.81 (1)	86.87	73.61	88.95

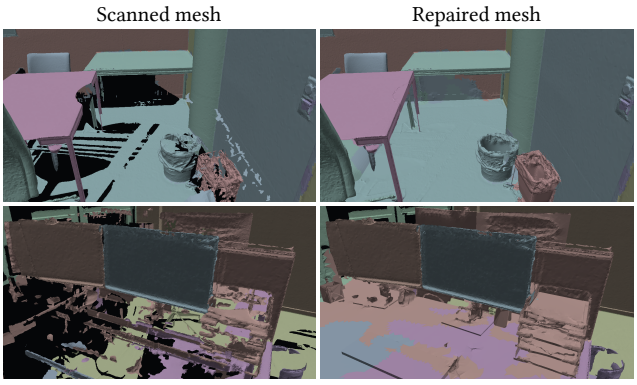

Figure A10. **Mesh repair.** We render two ScanNet++ scenes at the same pose from the scanned mesh (**left**) and from the mesh repaired by an LLM agent (**right**). The repair closes the holes in the floor and the furniture.

its first key frame. We provide the ground-truth mesh to every method that takes geometry, and Matrix-Game 3.5 [66], GEN3C [67], and Lyra 2.0 build their 3D caches from depth rendered from this mesh instead of the depth they estimate.

Metrics. We report the six scores of Orbia [54] with its released evaluator, each on a 0–100 scale. Control alignment measures the camera pose error and the accuracy of the move and look directions, and visual quality combines aesthetic, image-quality, flicker, smoothness, and preference scores. Input preservation compares later views of the surfaces visible in the input with ground truth, and content persistence (generated-content persistence in Orbia) compares each revisit of a region absent from the input with its first generated view. 3D consistency (3D self-consistency in Orbia) measures the agreement of depth, features, and scale across views, and how well a Gaussian scene predicted from the generated frames renders held-out views. Temporal stability tracks quality, control, and local geometry over 5 s windows. As its long-horizon part requires minute-long trajectories, which Orbia does not provide on ScanNet++, we report its short-horizon part.

G.2. Dynamic Environments

Datasets. We evaluate on five of the articulated HSSD homes [40] released with PARTNR [10], which we render with Habitat [77]. Their drawers and doors open and close, and their textured meshes provide ground truth. We also evaluate on MutableWorlds, eight worlds that we build with a coding agent (two alpine villages, a driving district, three harbors, a parcel depot, and a woodland campsite). As these worlds have no textures, we render their geometry with flat shading and turn the first frame of each trajectory into a realistic image with Gemini 3 Pro Image. This image is the only appearance that a method receives.

Table A10. **Trajectories in dynamic environments.** We list the number of trajectories of each type on HSSD and in MutableWorlds. As room-revisit trajectories also reopen drawers and doors, object permanence is scored on 24 trajectories on HSSD and 7 in MutableWorlds. Trajectories last 34 to 113 s on HSSD and 44 to 108 s in MutableWorlds, and every revisit comes at least 19 s after the first visit.

Type	HSSD	MutableWorlds	What happens
Reopen	13	1	The camera opens drawers or doors, leaves, and reopens them.
Room revisit	11	6	The camera also glances at the room and repeats the glances later.
Moved object	21	45	An object moves in view, moves unseen, or stays in place.
Total	45	52	

Table A11. **Rated questions.** We list the questions that Gemini 3.7 Flash answers with a rating from 0 to 10, given two crops of a method's own video side by side. Agent persistence (Sec. G.3) is zero unless its first question is rated at least 5.

Score	Question
Object permanence	Overall, would a viewer believe the second and later openings show the very same contents as the first?
Moved objects	Is the object at the centre of the right panel the very same physical object as the one at the centre of the left panel (same kind of object, shape, proportions, colours, materials and markings)?
Agent persistence	Is there a person (a human character; not an object, a box, a grey block or empty scenery) at the centre of both panels?
Agent persistence	Is the person at the centre of the right panel the very same character as the one at the centre of the left panel (same build, hair, clothing, colours and accessories)?

Trajectories. We render three types of single-camera trajectories in both datasets, in which the camera reopens drawers and doors, returns to rooms it has seen, or revisits objects that may have moved (Tab. A10). Every revisit replays the camera poses of the first visit, such that the ground-truth frames of the two visits differ only where the scene has changed. On HSSD, a revisit comes 19 to 94 s after the first visit. We render all trajectories at 16 FPS and 720×1280, with a camera that moves at up to 0.75 m/s and turns at up to $40°$/s.

Inputs. Every method receives the first frame and the camera path of a trajectory. All methods that take geometry also receive the same mesh as M2R, with its moving parts posed at every frame (or at each chunk update for 3D-cache methods), so they receive the updated scene geometry directly. We run every baseline with the settings of Sec. G.1.

Metrics. We measure memory by comparing the frames that a method generates at each revisit with its own frames at the same poses on the first visit, inside ground-truth masks. *Revisit* is the cosine similarity of DINOv2 patch features [64] over the static parts of a room view. *Object permanence* averages this similarity inside a reopened drawer or door with a rating by Gemini 3.7 Flash [25] of whether the contents are the same. For *moved objects*, we compare each revisited spot against two ground-truth renders at the same pose, one with the object and one without it, and compute the fraction of DINOv2 patches inside the object mask that are closer to the expected state. We average this score with a Gemini rating of whether it is the same object. Gemini rates each pair of crops three times from 0 to 10 with the questions of Tab. A11, and we scale all scores to 0–100. As a check, the ground-truth renders reach at least 98.8 on all three scores.

G.3. Multi-Agent Environments

Episodes. In MutableWorlds, we render 30 episodes with two agents and 23 with four, in which each agent later sees what another agent saw first. Each agent is followed by its own third-person camera, and the first frame of each view is made as for a single camera (Sec. G.2). In 22 of the two-agent episodes and all 23 four-agent episodes, each agent first inspects a place of its own, the agents meet, each visits the place of the next agent along the same camera path, they meet again at the same spots, and each returns to its own place. Seven of these two-agent episodes and two of the four-agent episodes also include a drawer that one agent opens and another reopens. The remaining eight two-agent episodes focus on moved objects across agents, where an object either moves unseen or stays in place while the two agents take turns inspecting the scene. The second look at a place comes 20 to 64 s after the first, and the two meetings are 24 to 54 s apart. Episodes last 63 to 128 s, and we render every view at 16 FPS and 720×1280.

Metrics. *Cross-agent consistency* measures whether a place looks the same to an agent as it did to the agent that saw it first. We compute the revisit, object permanence, and moved objects scores of Sec. G.2 with the first visit taken from the video of another agent and all agents masked, and average them (the first two with four agents, whose episodes have no moved object). *Agent persistence* measures whether an agent looks the same to another agent at both of their meetings. We average the DINOv2 similarity inside the box of that agent with a rating by Gemini of whether both meetings show the same character (Tab. A11). The score is zero unless both meetings show a person, as a method that draws no agent would otherwise be scored on the scenery behind the box.

Baselines. No baseline can share its memory between views. Each thus generates the view of every agent as a separate single-camera trajectory (Sec. G.2), whereas M2R generates all views of an episode together from one surface memory (Sec. D). Each baseline receives the agents in the form that its inputs take. AgentGarten, Code World Model, and Code Video Model receive a human-shaped figure in place of the box that M2R receives (Sec. C), WorldDirector receives a tracked box with a prompt that names a person, and the others receive only the prompt and the first frame.

H. Limitations

Although a coarse mesh suffices, M2R assumes that it is accurate, and errors in the mesh carry over into the generated video. It already tolerates the holes and artifacts of real scans, as repairing the ScanNet++ meshes raises its scores by at most 0.7 points (Tab. A8). Handling noisier geometry reconstructed from images would be a valuable next step. With multiple agents, M2R also requires the first frame of every view to be well aligned with the mesh. Frontier image models such as Gemini 3 Pro Image, which we use (Sec. G.2), often produce such frames, whereas obtaining them from older open-source models remains difficult. As the renderer receives each agent only as a box, it has to guess the appearance of a side of an agent that no view has shown. We currently keep the agent's appearance consistent across views with a few rules at inference (Sec. D), which explicit identity controls could replace. Finally, M2R renders non-human characters less reliably and less faithfully than humans. Only 34 of the 401 characters and less than 5% of the training videos are non-human (Sec. E), and we introduced them only in the final 2,000 iterations of the 513-frame bidirectional stage (Tab. A5) and the subsequent causal stages, exclusively in single-agent episodes. However, M2R already generalizes to many of them, which makes training on more such data a promising direction.

I. Societal Impact

M2R renders coarse, untextured meshes into photorealistic video in real time, which benefits applications such as gaming, embodied AI, and closed-loop simulation [2, 79]. Since an explicit world specifies geometry and state, a user can inspect and edit what happens in a generated video, and M2R determines only how it looks. However, as M2R renders realistic video of places and people from a single initial frame, it raises concerns about potential misuse for generating deceptive visual media (e.g., presenting a generated video as real footage). Appropriate safeguards such as watermarking generated content [86] should be considered for deployment. Finally, M2R builds on a pretrained video model [60] and is fine-tuned only on synthetic data [39], and thus inherits the biases of both.

J. Additional Qualitative Results

We show additional rollouts in static, dynamic, and multi-agent environments in Figs. A11 to A16.

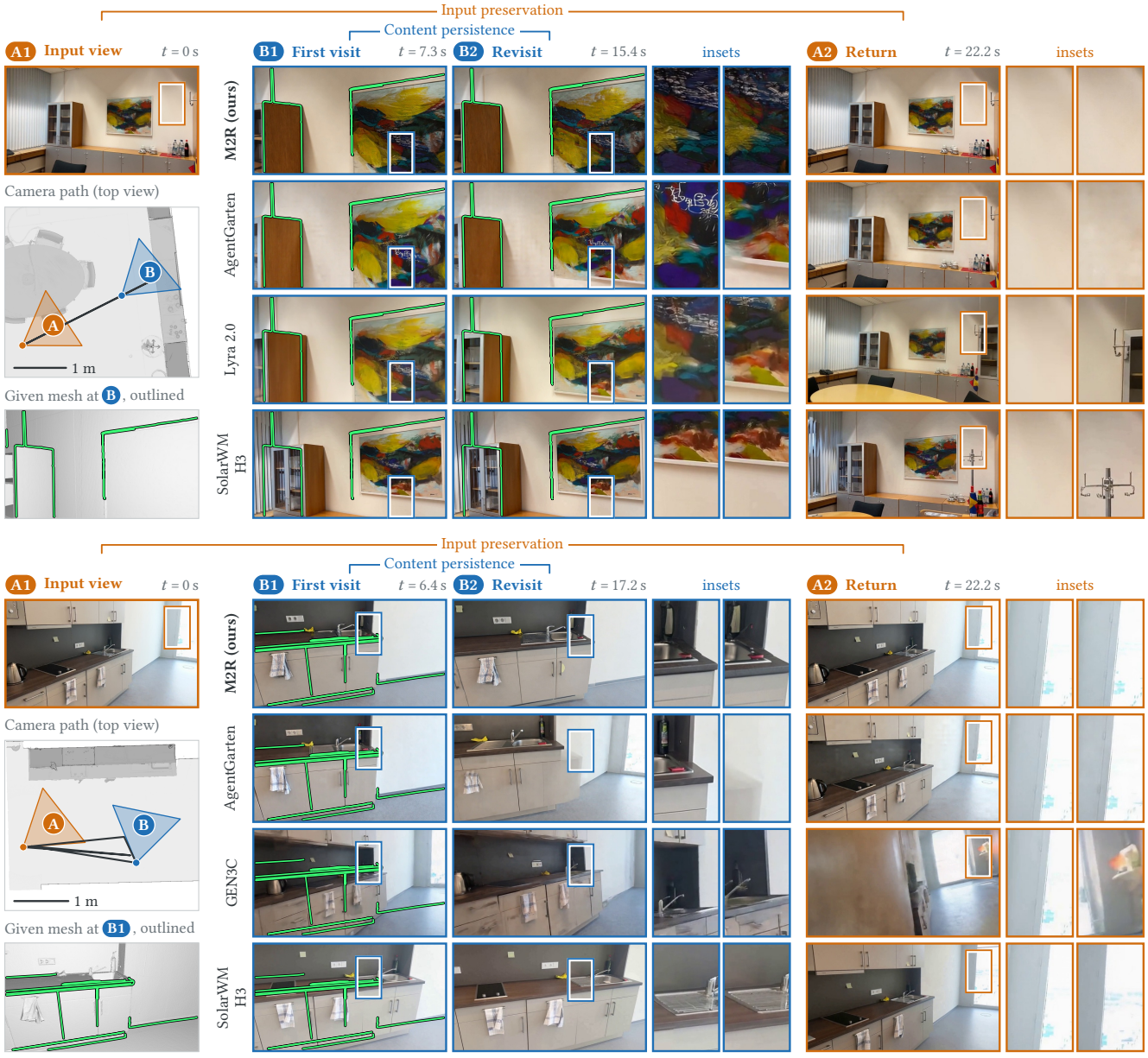

Figure A11. **ScanNet++.** We show two rollouts in the layout of Fig. 5, in an office with a painting (**top**) and in a kitchen that the camera revisits from a different angle (**bottom**). AgentGarten repaints the painting between the two visits (**top insets**), and SolarWM H3 stays far back from it. AgentGarten's revisit shows another part of the kitchen, and GEN3C does not return to the input view (**bottom**). M2R follows the camera and keeps both views in both rollouts.

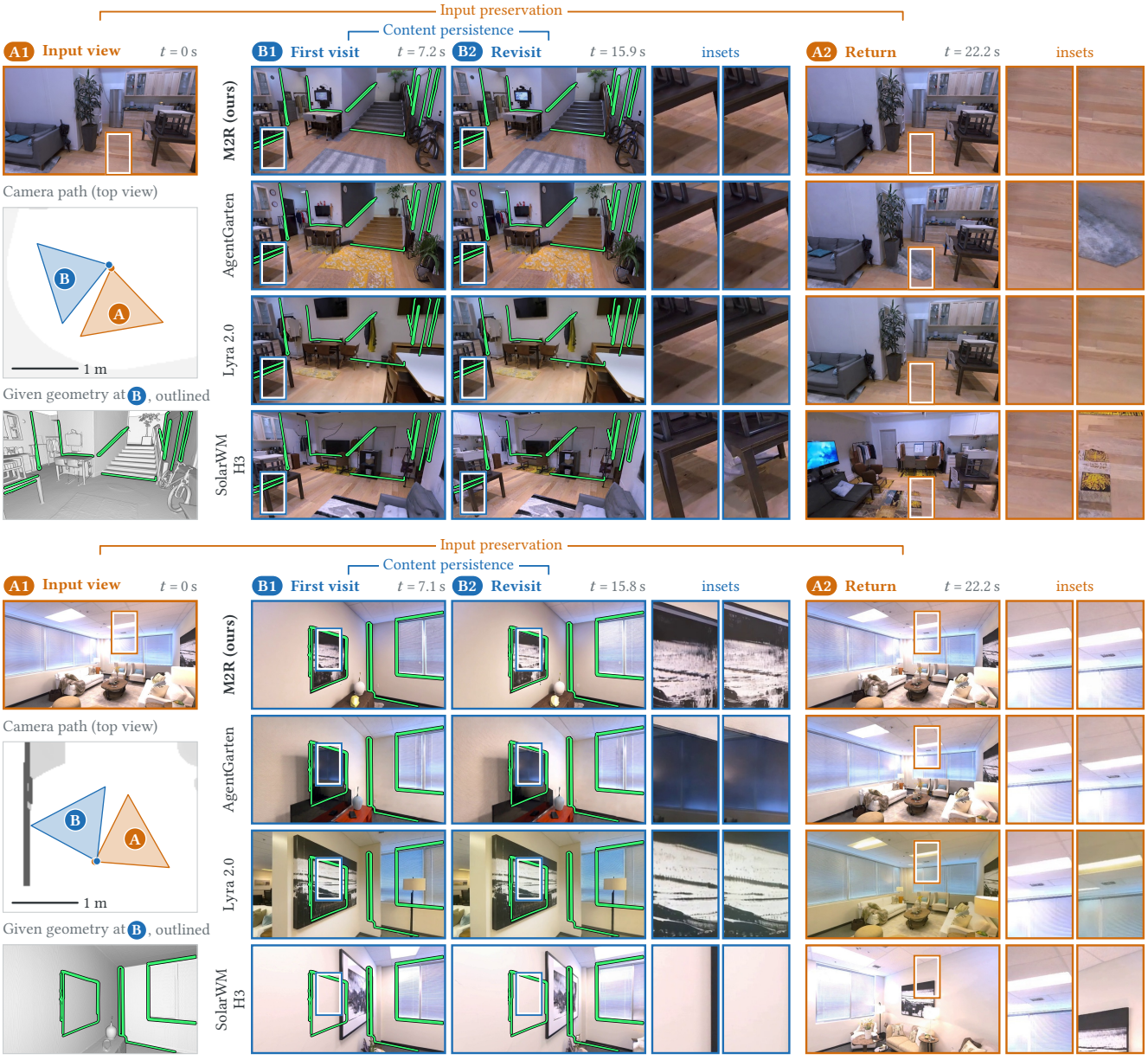

Figure A12. **Replica.** We show two rollouts in the layout of Fig. 5, in an apartment with stairs (**top**) and along a wall with a painting (**bottom**). On these short loops, most methods keep the revisited view, and they differ mainly in how they follow the given geometry and return to the input view. SolarWM H3 and Lyra 2.0 draw rooms that differ from the given geometry, and a rug appears in AgentGarten's return (**top insets**). AgentGarten draws a dark screen where the other methods draw a painting (**bottom**). M2R follows the geometry and keeps both views.

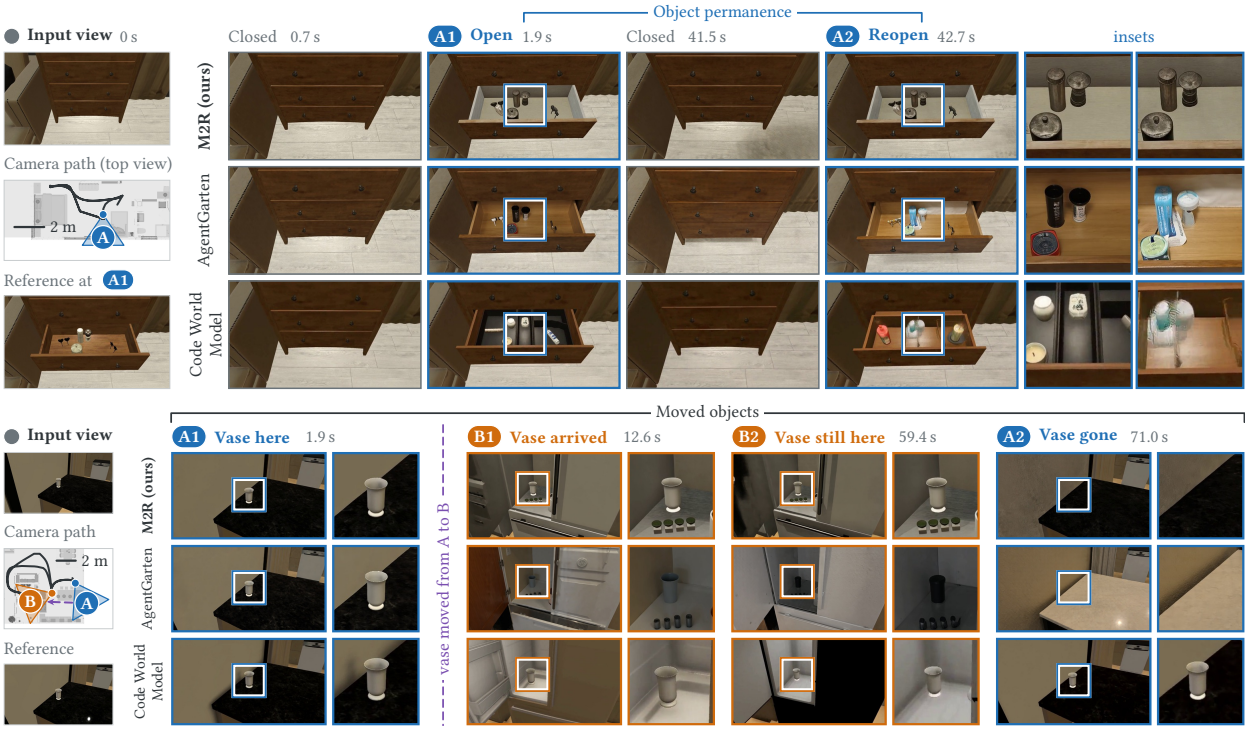

Figure A13. **HSSD.** We show a drawer that is reopened after 41 s (**top**) and a vase that is moved from a counter (A) into a fridge (B) while the camera looks away (**bottom**), in the layout of Fig. 6. Code World Model and AgentGarten change the contents of the drawer and keep the vase on the counter or draw a darker one, while M2R preserves both. The frame at the bottom left is the reference render at the first visit.

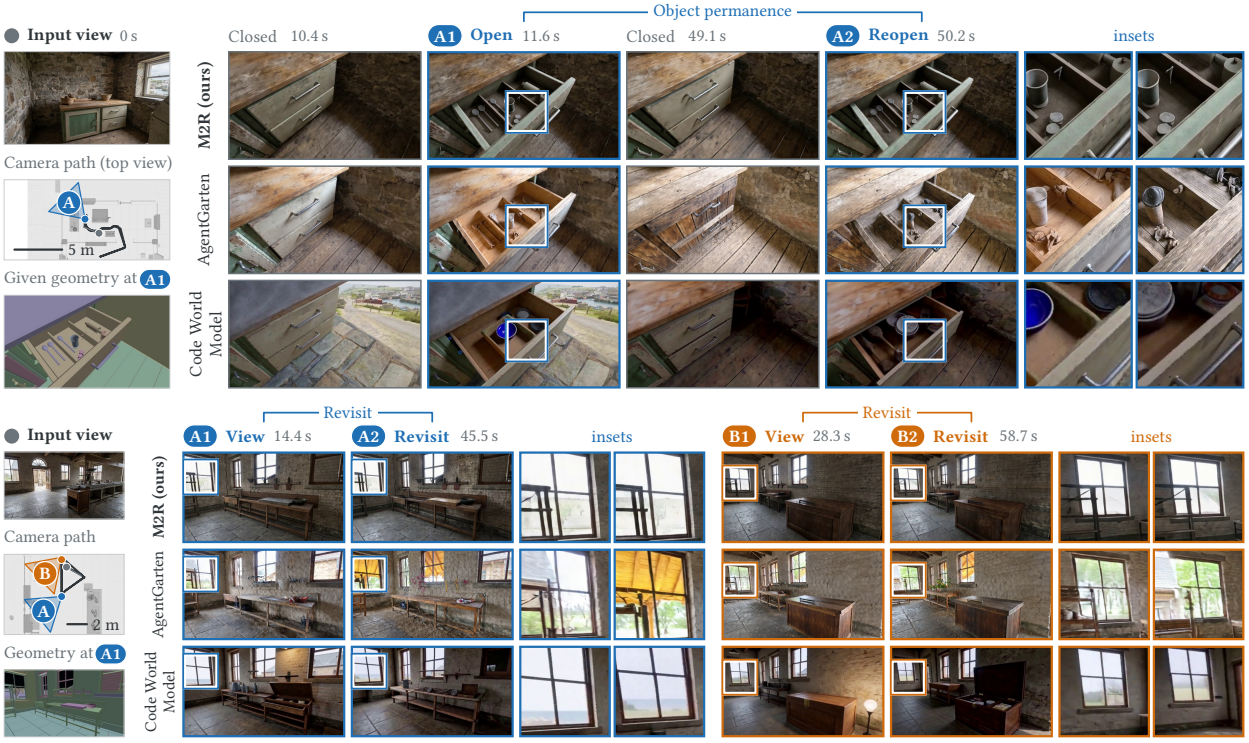

Figure A14. **MutableWorlds.** We show a bakery drawer that is reopened after 39 s (**top**) and a room that is revisited from two views (**bottom**). Code World Model moves from a quay into a room and changes the furniture, and AgentGarten changes the drawer, its contents, and the view through the window, while M2R preserves all of them.

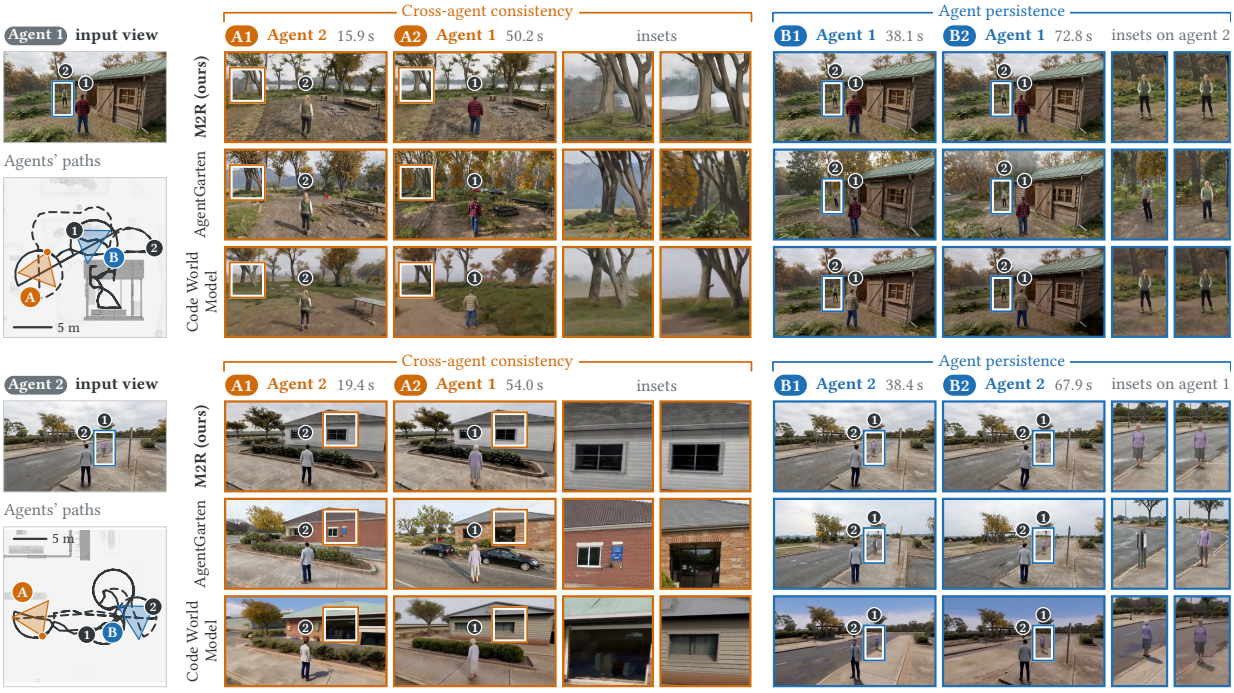

Figure A15. **Multiple agents.** We show two rollouts in the layout of Fig. 8, in a woodland campsite (**top**) and on a street (**bottom**), with an input view (the box marks the other agent) and both agents' paths (agent 2's dashed). AgentGarten and Code World Model show each agent a different place, and AgentGarten also draws another person or a signpost at the first meeting. Code World Model keeps the other agent, and M2R keeps both the place and the other agent.

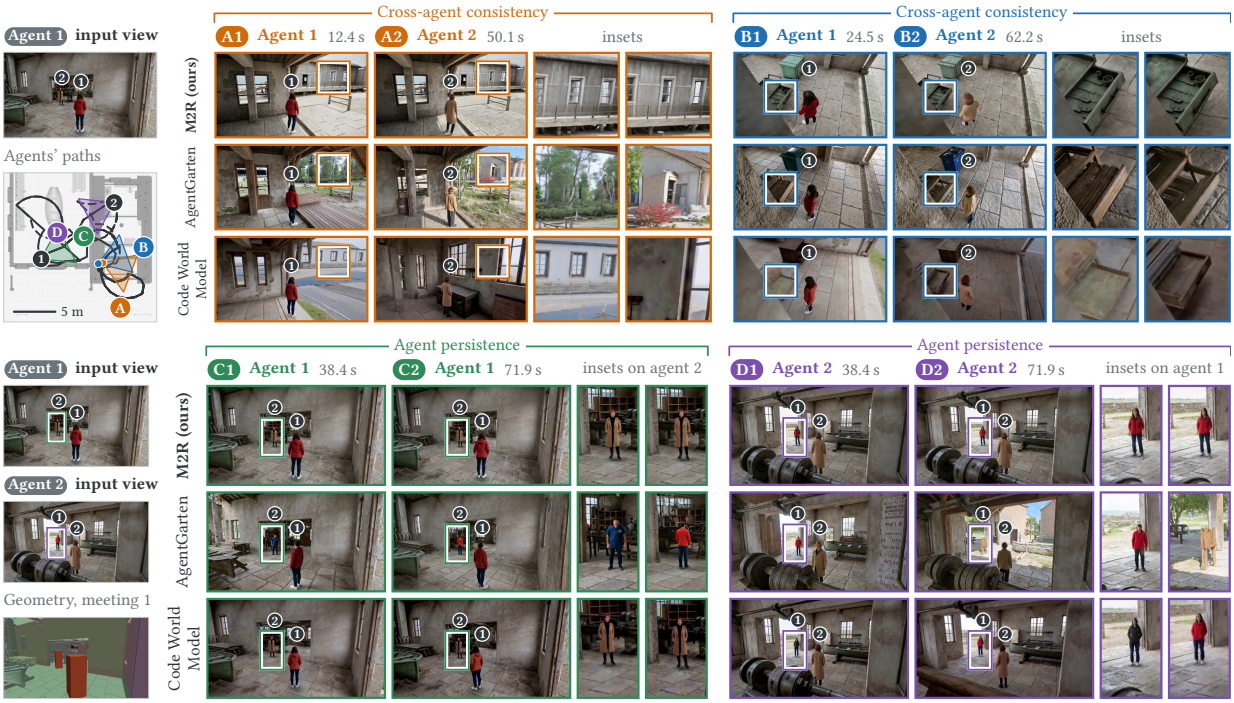

Figure A16. **Multiple agents with a dynamic object.** In a harbor workshop, we show cross-agent consistency for a place (**A**) and for a tool drawer that agent 1 opens and agent 2 reopens 38 s later (**B**), and agent persistence from both cameras at two meetings 34 s apart (**C**, **D**). AgentGarten and Code World Model show agent 2 another place, draw the drawer without its tools, and change or lose the other agent in at least one view, while M2R keeps the place, the tools, and both agents. The left column shows both input views, the agents' paths, and the given geometry at the first meeting, in which the agents are boxes.