HOLO4D: Holistic 4D Reconstruction as Geometric Control for Video Diffusion

Yushi Lan[1*] **Zeren Jiang**[1*] **Koichi Namekata**[1]
Xingang Pan[2] **Chuanxia Zheng**[2] **Andrea Vedaldi**[1]

[1]University of Oxford [2]Nanyang Technological University
https://nirvanalan.github.io/projects/holo4d

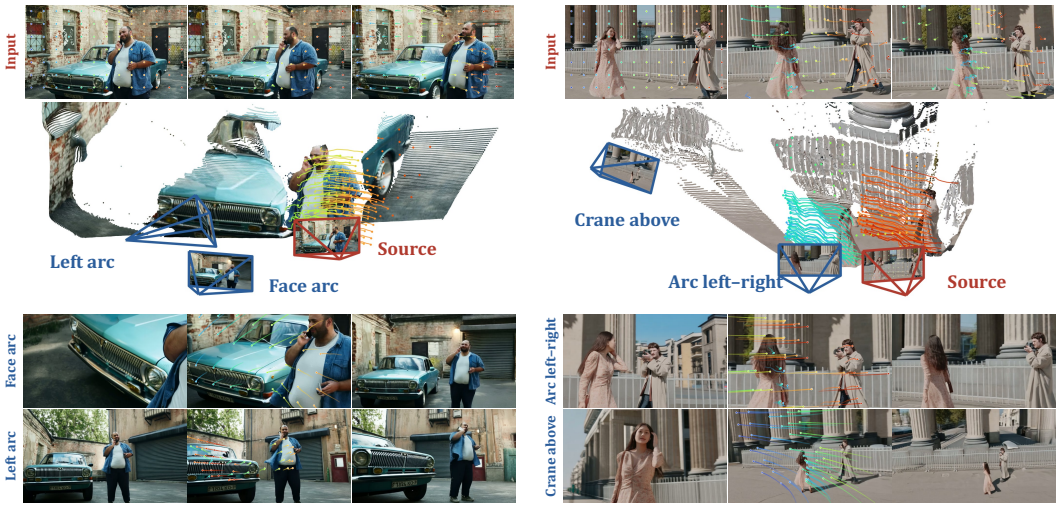

Figure 1: **4D reconstruction as a scaffold for video reshooting.** Top: input video with its 3D trajectories (projected onto the image plane for visualization). Middle: reconstruction with the source camera (red) and target cameras (blue) for camera paths from Vista4D's benchmark. Bottom: HOLO4D outputs for each path; the middle frame overlays the projected trajectories (subset shown).

Abstract

Synthesizing a monocular video from a new camera trajectory requires preserving the observed scene geometry and motion while completing regions that become newly visible. Feed-forward 4D reconstructors recover the structure of the observed scene, whereas video diffusion models provide a generative prior for completing unseen content. We introduce HOLO4D, a video reshooting framework that uses reconstructed 3D trajectories as a holistic source-view 4D scaffold. We project these trajectories into the requested views and encode their geometry and source appearance to guide a pretrained video diffusion model. Together with the source video and target cameras, the scaffold grounds generation in reconstructed observations while leaving unsupported regions to the generative prior. Experiments demonstrate the value of trajectory-based conditioning for camera control and novel-view synthesis. The framework accommodates different 4D reconstructors without retraining the video generator, allowing it to benefit from future advances in source reconstruction. More details can be found in our project page.

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Rendering a monocular video from a new camera trajectory would allow us to change the viewpoint of a recorded scene without changing the events it depicts [1]. For example, a camera moving around a walking person should reveal a different view of the same motion, rather than generate a different action or alter the surrounding scene. This requires both faithful reconstruction of observed content and plausible completion of surfaces that become visible only from the new viewpoint. The task therefore goes beyond recovering the geometry of the input frames: it must turn partial observations of a dynamic scene into a coherent video along a user-specified camera path.

Recent feed-forward reconstruction models provide an increasingly useful starting point. They estimate cameras, depth, and point maps directly from images and videos [2–5], while 4D reconstruction systems also recover the trajectories of scene points over time [6–9]. Together, these quantities describe the observed scene in a common spatial and temporal frame: camera motion can be separated from scene motion, and observations of a moving point can be linked across frames. However, reconstructing the observed scene does not by itself solve novel-view synthesis. Reprojecting its points into a new camera leaves holes where surfaces have never been observed [10], and inaccurate geometry can produce distortions as the viewpoint changes [11, 12]. The missing appearance must be inferred, rather than obtained by reprojection alone.

Video diffusion models offer the complementary ability to generate plausible appearance and motion from learned video priors [13]. Camera-controlled variants use these priors to synthesize new views, conditioned on camera embeddings or rendered source geometry [1, 14, 12]. Yet a realistic video need not preserve the geometry and dynamics of the particular scene being reconstructed. Combining reconstruction with generation is therefore not simply a matter of requesting a new camera pose: the generator also needs a representation of the scene that the camera is viewing. This motivates our central question: *how should the geometry, appearance, and motion recovered from a source video be exposed to a video diffusion model?*

An important part of this question is how the condition represents a scene over time. Lifting each frame independently with depth yields a sequence of point clouds, but does not explicitly specify which points correspond to the same moving surface in different frames. Vista4D [11] makes static scene content persistent by identifying static pixels and accumulating their observations in a common world frame. However, moving surfaces cannot be accumulated in the same way without accounting for their motion. Our starting point is that persistent 3D trajectories provide the required correspondence explicitly [6–8]. Specifically, a trajectory follows one scene point through time and associates its position with its observed appearance. A static point is the special case of a constant world-space trajectory, so static and moving points can use the same conditioning encoder.

We introduce HOLO4D, a video reshooting framework that converts reconstructed 3D trajectories and their source appearance into geometric features for video diffusion (Figure 2). We call this scaffold holistic because it represents observed geometry, appearance, and motion jointly for static and dynamic points. Projecting these trajectories into the target cameras specifies where observed scene content should appear and how it should move. We use these projected trajectories to guide the video diffusion model, alongside source-video conditioning [15] and the requested camera trajectory [16]. The tracks guide observed content throughout denoising, while the video prior supplies plausible appearance in newly revealed regions.

This conditioning design has two useful properties. First, we inject track features along the channel dimension [17], providing geometric guidance without an additional rendered-video token stream [11]. Second, the scaffold uses explicit geometric quantities rather than reconstructor-specific features. HOLO4D can therefore use any 4D reconstructor that provides the required geometry and trajectories [6, 8, 5, 18]. Our experiments show that replacing the upstream reconstruction can substantially improve camera adherence without retraining the video generator.

In summary, our contributions are threefold: (1) a holistic geometric scaffold that represents static and dynamic scene observations through persistent 3D trajectories with source appearance; (2) a conditioning design that combines source-video context with channel-wise injection of projected 3D trajectory features to guide novel-view generation; and (3) camera-adherence and held-out-view evaluations against strong baselines, analyses of reconstructor substitutions and trajectory perturbations, and demonstrations on robot videos.

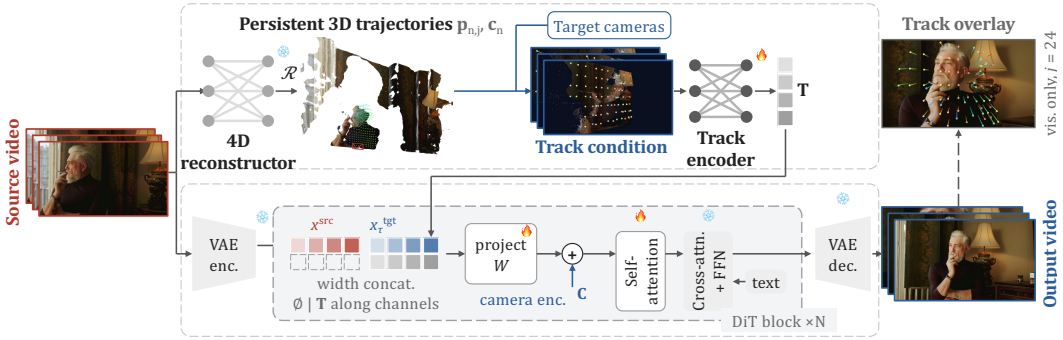

Figure 2: **HOLO4D overview.** Top (4D scaffold): a 4D reconstructor $\mathcal{R}$ recovers persistent 3D trajectories, paired with sampled source colors, which are projected into the target cameras and encoded into track features $\mathbf{T}$. Bottom (video diffusion): source and noisy target latents are concatenated along width; each DiT block concatenates $\mathbf{T}$ with its hidden features along channels (null on the source half) and adds camera features $\mathbf{C}$ encoded from Plücker rays. Flames and snowflakes mark trained and frozen modules (Appendix B). Trajectory panels show a subset of the dense tracks; the top-right overlay is for visualization only. The output is obtained by iterative denoising; Figure 11 details the DiT block.

2 Related Work

3D and 4D reconstruction. Dynamic scenes can be reconstructed by fitting deformable radiance fields [19–22] or time-varying Gaussian representations [23–25]. ViDAR [26] additionally uses a personalized diffusion prior to supervise per-scene Gaussian reconstruction. We instead condition a shared video generator on reconstructed geometry, without per-scene optimization.

Learned models recover point maps, cameras, and depth from images [27, 2, 4, 28, 29, 5] or accumulate observations over video [30, 3]. Video depth and dynamic-scene estimators provide geometry from moving-camera videos [31–37]. Recent 4D methods estimate dynamic point maps, motion, or persistent trajectories [38, 39, 8, 40, 41, 6, 7, 9], while 4DGT [42] predicts 4D Gaussians from posed videos. We use compatible geometric outputs as a scaffold for video reshooting.

Geometry-guided view synthesis. Geometry-guided rendering aggregates source-view information to synthesize novel views [43–46]. Ha et al. [46] also use temporal guidance and inpainting for multiview video. Our setting uses reconstructed 3D trajectories to condition video diffusion for reshooting a monocular video under large viewpoint changes.

Camera-controlled video generation. Camera-conditioned diffusion models encode target views through poses or rays [47–53], and have been adapted to video reshooting [1, 54]. Explicit conditions instead ground generation in rendered source geometry [14, 12, 55] or warped observations [56]. Related methods generate multiview videos before Gaussian reconstruction [57], or render Gaussian features for sparse-view synthesis [44].

Vista4D [11] conditions on source video and rendered point clouds with temporally persistent static pixels. LaVR [58] uses latent reconstruction states, whereas our interface is defined by explicit geometric quantities rather than reconstructor-specific hidden features. Go-with-the-Track [17] conditions generation on reference images and reference-anchored 2D tracks for compositing and motion control, and also supports camera retargeting. Edit-by-Track [59] edits camera and object motion in a video by conditioning on paired source and target 3D point tracks. For video reshooting, HOLO4D encodes 3D trajectory positions, source color, validity, and target-camera depth into geometric features, alongside the synchronized source video.

Diffusion models for geometry. Diffusion priors also support geometry prediction, from monocular depth [60, 61] to video geometry and motion [37, 15, 9]. Gen3R [62] jointly generates appearance and geometric latents. VIST3A [63] couples a video generator to a reconstruction decoder, while Plan4D [64] further generates plannable 4D worlds by stitching video generators and reconstruction decoders. HOLO4D instead takes reconstructed geometry as input and generates target-view RGB.

Video models for robots. Video-generation backbones increasingly serve robot learning, e.g., as world action models [65]; HOLO4D instead reshoots recorded robot videos from new viewpoints (Section 4.5).

3 Method

Given a source video $V^{\mathrm{src}} = (I_0^{\mathrm{src}}, \ldots, I_{N-1}^{\mathrm{src}})$ and a target camera trajectory, HOLO4D generates a video $V^{\mathrm{tgt}} = (I_0^{\mathrm{tgt}}, \ldots, I_{N-1}^{\mathrm{tgt}})$ that shows the same scene and motion from the requested cameras. Source and target frames with the same index refer to the same scene time. In Figure 2, the top branch reconstructs persistent 3D trajectories from the source video, forming a *4D scaffold*, projects them into the target cameras, and encodes them into target-aligned track features (Section 3.1). The bottom branch, a pretrained video diffusion model, generates the target video from the source video, the target cameras, and these features (Section 3.2).

3.1 Constructing a 4D scaffold

Reshooting a dynamic scene requires specifying where observed content should appear in each target frame and how its position changes over time. A video diffusion model conditioned only on the source video and target cameras [1] must infer this layout implicitly from appearance. Without an explicit scene representation, it struggles to preserve source-scene geometry and follow the requested camera under large viewpoint changes [58, 11]. To provide explicit geometric guidance, existing methods lift source frames into point clouds using estimated depth and render them from the target cameras [14, 12]. However, even when static points are accumulated across frames [11], independently reconstructed dynamic points do not explicitly establish correspondence between moving surfaces across frames. We therefore represent the source scene by persistent 3D trajectories [66], each carrying the appearance of a source observation.

To construct this scaffold, we use a 4D reconstructor $\mathcal{R}$ to recover camera poses and point trajectories from the source video [6–8, 5, 18]. Each trajectory follows a scene point n through its positions $\mathbf{p}_{n,j}$ at successive frames j and carries a color $\mathbf{c}_n$, sampled from the source frame q_n where the point was queried. We express these positions and the target cameras in the coordinate frame of the first source camera [27, 8], where static points have constant positions and moving points trace out their motion. Any reconstructor providing these trajectories and cameras can supply the scaffold.

To express the scaffold in the target views, we project each $\mathbf{p}_{n,j}$ with the target intrinsics K_j^{tgt} and pose π_j^{tgt} of frame j. This determines the point's target-view pixel location and its target-camera depth $d_{n,j}$, both computed from the reconstructed 3D position. A validity flag $v_{n,j}$ marks observations that are valid according to the upstream reconstructor and project inside the target image; it does not account for occlusion by other points. Each observation carries its frame index, position $\mathbf{p}_{n,j}$, depth $d_{n,j}$, validity $v_{n,j}$, and color $\mathbf{c}_n$. The color stays fixed along the trajectory, linking source appearance to the point's reconstructed motion.

Finally, we convert the projected observations into a feature volume aligned with the target video. A sinusoidal encoding [67] followed by a shared MLP maps the attributes of each observation to a feature vector, which is scattered to its projected location on a target-view grid. Features of observations that fall into the same cell are combined by a channel-wise maximum, and an occupancy channel records which cells received any observation. A causal 3D convolutional adapter then brings this volume to the DiT token grid, producing the track features $\mathbf{T}$; target tokens with no projected observations receive a learned null embedding. For a fixed target-video resolution and duration, the resulting feature volume has a size independent of the number of trajectories. Normalization, masking, and dataset-specific depth handling are given in Appendix B.

3.2 Conditioning video diffusion

The generator is a pretrained video diffusion transformer that combines source-video context, target-camera features, and the track features $\mathbf{T}$. The VAE encodes the source video into latents x^{src}, concatenated with the noisy target latents x_τ^{tgt} along the width [15]. This keeps synchronized source and target frames at the same temporal index, whereas temporal concatenation [11, 1] places them

at different indices. Learned domain embeddings distinguish the source and target halves. The text prompt enters through the backbone's pretrained cross-attention.

The track features enter the transformer along the channel dimension, following Go-with-the-Track [17], rather than as an additional sequence of tokens. We place them on the target half of the sequence and extend $\mathbf{T}$ on the source half with the same learned null embedding used for unoccupied target tokens. Channel-wise injection leaves the self-attention sequence length unchanged, and our model achieves a $1.34\times$ denoising speedup over Vista4D (Table 24).

The projected trajectories already convey the target viewpoint through their image locations. To provide auxiliary camera information in regions without valid projected observations, we also encode the target cameras as Plücker rays [16] into camera features $\mathbf{C}$. Source cameras establish the common reconstruction frame and are not separately embedded in the generator.

In the implementation, we inject the track and camera features before self-attention in every DiT block. Writing the hidden features of block ℓ after normalization and timestep modulation as $\mathbf{h}^{(\ell)}$, we compute the self-attention input as

$$\widehat{\mathbf{h}}^{(\ell)} = W^{(\ell)}[\mathbf{h}^{(\ell)}; \mathbf{T}] + \mathbf{C}, \tag{1}$$

where $[\cdot;\cdot]$ concatenates channels and $W^{(\ell)}$ is a learned projection back to the hidden width. Following the zero-initialized conditioning principle of ControlNet [68], the projection initially preserves the hidden features and assigns zero weight to the track channels. The complete block layout is shown in Figure 11, with implementation details in Appendix B.

3.3 Training and inference

A training example consists of a source video with 3D trajectory annotations, target cameras, and the target video seen from those cameras. The trajectory annotations are required to construct the conditioning features $\mathbf{T}$ and are obtained from an upstream reconstructor or, where available, dataset ground truth. We encode the target video into latents x^{tgt} and corrupt them with Gaussian noise ϵ as $x_\tau^{\mathrm{tgt}} = (1-\tau)\epsilon + \tau x^{\mathrm{tgt}}$, where $\tau \in [0,1]$ is the diffusion time, distinct from the video frame index; the source latents stay clean. The generator $u_{\boldsymbol{\theta}}$ is trained to predict the velocity of this interpolation with the rectified-flow objective [69]

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\left\| u_{\boldsymbol{\theta}}(x_\tau^{\mathrm{tgt}}, \tau; x^{\mathrm{src}}, \mathbf{T}, \mathbf{C}) - (x^{\mathrm{tgt}} - \epsilon) \right\|_2^2\right], \tag{2}$$

where the expectation is over training examples, noise, and diffusion time; auxiliary conditions are detailed in Appendix B. Only the target latents are supervised; no geometry is predicted.

At test time, we reconstruct the source video, build $\mathbf{T}$ and $\mathbf{C}$ for the requested cameras, and integrate the learned velocity from noise to target latents, which the VAE decodes into the output video. Videos longer than the training window are generated window by window (Section 4.5 and Appendix B).

3.4 Training details and datasets

We initialize from Wan2.1-T2V-14B [70] and train on 49-frame videos for 15k steps at 144×256, followed by 4.5k steps at 384×672. AdamW [71] uses learning rates 1×10^{-5} and 2×10^{-5} and effective batch sizes 8 and 32 for the two stages, respectively. We train conditioning modules and DiT self-attention, keeping pretrained cross-attention and feed-forward layers frozen [72].

Datasets. We train on a mixture of synthetic multiview videos of dynamic scenes [73, 74, 1], real videos of static scenes [75, 76], and synthetic static-scene sequences [77, 78]. For dynamic scenes, source–target pairs come from synchronized multiview videos. For static scenes, we form source–target pairs from different chunks of the same monocular video, similar to 4Real-Video-V2 [79]; temporal synchronization is unnecessary since the scene is static. This mixture covers both scene motion and camera motion, with estimated or ground-truth 3D trajectory annotations; dataset names, sampling weights, and annotation sources are given in Appendix C.

To expose the generator to imperfect and incomplete conditions, the second training stage varies trajectory density and query-frame sampling and applies mild geometric perturbations. Module dimensions and augmentation settings are specified in Appendix B.



Figure 3: **Video reshooting on Vista4D's benchmark trajectories.** Top: dense source geometry, source camera (red), and requested camera path (blue; circle: $j = 0$, frusta: $j = 24, 48$). The track condition projects source trajectories into the target view; dark regions have no support. The HOLO4D frame at $j = 24$ overlays a subset of the tracks. Red boxes (with magnified insets) mark Vista4D's errors: the diner is turned toward the camera, and the car loses its rear window. HOLO4D uses 50 steps with the two-stage backend (VGGT-Ω + CoWTracker; left) or 4RC (right); baseline inputs and released settings are described in Appendix E.

4 Experiments

We evaluate camera control and novel-view synthesis (Sections 4.2 and 4.3), analyze geometric conditioning (Section 4.4), and demonstrate robot-video applications (Section 4.5).

4.1 Experimental setup

Baselines. We compare HOLO4D with several representative state-of-the-art methods, including ReCamMaster [1], TrajectoryCrafter [14], EX-4D [55], Gen3C [12], and the concurrent Vista4D [11]. ReCamMaster is a *camera-only* baseline: it conditions on the source video and the target cameras. The others are *rendered-geometry* baselines: they also render source geometry from target cameras; Vista4D keeps static geometry persistent across frames.

Evaluation datasets. For camera control, we select 50 DAVIS [80] clips and randomly sample two target camera trajectories per clip, and also evaluate on Vista4D's released 110-trajectory benchmark.



Figure 4: **Novel-view synthesis on iPhone.** Rows show spin, block, and teddy at $j = 24$, with ground-truth target views (GT) on the right. HOLO4D uses the two-stage backend and 50 steps.

Five iPhone/DyCheck scenes [81] provide synchronized held-out views, while held-out Syn4D [73] scenes provide exact target RGB and geometry. For each metric, all methods are evaluated at the same resolution; detailed protocols are in Appendices D, F and I.

4D scaffold reconstruction. Our default source reconstructor $\mathcal{R}$ is 4RC [6]. We also test two-stage 4D reconstruction from 3D geometry and 2D tracking, as used for annotation in MolmoMotion [82], by combining VGGT-Ω [5] with CoWTracker [18]. For this two-stage backend, we optionally use GMOS [83] to hold background points fixed in 3D while preserving foreground motion.

Inference. For HOLO4D, we query points in the first source frame and use 50 denoising steps on DAVIS and iPhone; Syn4D settings are in Appendix I. Baselines use their released sampling settings.

4.2 Quantitative comparisons

Novel-view video synthesis. Table 1 compares generated views with synchronized iPhone recordings. PSNR, SSIM, and LPIPS score full images; masked ("m") variants score only co-visible pixels.

Table 1: **Novel-view synthesis on iPhone [81].** α: 4RC; β: two-stage backend; same generator, no GMOS. "m": masked metrics. Bold/underline: best/second best.

Method	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
ReCamMaster	10.34	0.478	0.628	10.56	0.326	0.689
TrajectoryCrafter	11.56	0.438	0.569	11.14	0.288	0.631
EX-4D	<u>13.31</u>	**0.520**	0.464	<u>12.91</u>	**0.366**	0.532
Gen3C	12.42	0.469	0.489	12.20	0.312	0.550
Vista4D	12.84	0.487	0.484	12.46	0.328	0.535
HOLO4D-14B$^{\alpha}$	13.22	0.484	<u>0.435</u>	12.68	0.326	<u>0.509</u>
HOLO4D-14B$^{\beta}$	**14.03**	<u>0.507</u>	**0.379**	**13.20**	<u>0.348</u>	**0.465**

With the two-stage backend, HOLO4D obtains the best PSNR (13.20 dB) and LPIPS in both forms, ahead of EX-4D and Vista4D; EX-4D leads in SSIM, and our 4RC variant is second in LPIPS. Both variants use the same generator without retraining, illustrating the effect of the source geometry and its preprocessing. Figure 4 shows visual comparisons; Table 18 compares with per-scene ViDAR.

Camera control accuracy. Table 2 compares camera control on 100 DAVIS video–trajectory pairs using VGGT-Ω camera recovery [5]. ATE measures trajectory error; RPE-T and RPE-R measure adjacent-frame translation and rotation errors. Translation metrics use 54 pairs; rotation uses all 100. HOLO4D has the lowest trajectory and translation errors of all methods and lower rotation error than all rendered-geometry baselines. With both methods using 4RC source reconstruction on DAVIS, HOLO4D reduces ATE by 54% and RPE-T by 33% compared with the concurrent Vista4D. ReCamMaster attains a slightly lower rotation error (0.099° versus 0.102°) but the largest trajectory error of all methods, consistent with outputs that stay close to the source viewpoint.

Video quality. Table 2 reports five VBench [84] dimensions on the same DAVIS videos. HOLO4D scores highest on imaging quality and, among rendered-geometry baselines, on subject and background consistency; it also exceeds Vista4D on aesthetic quality, with a marginally lower temporal-style score. ReCamMaster's high consistency accompanies the largest trajectory error: consistency alone does not establish camera fidelity. FVD computed against source videos can reward copying the input despite failed camera control. We therefore report FVD only on Syn4D, using ground-truth target-view videos as references; HOLO4D achieves the lowest FVD with ground-truth source geometry (Table 3).

Novel-view synthesis on synthetic scenes. Syn4D provides exact target views and source geometry, removing reconstruction error from the comparison. With ground-truth source depth and cameras for rendered-geometry baselines and 3D trajectories for HOLO4D, HOLO4D leads on all camera and

Table 2: **Camera control and video quality on DAVIS.** HOLO4D outperforms rendered-geometry baselines in camera control, imaging quality, and subject and background consistency. ATE: $\times 10^2$; RPE-T: $\times 10^3$; RPE-R: degrees. Bold and underline mark **best** and second-best results.

		Camera control			VBench [84]				
Method	**Control**	**ATE** ↓	**RPE-T** ↓	**RPE-R** ↓	**Aest.** ↑	**Imag.** ↑	**Subj.** ↑	**Backg.** ↑	**Temp.** ↑
ReCamMaster	Extrinsics	9.547	15.37	**0.099**	0.493	0.503	**0.922**	**0.942**	**0.256**
TrajectoryCrafter	Point cloud	3.533	4.817	0.207	**0.500**	0.548	0.894	0.923	0.253
EX-4D	Point cloud	2.903	3.560	0.587	0.442	0.377	0.827	0.899	0.238
Gen3C	Point cloud	5.168	7.581	0.386	0.454	0.450	0.856	0.911	0.248
Vista4D	Point cloud	3.576	3.608	0.132	0.478	0.534	0.904	0.926	0.256
HOLO4D-14B	3D tracks	**1.657**	**2.415**	0.102	0.493	**0.601**	0.910	0.931	0.253

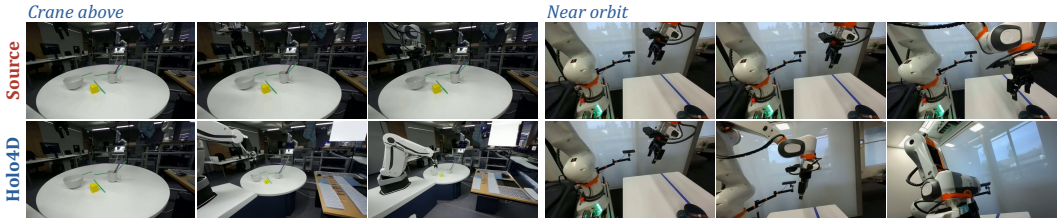

Figure 5: **Zero-shot robot reshooting.** Two DROID [85] episodes (top: source) reshot along a crane-above path and a near orbit (bottom), at $j = 0, 24, 48$. HOLO4D-14B uses no robot training data; inference uses 4RC and 50 steps. More examples and track conditions are in Figure 17.

image metrics (Table 3). With 4RC estimates on a separate 32-pair set, HOLO4D leads on camera errors and LPIPS, while camera-only ReCamMaster leads on PSNR and SSIM (Table 19). Test scenes are held out, but Syn4D is in our training mixture and the baseline checkpoints are not adapted to it; real-video evaluations test beyond our training datasets.

Depth consistency of generated views. We reconstruct generated-video depth and compare it with exact target depth on Syn4D (Table 20). HOLO4D has the lowest AbsRel and highest δ_1 under both output reconstructors, VGGT-Ω and 4RC; VGGT-Ω AbsRel falls from Vista4D's 0.628 to 0.423. On iPhone, with reconstructed target-video depth as a pseudo-reference, our two-stage variant also has the lowest AbsRel under both reconstructors, supporting improved generated-view geometry.

Comparison with per-scene optimization. ViDAR [26] takes about 5 h 18 min per iPhone scene on an A6000 for optimization; HOLO4D needs no scene-specific optimization. Given the same first 49 source frames, our two-stage variant has higher PSNR and lower LPIPS, including masked variants, while ViDAR has higher SSIM (Table 18); ViDAR natively uses the full sequence.

Sampling cost. Because the track condition enters along channels, HOLO4D adds no tokens to self-attention, whereas Vista4D appends a rendered-video token sequence. In a matched comparison at $384 \times 672 \times 49$ with 50 denoising steps on one H200, HOLO4D achieves a $1.34\times$ denoising speedup (7.39 versus 9.89 s) at similar peak memory (Table 24); end-to-end time per video is nonetheless similar (570.9 versus 581.9 s), since loading, conditioning, decoding, and writing offset the gain. Source reconstruction is paid once per input video and reused across target trajectories (8.8 s for 4RC and 11.8 s for the two-stage backend in warm A6000 runs, Table 23); only projection and encoding are redone per trajectory.

4.3 Qualitative comparisons

Figure 3 shows a restaurant side dolly and a car-following shot from Vista4D's benchmark. ReCamMaster stays close to the source framing; TrajectoryCrafter and Gen3C change or duplicate people and, like EX-4D, lose the car. EX-4D also leaves large holes. Vista4D turns the foreground diner toward the new camera and omits the car's rear window (red boxes and insets). HOLO4D preserves the diner's orientation, rear window, and lights while following the requested path; Tables 2 and 6 quantify camera accuracy, and Figure 16 shows two further trajectories. On iPhone (Figure 4), HOLO4D preserves the person and scene objects at the target viewpoint.

Table 3: **Novel-view synthesis on Syn4D.** Ground-truth source geometry at inference removes upstream reconstruction error. ATE: $\times 10^2$; RPE-T: $\times 10^3$; RPE-R: degrees. Bold and underline mark **best** and second-best results.

Method	ATE ↓	RPE-T ↓	RPE-R ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FVD ↓
ReCamMaster	12.68	62.20	0.749	13.66	0.118	0.236	1849
TrajectoryCrafter	5.018	20.61	0.225	13.64	0.256	0.178	1878
EX-4D	8.163	57.89	0.563	12.69	0.273	0.271	1916
Gen3C	5.724	40.03	0.469	13.10	0.103	0.211	1633
Vista4D	5.953	25.19	0.255	14.28	0.285	0.151	1214
HOLO4D-14B	**1.065**	**6.154**	**0.073**	**16.52**	**0.441**	**0.119**	**973**

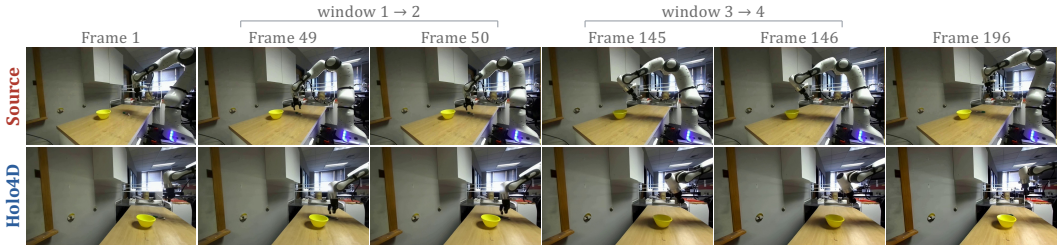

Figure 6: **Long-video rollout.** A 196-frame DROID [85] episode reshot to a static target view in five 49-frame windows (4RC, 50 steps): HOLO4D-14B for window 1, the I2V continuation model for windows 2–5 (Appendix B). Brackets mark the window boundaries shown.

4.4 Analysis of the geometric scaffold

Conditioning design. Table 4 compares conditioning variants trained with the same stage-1 recipe (15k steps at 144×256). Removing the per-track color raises ATE by 44% even though the source video remains available, so track-attached appearance is not redundant with the video. Removing target depth has a smaller effect, likely because depth follows from the 3D positions and target cameras. Adding rendered RGB and mask features to the track features improves none of the camera metrics, favoring trajectory-only conditioning for camera control at this training budget.

Table 4: **Stage-1 conditioning design on DAVIS.** All variants condition on trajectories. ATE: $\times 10^2$; RPE-T: $\times 10^3$; RPE-R: degrees.

Condition	ATE ↓	RPE-T ↓	RPE-R ↓
w/o source color	2.327	2.125	0.094
w/o target depth	1.882	2.135	0.068
+ rendered RGB/masks	1.83	2.48	0.098
Full (HOLO4D)	**1.621**	**2.077**	**0.065**

Reconstruction backends. The same generator accepts 4RC, V-DPM, and VGGT-Ω paired with Open-D4RT or CoWTracker as $\mathcal{R}$ without retraining (Table 5), and the backends trade off differently: the CoWTracker variant gives the best ATE, PSNR, and SSIM, V-DPM the best rotation error, and 4RC the best LPIPS. Ground-truth geometry improves all five metrics, leaving room for better reconstructors.

Table 5: **Reconstruction backends and track noise on Syn4D.** Top: backends (*VGGT-Ω depth and cameras); bottom: GT tracks with relative noise α. ATE $\times 10^2$; RPE-R in degrees. No GMOS. Bold/underline: best/second best per block.

Source geometry	ATE ↓	RPE-R ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Open-D4RT*	1.858	0.151	13.10	0.158	0.169
V-DPM	1.660	**0.079**	14.96	0.268	0.151
4RC	1.353	0.087	14.98	0.278	**0.131**
CoWTracker*	**1.205**	0.107	**15.09**	**0.294**	0.137
GT geometry	**1.065**	**0.073**	**16.52**	**0.441**	**0.119**
$\alpha = 0.1$	3.033	0.221	13.44	0.171	0.172
$\alpha = 0.5$	2.568	0.540	13.49	0.111	0.238
$\alpha = 1.0$	2.228	0.574	13.37	0.100	0.241
$\alpha = 4.0$	2.163	0.604	13.26	0.093	0.242

Trajectory accuracy. To isolate the effect of geometric error, we add Gaussian noise to ground-truth tracks, with standard deviation α times that of the source tracks along each axis, keeping all other inputs fixed. Even mild noise ($\alpha = 0.1$) triples rotation error ($0.073°$ to $0.221°$) and degrades SSIM and LPIPS although the clean source video remains available, showing that clean video context does not fully compensate for geometric error. ATE does not grow beyond $\alpha = 0.1$; independent alignment of each recovered camera trajectory can absorb part of the error. Full results and protocols are in Appendices G and I.

Rotation control with accurate geometry. With ground-truth source geometry on nine held-out Syn4D clips, HOLO4D has lower error on all five camera metrics than Vista4D, including source-

Table 7: **Policy learning with Fast-WAM.** Success (%) after 20k LIBERO steps. Control and HOLO4D differ only in video self-attention initialization; the official release is a separate reference. Bold/underline: best/second best.

	Overall		LIBERO-plus perturbation category						
Initialization	**LIBERO**	**LIB.-plus**	**Background**	**Camera**	**Language**	**Light**	**Layout**	**Robot**	**Noise**
Official release	**96.80**	<u>50.91</u>	50.09	**16.82**	69.94	81.79	61.97	46.00	**39.41**
Control (Wan2.2)	95.60	50.81	<u>56.32</u>	<u>9.01</u>	<u>70.79</u>	<u>85.20</u>	**66.62**	<u>49.10</u>	31.73
HOLO4D init.	<u>96.45</u>	**52.10**	**63.38**	8.01	**71.57**	**89.14**	<u>66.16</u>	**49.16**	<u>32.92</u>

aligned rotation ($0.242°$ versus $0.372°$; Table 14). Replacing GT with 4RC estimates on the same clips and checkpoint raises this error to $0.931°$ but barely changes relative rotation error ($0.051°$ to $0.055°$), consistent with an orientation offset despite similar local motion. Accurate relative motion therefore does not guarantee the requested viewpoint.

Camera control on Vista4D trajectories. On Vista4D's 110 trajectories (Table 6), Vista4D uses its released Pi3 geometry and reprojections, while HOLO4D uses 4RC or the two-stage backend (protocols in Appendix E). With scalar-depth-aligned 4RC geometry, HOLO4D has similar ATE, lower RPE-T and Pi3 translation error, but higher rotation errors than Vista4D. Substituting the two-stage backend cuts rotation error by 74% under VGGT-Ω and 59% under Pi3 ($14.98°$ to $6.098°$), improves ATE and RPE-T, and slightly raises Pi3 translation error. Pi3 rotation remains above Vista4D's $4.169°$. The comparison with Vista4D does not isolate the generator.

Table 6: **Camera control on Vista4D's 110 trajectories.** α: 4RC; β: two-stage. ATE $\times 10^2$; RPE-T $\times 10^3$; rotations in degrees.

Method	**VGGT-Ω**			**Pi3**	
	ATE ↓	**RPE-T ↓**	**RPE-R ↓**	**TransErr^2 ↓**	**RotErr ↓**
Vista4D	<u>46.04</u>	36.81	**0.255**	4.573	**4.169**
HOLO4D-14B$^\alpha$	46.31	<u>21.33</u>	1.142	**2.164**	14.98
HOLO4D-14B$^\beta$	**13.76**	**9.064**	<u>0.301</u>	<u>2.367</u>	<u>6.098</u>

4.5 Robot videos

Zero-shot reshooting. HOLO4D-14B is trained without robot videos, yet reshoots DROID manipulation episodes and humanoid-robot clips without fine-tuning (Figures 5 and 17).

Long-video rollout. We generate overlapping 49-frame windows: the first with HOLO4D-14B, then with an image-to-video continuation model conditioned on the last generated frame (Appendix B). The continuation model trains on OmniWorld-DROID, so later windows are not zero-shot. Figure 6 qualitatively shows consistent appearance across two window boundaries in a 196-frame episode.

Transfer to robot policy learning. We initialize Fast-WAM's [65] 30 video self-attention blocks from a 5B HOLO4D variant (Appendix L) and train on LIBERO [86] alongside a matched Wan2.2-initialized control (Table 7). Results are mixed: LIBERO has few tasks and is near saturation (95.60% vs. 96.45%), leaving little room to distinguish the two. On LIBERO-plus [87], HOLO4D gains 1.29 percentage points overall, mainly under background-texture (+7.1) and lighting (+3.9) perturbations, but loses on the spatial suite. Camera-viewpoint perturbations remain hard for all checkpoints (single seed). Whether novel-view training helps embodied policies remains open and merits further study.

5 Conclusion

HOLO4D guides video diffusion with persistent 3D trajectories that carry source appearance, leaving regions beyond the scaffold to the generative prior. It achieves the lowest trajectory and translation errors on DAVIS and the best PSNR and LPIPS on iPhone with the two-stage backend, and works with different reconstructors without retraining; it remains sensitive to geometric error, incomplete source coverage, and difficult camera trajectories.

Future work. Since video generators are general-purpose vision learners [88], one model could jointly perform 4D reconstruction and novel-view synthesis so that both improve together. Querying trajectories in every source frame, not just the first (e.g., UniQueR [89]), would cover later-appearing content; stronger base models such as MiniMax H3 [90] should improve visual quality.

Acknowledgments and Disclosure of Funding

This research was supported by the Sony Research Award Programme 2024 and ERC CoG UNION (101001212). We thank Isambard-AI and Dawn AIRR supercomputers for supporting this project. The authors declare no competing interests.

References

[1] Jianhong Bai, Menghan Xia, Xiao Fu, Xintao Wang, Lianrui Mu, Jinwen Cao, Zuozhu Liu, Haoji Hu, Xiang Bai, Pengfei Wan, and Di Zhang. ReCamMaster: camera-controlled generative rendering from a single video. In *ICCV*, 2025.

[2] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *CVPR*, 2025.

[3] Yushi Lan, Yihang Luo, Fangzhou Hong, Shangchen Zhou, Honghua Chen, Zhaoyang Lyu, Shuai Yang, Bo Dai, Chen Change Loy, and Xingang Pan. STream3R: Scalable sequential 3D reconstruction with causal transformer. In *ICLR*, 2026.

[4] Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. In *ICLR*, 2026.

[5] Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *CVPR*, 2026.

[6] Yihang Luo, Shangchen Zhou, Yushi Lan, Xingang Pan, and Chen Change Loy. 4RC: 4D reconstruction via conditional querying anytime and anywhere. In *ICML*, 2026.

[7] Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K. Barral, Raia Hadsell, Zoubin Ghahramani, Andrew Zisserman, Junlin Zhang, and Mehdi S. M. Sajjadi. Efficiently reconstructing dynamic scenes one D4RT at a time. In *CVPR*, 2026.

[8] Edgar Sucar, Eldar Insafutdinov, Zihang Lai, and Andrea Vedaldi. V-DPM: Video reconstruction with dynamic point maps. In *CVPR*, 2026.

[9] Ruijie Zhu, Jiahao Lu, Wenbo Hu, Xiaoguang Han, Jianfei Cai, Ying Shan, and Chuanxia Zheng. MotionCrafter: Dense geometry and motion reconstruction with a 4D VAE. In *CVPR*, 2026.

[10] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *CVPR*, 2020.

[11] Kuan Heng Lin, Zhizheng Liu, Pablo Salamanca, Yash Kant, Ryan Burgert, Koichi Namekata, Yiwei Zhao, Bolei Zhou, Micah Goldblum, Paul Debevec, and Ning Yu. Vista4D: Video reshooting with 4D point clouds. In *CVPR*, 2026.

[12] 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.

[13] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. Technical report, OpenAI, 2024.

[14] Mark Yu, Wenbo Hu, Jinbo Xing, and Ying Shan. TrajectoryCrafter: redirecting camera trajectory for monocular videos via diffusion models. In *ICCV*, 2025.

[15] Zhaoxi Chen, Tianqi Liu, Long Zhuo, Jiawei Ren, Zeng Tao, He Zhu, Fangzhou Hong, Liang Pan, and Ziwei Liu. 4DNeX: feed-forward 4D generative modeling made easy. *arXiv*, 2508.13154, 2025.

[16] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *NeurIPS*, 2021.

[17] Koichi Namekata, Yash Kant, Zhizheng Liu, Ryan D. Burgert, Yuancheng Xu, Kuan Heng Lin, Emmett Steven, Julien Philip, Li Ma, Andrea Vedaldi, Paul Debevec, and Ning Yu. Go-with-the-track: Video compositing and motion control with point tracking. In *SIGGRAPH*, 2026.

[18] Zihang Lai, Eldar Insafutdinov, Edgar Sucar, and Andrea Vedaldi. CoWTracker: Tracking by warping instead of correlation. In *CVPR*, 2026.

[19] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021.

[20] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields. *ACM TOG*, 40(6), 2021.

[21] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, 2021.

[22] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. DynIBaR: Neural dynamic image-based rendering. In *CVPR*, 2023.

[23] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024.

[24] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian splatting for real-time dynamic scene rendering. In *CVPR*, 2024.

[25] Qianqian Wang, Vickie Ye, Hang Gao, Weijia Zeng, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4D reconstruction from a single video. In *ICCV*, 2025.

[26] Michal Nazarczuk, Sibi Catley-Chandar, Thomas Tanay, Zhensong Zhang, Gregory Slabaugh, and Eduardo Pérez-Pellitero. ViDAR: Video Diffusion-Aware 4D Reconstruction From Monocular Inputs. In *NeurIPS*, 2025.

[27] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, 2024.

[28] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *ICLR*, 2026.

[29] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: universal feed-forward metric 3D reconstruction. In *3DV*, 2026.

[30] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3D perception model with persistent state. In *CVPR*, 2025.

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

[32] Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. DepthCrafter: generating consistent long depth sequences for open-world videos. In *CVPR*, 2025.

[33] Tian-Xing Xu, Xiangjun Gao, Wenbo Hu, Xiaoyu Li, Song-Hai Zhang, and Ying Shan. GeometryCrafter: consistent geometry estimation for open-world videos with diffusion priors. In *ICCV*, 2025.

[34] Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. MegaSaM: accurate, fast and robust structure and motion from casual dynamic videos. In *CVPR*, 2025.

[35] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. MonST3R: a simple approach for estimating geometry in the presence of motion. In *ICLR*, 2025.

[36] David Yifan Yao, Albert J. Zhai, and Shenlong Wang. Uni4D: Unifying visual foundation models for 4D modeling from a single video. In *CVPR*, 2025.

[37] Zeren Jiang, Chuanxia Zheng, Iro Laina, Diane Larlus, and Andrea Vedaldi. Geo4D: Leveraging video generators for geometric 4D scene reconstruction. In *ICCV*, 2025.

[38] Haiwen Feng, Junyi Zhang, Qianqian Wang, Yufei Ye, Pengcheng Yu, Michael J. Black, Trevor Darrell, and Angjoo Kanazawa. St4RTrack: simultaneous 4D reconstruction and tracking in the world. In *ICCV*, 2025.

[39] Edgar Sucar, Zihang Lai, Eldar Insafutdinov, and Andrea Vedaldi. Dynamic Point Maps: A versatile representation for dynamic 3D reconstruction. In *ICCV*, 2025.

[40] Xinhang Liu, Yuxi Xiao, Donny Y. Chen, Jiashi Feng, Yu-Wing Tai, Chi-Keung Tang, and Bingyi Kang. Trace anything: Representing any video in 4D via trajectory fields. In *ICLR*, 2026.

[41] Jay Karhade, Nikhil Keetha, Yuchen Zhang, Tanisha Gupta, Akash Sharma, Sebastian Scherer, and Deva Ramanan. Any4D: Unified feed-forward metric 4D reconstruction. In *CVPR*, 2026.

[42] Zhen Xu, Zhengqin Li, Zhao Dong, Xiaowei Zhou, Richard Newcombe, and Zhaoyang Lv. 4DGT: Learning a 4D Gaussian transformer using real-world monocular videos. In *NeurIPS*, 2025.

[43] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas A. Funkhouser. IBRNet: Learning multi-view image-based rendering. In *CVPR*, 2021.

[44] Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. MVSplat360: Feed-forward 360 scene synthesis from sparse views. In *NeurIPS*, 2024.

[45] Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, João F. Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3D: Feed-forward generalisable 3D scene reconstruction from a single image. In *3DV*, 2025.

[46] Hyunho Ha, Lei Xiao, Christian Richardt, Thu Nguyen-Phuoc, Changil Kim, Min H. Kim, Douglas Lanman, and Numair Khan. Geometry-guided online 3D video synthesis with multi-view temporal consistency. In *CVPR*, 2025.

[47] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3D object. In *ICCV*, 2023.

[48] Chuanxia Zheng and Andrea Vedaldi. Free3D: Consistent novel view synthesis without 3D representation. In *CVPR*, 2024.

[49] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitrii Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. SV3D: Novel multi-view synthesis and 3D generation from a single image using latent video diffusion. In *ECCV*, 2024.

[50] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. MotionCtrl: A unified and flexible motion controller for video generation. In *SIGGRAPH*, 2024.

[51] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. CameraCtrl: enabling camera control for text-to-video generation. In *ICLR*, 2025.

[52] Hao He, Ceyuan Yang, Shanchuan Lin, Yinghao Xu, Meng Wei, Liangke Gui, Qi Zhao, Gordon Wetzstein, Lu Jiang, and Hongsheng Li. CameraCtrl II: Dynamic scene exploration via camera-controlled video diffusion models. In *ICCV*, 2025.

[53] Sherwin Bahmani, Ivan Skorokhodov, Aliaksandr Siarohin, Willi Menapace, Guocheng Qian, Michael Vasilkovsky, Hsin-Ying Lee, Chaoyang Wang, Jiaxu Zou, Andrea Tagliasacchi, David B. Lindell, and Sergey Tulyakov. VD3D: taming large video diffusion transformers for 3D camera control. In *ICLR*, 2025.

[54] Basile Van Hoorick, Rundi Wu, Ege Ozguroglu, Kyle Sargent, Ruoshi Liu, Pavel Tokmakov, Achal Dave, Changxi Zheng, and Carl Vondrick. Generative camera dolly: Extreme monocular dynamic novel view synthesis. In *ECCV*, 2024.

[55] Tao Hu, Haoyang Peng, Xiao Liu, and Yuewen Ma. EX-4D: EXtreme viewpoint 4D video synthesis via depth watertight mesh. *arXiv*, 2506.05554, 2025.

[56] Meng You, Zhiyu Zhu, Hui Liu, and Junhui Hou. NVS-Solver: video diffusion model as zero-shot novel view synthesizer. In *ICLR*, 2025.

[57] Rundi Wu, Ruiqi Gao, Ben Poole, Alex Trevithick, Changxi Zheng, Jonathan T. Barron, and Aleksander Holynski. CAT4D: create anything in 4D with multi-view video diffusion models. In *CVPR*, 2025.

[58] Mingyang Xie, Numair Khan, Tianfu Wang, Naina Dhingra, Seonghyeon Nam, Haitao Yang, Zhuo Hui, Christopher Metzler, Andrea Vedaldi, Hamed Pirsiavash, and Lei Luo. LaVR: Scene latent conditioned generative video trajectory re-rendering using large 4D reconstruction models. In *CVPR*, 2026.

[59] Yao-Chih Lee, Zhoutong Zhang, Jiahui Huang, Jui-Hsien Wang, Joon-Young Lee, Jia-Bin Huang, Eli Shechtman, and Zhengqi Li. Generative video motion editing with 3d point tracks. In *CVPR*, pages 18306–18318, 2026.

[60] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *CVPR*, 2024.

[61] Gangwei Xu, Haotong Lin, Hongcheng Luo, Xianqi Wang, Jingfeng Yao, Lianghui Zhu, Yuechuan Pu, Cheng Chi, Haiyang Sun, Bing Wang, Guang Chen, Hangjun Ye, Sida Peng, and Xin Yang. Pixel-perfect depth with semantics-prompted diffusion transformers. In *NeurIPS*, 2025.

[62] Jiaxin Huang, Yuanbo Yang, Bangbang Yang, Lin Ma, Yuewen Ma, and Yiyi Liao. Gen3R: 3D scene generation meets feed-forward reconstruction. In *CVPR*, 2026.

[63] Hyojun Go, Dominik Narnhofer, Goutam Bhat, Prune Truong, Federico Tombari, and Konrad Schindler. Text-to-3D by stitching a multi-view reconstruction network to a video generator. In *ICLR*, 2026.

[64] Hangwei Zhang, Rujing Dang, Ruimin Lin, Jiahong Wu, Xiangxiang Chu, and Chuanxia Zheng. Plan4d: Generative plannable 4d worlds. In *NeurIPS*, 2026.

[65] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv*, 2603.16666, 2026.

[66] Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. SpatialTracker: tracking any 2D pixels in 3D space. In *CVPR*, 2024.

[67] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020.

[68] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.

[69] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[70] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming

Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv*, 2503.20314, 2025.

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

[72] Xiao Fu, Shitao Tang, Min Shi, Xian Liu, Jinwei Gu, Ming-Yu Liu, Dahua Lin, and Chen-Hsuan Lin. Plenoptic video generation. In *CVPR*, 2026.

[73] Zeren Jiang, Yushi Lan, Yihang Luo, Yufan Deng, Zihang Lai, Edgar Sucar, Christian Rupprecht, Iro Laina, Diane Larlus, Chuanxia Zheng, and Andrea Vedaldi. Syn4D: A multiview synthetic 4D dataset. In *ECCV*, 2026.

[74] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. In *CVPR*, 2022.

[75] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: a large-scale scene dataset for deep learning-based 3D vision. In *CVPR*, 2024.

[76] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM TOG*, 37, 2018.

[77] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *CVPR*, 2018.

[78] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. TartanAir: a dataset to push the limits of visual SLAM. In *IROS*, 2020.

[79] Chaoyang Wang, Ashkan Mirzaei, Vidit Goel, Willi Menapace, Aliaksandr Siarohin, Michael Vasilkovsky, Ivan Skorokhodov, Vladislav Shakhrai, Sergei Korolev, Sergey Tulyakov, and Peter Wonka. Fused view-time attention and feedforward reconstruction for 4D scene generation. In *NeurIPS*, 2026.

[80] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv*, 1704.00675, 2017.

[81] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, 2022.

[82] Jianing Zhang, Chenhao Zheng, Yajun Yang, Max Argus, Rustin Soraki, Winson Han, Taira Anderson, Chun-Liang Li, Shuo Liu, Jiafei Duan, Zhongzheng Ren, Jieyu Zhang, and Ranjay Krishna. MolmoMotion: Forecasting point trajectories in 3D with language instruction. *arXiv*, 2606.18558, 2026.

[83] Junyu Xie, Tengda Han, Weidi Xie, and Andrew Zisserman. GMOS: Grounding moving object segmentation in 3D space and time. In *NeurIPS*, 2026.

[84] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: comprehensive benchmark suite for video generative models. In *CVPR*, 2024.

[85] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: a large-scale in-the-wild robot manipulation dataset. In *RSS*, 2024.

[86] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023.

[87] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv*, 2510.13626, 2025.

[88] Letian Wang, Chuhan Zhang, Rishabh Kabra, Jasper Uijlings, Steven Waslander, Andrew Zisserman, Joao Carreira, Kaiming He, Misha Andriluka, Eduard Gabriel Bazavan, Andrei Zanfir, and Cristian Sminchisescu. Video generation models are general-purpose vision learners. In *ECCV*, 2026.

[89] Chensheng Peng, Quentin Herau, Jiezhi Yang, Yichen Xie, Yihan Hu, Wenzhao Zheng, Matthew Strong, Masayoshi Tomizuka, and Wei Zhan. UniQueR: Unified query-based feedforward 3D reconstruction. In *ECCV*, 2026.

[90] MiniMax-AI. MiniMax H3. Official GitHub repository, 2026.

[91] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *PVLDB*, 16(12):3848–3860, 2023.

[92] Yang Zhou, Yifan Wang, Jianjun Zhou, Wenzheng Chang, Haoyu Guo, Zizun Li, Kaijing Ma, Xinyue Li, Yating Wang, Haoyi Zhu, Mingyu Liu, Dingning Liu, Jiange Yang, Zhoujie Fu, Junyi Chen, Chunhua Shen, Jiangmiao Pang, Kaipeng Zhang, and Tong He. OmniWorld: A multi-domain and multi-modal dataset for 4D world modeling. In *ICLR*, 2026.

[93] Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. UniMax: Fairer and more effective language sampling for large-scale multilingual pretraining. In *ICLR*, 2023.

[94] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. *arXiv*, 2511.16719, 2025.

A Changes from the Submitted Version

The task, title, and central idea are unchanged: HOLO4D exposes source-view 4D reconstruction to a pretrained video diffusion model to reshoot a monocular video along a new camera path. After review, we simplified conditioning, retrained the model, and expanded evaluation. Changed checkpoints, resolutions, and camera evaluators preclude direct comparison with submitted numbers. Table 8 summarizes the differences; the main one is that the rendered RGB-D (depth) branch is removed and only the 3D-track branch is kept. Figure 7 reproduces the submitted pipeline, and Figures 8 to 10 reproduce its qualitative results.

Table 8: **Submitted version versus this version.** The submitted column summarizes the described design; the rendering used for the submitted results is distinguished below.

	Submitted version	This version
Source video	VAE latents, width-concatenated with target latents	Unchanged
RGB-D branch	Target-view RGB, normalized depth, and validity rendered from a static/dynamic point cache; Conv3d geometry tokens	Removed
Track branch	Tracked positions; z-buffer aggregation in the described design; fused with RGB-D tokens	Positions, source color, target-camera depth, and validity; channel-wise max with occupancy; injected along channels in every DiT block
Target cameras	Plücker rays	Plücker rays, encoded into camera features
Backbone	Wan2.1-I2V-14B; Wan2.1-T2V-1.3B	Wan2.1-T2V-14B
Resolution	144×256, then 288×512	144×256, then 384×672
Depth	Separate readout head on VAE decoder features	No readout; depth reconstructed from generated videos
Camera evaluator	ViPE	VGGT-Ω and Pi3
Video quality	FVD and CLIP on DAVIS	VBench on DAVIS; FVD on Syn4D
Robot videos	–	Reshooting, long videos, policy transfer

Submitted conditioning. The submitted paper described four conditioning signals: the source video, a hybrid RGB-D cache, target-camera Plücker rays, and 3D-track tokens. A 3D-motion threshold separated static and dynamic pixels. Static pixels were fused from $K = 7$ keyframes selected by farthest-point sampling over camera positions and viewing directions; dynamic pixels came from the synchronized source frame. The described design merged their renders by z-buffer selection into target-view RGB, normalized depth, and validity. Track tokens encoded 3D trajectory positions and were fused with the RGB-D tokens. The actual submitted 14B DAVIS and iPhone results used multiframe softsplat RGB-D conditioning; the 1.3B DAVIS results used single-frame z-buffer conditioning. Both included 3D tracks.

Final conditioning. This version removes the rendered RGB-D branch, which supplied target-view RGB, depth, and validity, and keeps only the 3D-track branch, together with the source video and target-camera rays (Section 3, Figure 2 versus Figure 7). Observations encode source color and target-camera depth (Equation (3)); validity combines upstream validity with in-image projection, without the occlusion test of z-buffer visibility. Since a static point is a trajectory with a constant position, the static–dynamic threshold and keyframe selection are no longer needed. In a controlled stage-1 comparison, adding rendered RGB and masks to the trajectory features did not improve camera control (Table 4); Tables 15 to 17 report further comparisons with rendered conditioning from earlier checkpoints. Track features enter every DiT block along channels and add no attention tokens (Appendix J).

Model and training. The submission reported a 14B model built on Wan2.1-I2V-14B and a 1.3B model built on Wan2.1-T2V-1.3B, both trained at 144×256 and 288×512. The main results in this version use a 14B model initialized from Wan2.1-T2V-14B and trained at 144×256 and then 384×672, with a revised data mixture (Appendices B and C). Early 1.3B checkpoints appear only in the conditioning study of Table 15, not as main results. The submission also probed target-view depth with a separately trained head on intermediate VAE decoder features; this version omits that probe and evaluates depth reconstructed from generated RGB videos (Table 20).

Evaluation. Cameras of generated videos are now recovered with VGGT-Ω instead of ViPE (Appendix D). Video quality on DAVIS is measured with VBench; FVD is reported on Syn4D, where

target-view references exist. New evaluations cover Vista4D's 110-trajectory benchmark (Appendix E), Syn4D with exact target RGB and geometry (Appendix I), per-scene optimization with ViDAR (Table 18), alternative source reconstructors used without retraining, noise injected into the trajectories (Tables 5 and 21), and runtime (Appendix J). Several of these were requested by reviewers and first reported in the author response.

Robot videos. Section 4.5 is new. It shows zero-shot robot-video reshooting with HOLO4D-14B, long-video generation using a continuation model trained partly on robot videos (Appendix B), and Fast-WAM policy learning initialized from a 5B HOLO4D variant (Appendix L).

Presentation. The abstract, introduction, and method are rewritten for trajectory-based conditioning, and all figures are redrawn.

Submitted visual results. For reference, Figures 8 to 10 reproduce the qualitative results of the submitted version: the frames are unchanged and are laid out in the format of the current figures. They come from the submitted checkpoints with rendered RGB-D and track conditioning; the 14B DAVIS and iPhone results used multiframe softsplat rendering. The corresponding results of the final model are in Figures 3, 4, 15 and 16.

B Implementation Details

GMOS preprocessing. For the DAVIS two-stage backend, GMOS [83] separates query-frame background and foreground identities. Background points are fixed at their query-frame 3D positions and kept valid; foreground points retain their depth-lifted trajectories and validity. Both enter the same track encoder, without additional mask channels or a separate foreground/background branch.

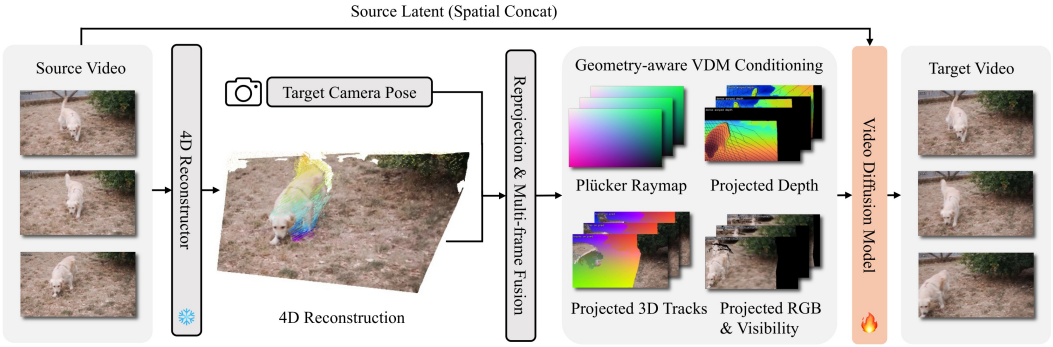

Figure 7: **Submitted pipeline** (reproduced from the submitted version). Static and dynamic source pixels were rendered into a hybrid target-view RGB-D cache (projected RGB, depth, and visibility), which conditioned the video diffusion model together with Plücker rays and projected 3D tracks. This version removes the rendered RGB-D branch and keeps the track branch (Figure 2).

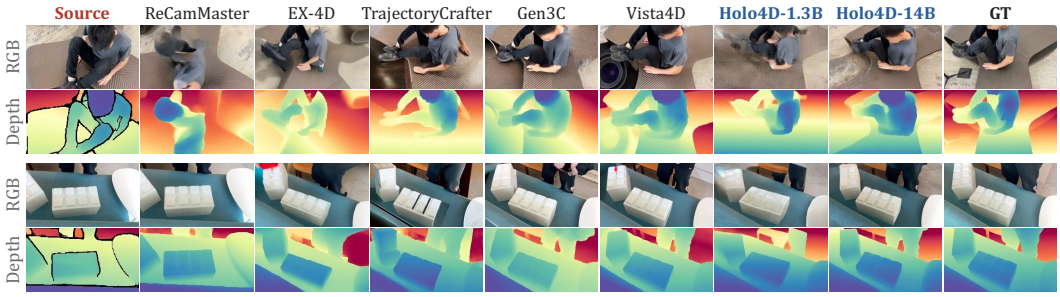

Figure 8: **Submitted novel-view RGB and depth on iPhone** (frames from the submitted version). Two scenes (spin, block), each shown as a target-view RGB row over a depth row; GT is the held-out view, and its depth is a 4RC pseudo-reference. HOLO4D-1.3B and HOLO4D-14B are the submitted checkpoints. Final-model results are in Figure 4.



Figure 9: **Submitted qualitative comparison on DAVIS** (frames from the submitted version). "Holo4D (submitted)" is the submitted 14B checkpoint with rendered RGB-D and track conditioning; the second row shows source pixels reprojected into the target view (labelled GT in the submission). See Figures 3 and 16 for the final model.

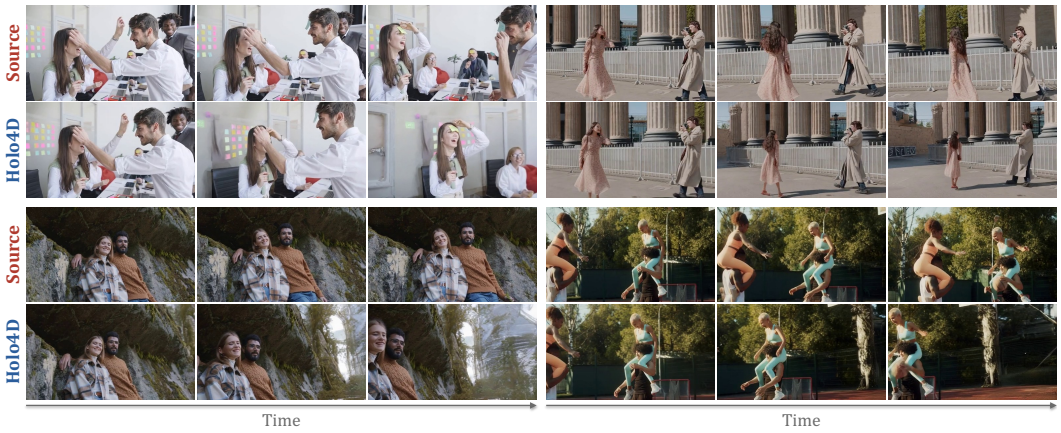

Figure 10: **Submitted qualitative results on real-world videos** (frames from the submitted version). For each of four videos, source frames are shown above the target-view frames generated by the submitted model; three of the five frames shown in the submission are kept. Final-model results are in Figure 15.

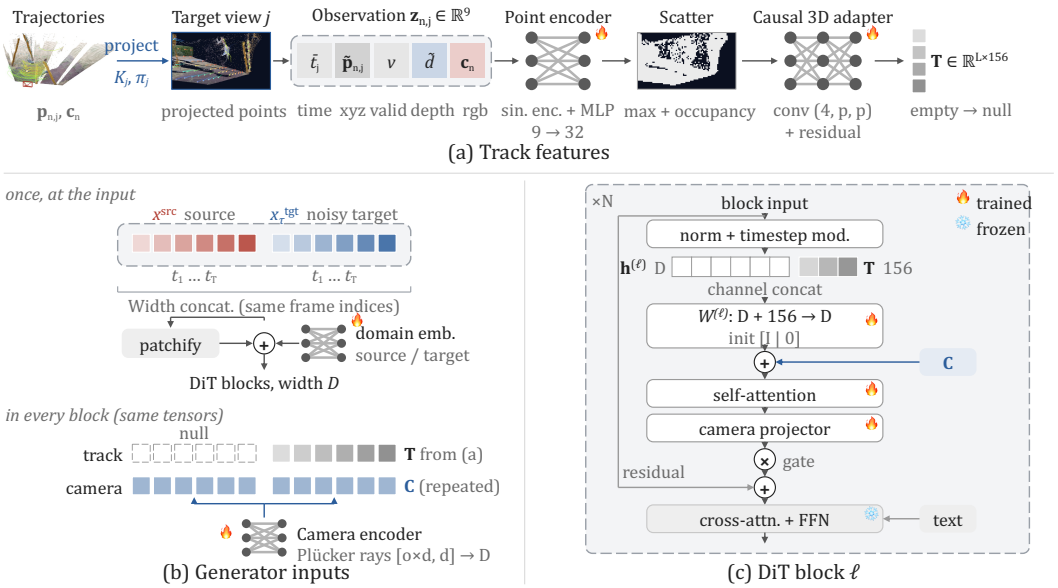

Figure 11: **Trajectory conditioning.** (a) Each trajectory is projected into target frame j; its observation $\mathbf{z}_{n,j}$ (frame time, normalized position, validity, target-camera depth, and source color) is encoded, scattered to the target grid with a channel-wise maximum and an occupancy channel, and brought to the DiT token grid by a causal 3D adapter, giving $\mathbf{T}$. (b) Source and noisy target latents are concatenated along width once, at the input; in every block, $\mathbf{T}$ occupies the target half (null on the source half) and the camera features $\mathbf{C}$ from one shared encoder are repeated on both halves. (c) Each of the 40 DiT blocks concatenates $\mathbf{T}$ with the normalized hidden features, projects back to width D with $W^{(\ell)}$ initialized as $[I \,|\, 0]$, and adds $\mathbf{C}$ before self-attention; the camera projector is identity-initialized and gated. Flames and snowflakes mark trained and frozen modules (network icons: dark = trained, light = frozen); token shading runs from the first to the last frame. Standard cross-attention/FFN residuals and the auxiliary query-frame input are omitted.

Source–target alignment. Following the dynamic point-map notation of V-DPM [8], $P_i^{\mathrm{src}}(t_j, \pi_0^{\mathrm{src}})(u)$ denotes the position at video time t_j of the point observed at pixel u in source frame i, expressed in the first source-camera frame. For point n, queried at pixel u_n in source frame q_n, the main text uses $\mathbf{p}_{n,j} = P_{q_n}^{\mathrm{src}}(t_j, \pi_0^{\mathrm{src}})(u_n)$ and $\mathbf{c}_n = I_{q_n}^{\mathrm{src}}(u_n)$. Source and target latents are concatenated along width, so the two views retain the same temporal indices. The source half receives null track features; projected track features guide the target half. Source cameras define the common geometric frame, while target cameras determine projection and the Plücker-ray condition. No separate source-camera rays are encoded.

Backbone. We initialize from Wan2.1-T2V-14B [70], which has 40 DiT blocks and a 16-channel video VAE with spatial compression 8 and temporal compression 4. We fine-tune self-attention and the new conditioning modules, while keeping cross-attention and feed-forward layers frozen. The new modules comprise the source–target domain embedding, query-frame indicator embedding, track encoder and injectors, and camera adapter and projectors.

Track encoding. For point n in video frame j, the input attributes are

$$\mathbf{z}_{n,j} = \left[\bar{t}_j;\ \tilde{\mathbf{p}}_{n,j};\ v_{n,j};\ \tilde{d}_{n,j};\ \mathbf{c}_n\right] \in \mathbb{R}^9. \tag{3}$$

The frame index is normalized as $\bar{t}_j = j / \max(N-1, 1)$. Tildes denote normalized position and depth: each position coordinate is scaled using its per-sample minimum and maximum over all point identities and frames, then zeroed for invalid observations. Depth is scaled by its per-sample maximum. Validity combines the upstream track mask with valid projection into the target image; it is not an exact target-view occlusion test. Depth is the point's target-camera z coordinate, rather than a separate depth-image condition. Source color is sampled at the query frame and held fixed along the track. We discard identities invalid in every target frame, then subsample or pad to $M = HW$ identities, with padded rows inactive. At 384×672, this gives $M = 258{,}048$.

A sinusoidal encoding and MLP map observations to 32-dimensional features. Target-view projection determines where to scatter these features, with channel-wise maximum aggregation and an occupancy indicator. Temporal patchification repeats the first frame to respect the VAE's causal layout, then applies a 3D convolution with kernel and stride $(4, p, p)$, where p is the spatial downsampling factor, followed by one causal residual block and projection to 156 channels. The resulting features match the DiT token grid. Empty tokens use a learned, zero-initialized embedding. The source half of the width-concatenated latent sequence receives null track tokens.

Camera conditioning. One shared camera encoder produces the target-camera features used in every DiT block. These features are repeated along width to match both source and target tokens. This repetition does not introduce source-camera rays; explicit source-camera rays would be compatible with this design but are untested. Each block has its own track concat–projection and camera output projector. Camera rays use the Plücker convention $[\mathbf{o} \times \mathbf{d}, \mathbf{d}]$. Pixel unshuffle and a 2D convolution patchify the ray map; a zero-initialized pointwise 3D convolution maps its six channels to the DiT width. Within each block, track concatenation and projection follow normalization and timestep modulation. The camera feature is then added before self-attention, whose output passes through an identity-initialized camera projector and the gated residual connection. Cross-attention and the feed-forward sublayer follow.

Query-frame indicator. A query-frame indicator identifies the source frames used to initialize trajectories. A binary sequence is one at the selected query frame and zero elsewhere; each value is broadcast over the image and repeated across three channels for encoding with the video VAE. The resulting control latents occupy the source half of the width-concatenated input, with zeros on the target half. A separate, zero-initialized Conv3D patch embedding maps these control latents to the hidden width, and its output is added to the video patch embeddings before the DiT blocks. This input addition uses neither the per-block track concat–projection nor a concatenation with the RGB latent channels.

Training. Stage 1 uses 49 frames at 144×256 for 15,000 steps, with effective batch size 8 and learning rate 1×10^{-5}. Stage 2 continues the same checkpoint to step 19,500 at 384×672, with effective batch size 32 and learning rate 2×10^{-5} on 32 GPUs using FSDP [91]. Both stages run on NVIDIA GH200 GPUs for about 24 hours each. Both stages optimize RGB flow matching. No geometric-conditioning dropout is used; text-conditioning dropout is 0.1.

Long-video rollout. We extend generation beyond the 49-frame training window by following Vista4D's window-by-window inference [11]. (i) Split the source into 49-frame windows with one-frame overlap: k windows cover $49 + 48(k-1)$ frames. Pad the last window by repeating its final source frame. (ii) Reconstruct each window independently using the backends specified below, querying trajectories at its first frame; identities are not shared across windows. Register each reconstruction to the first window through the shared boundary frame, holding its source camera fixed while fitting a scale to the shared point map. Use the corresponding segment of the requested target-camera path. (iii) Generate the first window with HOLO4D-14B (step 19,500), conditioned on the source video, tracks, and cameras. (iv) Generate later windows with an image-to-video continuation model using the same conditions plus the previous window's last generated frame. This shared boundary image is never a source or ground-truth frame. Remove duplicate boundary frames and padding when stitching.

The continuation model uses a Wan2.1-I2V-14B [70] backbone and the stage-1 HOLO4D-14B track and camera modules. Its image condition enters through the first-frame VAE latent and CLIP image feature. During training, this is the first ground-truth target frame, dropped with probability 0.3 and noised at level 0.05. Training proceeds at 144×256 from step 15,000 to 19,000, then at 384×672 to step 22,500, with effective batch size 32 and learning rate 2×10^{-5}. The mixture is MultiCamVideo (22.5%), Syn4D (30%), DL3DV (15%), MVS-Synth (7.5%), and OmniWorld-DROID [92, 85] (25%); unlike HOLO4D-14B, this model has seen robot videos. Window 1 uses 4RC for DROID and the two-stage backend for humanoid clips; later windows use 4RC without GMOS. Each 4RC reconstruction uses all 49 source frames. All windows use 50 denoising steps, CFG 5, shift 5, and seed 42 (Figures 6 and 18).

Condition augmentation. Stage 2 perturbs geometric conditions with probability 0.25. The configured track-offset and temporal-jitter standard deviations are 0.005 and 0.002; image-coordinate noise ratio is 0.10. Frame-depth jitter, depth offset, and depth-scale perturbation use 0.20, 0.002, and 0.003, respectively. RGB perturbations use 0.003/0.006, and mask augmentation probability is 0.10.

Table 9: **Training data.** Sampling weights (%) per stage, not proportions of distinct videos. Static scenes need no tracker: backprojected source points stay fixed in 3D. Dynamic pairs come from synchronized multiview videos; static pairs from different chunks of the same video. DA3: DepthAnything 3 [4].

Index	Dataset	Real / Syn.	Dyn. / Static	Depth, cameras	3D tracks	Stage 1	Stage 2
1	Syn4D [73]	Synthetic	Dynamic	Ground truth	Ground truth	31.65	40.00
2	MultiCamVideo [1]	Synthetic	Dynamic	VGGT-Ω	CoWTracker	0.00	30.00
3	DL3DV [75]	Real	Static	DA3 + dataset	Fixed points	32.91	20.00
4	MVS-Synth [77]	Synthetic	Static	Ground truth	Fixed points	6.33	10.00
5	Kubric [74]	Synthetic	Dynamic	Ground truth	Box motion	12.66	0.00
6	TartanAir [78]	Synthetic	Static	Ground truth	Fixed points	12.66	0.00
7	RE10K [76]	Real	Static	DA3 + dataset	Fixed points	3.80	0.00

Sparse-track sampling is enabled with probability 0.5, with strides 2, 4, or 8; multi-query sampling uses two or three query frames with probability 0.5.

C Training Data

Table 9 reports sampling weights for both training stages. The first stage uses a broad mixture of synthetic and real, static and dynamic scenes; the second concentrates on four datasets at higher resolution. Geometry supplies conditions for RGB generation, not auxiliary prediction targets. Syn4D's `flying_group` and MVS-Synth sequence `0000` are excluded from both stages. Weights are rounded to two decimal places; category totals are computed before rounding.

Stage 1 combines Syn4D and two equally weighted Kubric subsets for dynamic scenes, together with real static scenes from DL3DV and RE10K and synthetic static scenes from MVS-Synth and TartanAir. Each Kubric subset contributes approximately 6.33% of the stage-1 sampling weight. Stage 2 uses Syn4D and MultiCamVideo for dynamic scenes and DL3DV and MVS-Synth for static scenes, increasing the dynamic share from 44.30% to 70%.

Table 9 lists the geometry and track source of each dataset (VGGT-Ω [5], CoWTracker [18], DepthAnything 3 [4]). The 30% second-stage MultiCamVideo weight comprises 21% dense and 9% sparse query-frame data.

For static scenes, we form source–target pairs from different chunks of the same monocular video, similar to 4Real-Video-V2 [79]; temporal synchronization is unnecessary since the scene is static. Dynamic training pairs come from synchronized multiview videos; we do not use dynamic monocular videos for training.

D Camera-Adherence Evaluation on DAVIS

We construct our DAVIS evaluation set by selecting 50 videos [80] and randomly sampling two target trajectories per video from the ReCamMaster [1] and Vista4D [11] trajectory pools. Baselines use released checkpoints and native generation resolutions; Vista4D uses 4RC-derived source geometry for this DAVIS comparison. We evaluate the first 49 output frames. Camera recovery with VGGT-Ω and Pi3 uses 384×672 frames; VBench uses the same resolution at 16 FPS with each clip's caption. The main HOLO4D result uses 4RC source depth, cameras, and dense tracks, with 50 denoising steps, CFG 5, shift 5, and seed 42.

Table 10: **Backbones and native resolutions.**

Method	Backbone	Res.
ReCamMaster	Wan2.1-T2V-1.3B	480×832
TrajectoryCrafter	CogVideoX-5B	384×672
EX-4D	Wan2.1-I2V-14B	512×512
Gen3C	Cosmos-7B	704×1280
Vista4D	Wan2.1-T2V-14B	384×672
HOLO4D-14B	Wan2.1-T2V-14B	384×672

VGGT-Ω recovers cameras from generated videos. Translation metrics use the 54 translation-bearing trajectories; rotation uses all 100. DAVIS has no captured target-view reference for these paths, so we report camera control and VBench scores. All methods are evaluated with the same VGGT-Ω evaluator revision on all 100 videos; baseline videos are the stored outputs of their released models, not resampled. For ATE and RPE-T, we anchor both trajectories at their first camera and rescale the predicted camera centers to match the centered Frobenius norm of the requested centers. We average per-video RMSEs; relative errors use adjacent frames. No subsequent global rotation alignment is

applied to ATE, so orientation errors after the first frame contribute to it. Its magnitude also depends on the requested trajectories and is not directly comparable across DAVIS and Vista4D trajectory sets.

Table 12 evaluates the same checkpoint with the two-stage backend, where GMOS static/dynamic fusion fixes background identities in 3D. At 50 steps, this condition lowers ATE from 1.657×10^{-2} to 0.719×10^{-2}, while its VBench scores are lower (Table 11). With this condition, 25 denoising steps give a lower rotation error than 50 (0.077° versus 0.101°). The 4RC condition uses its reconstructed trajectories directly, without GMOS preprocessing. A 5B model with a dense identity-target track condition (Wan2.2-TI2V-5B, generated at 704×1280 with 50 steps) behaves like the 14B model: with 4RC it has ATE 1.464×10^{-2} and rotation error 0.116°, and the two-stage backend with GMOS lowers these to 0.608×10^{-2} and 0.105° (Table 12). The HOLO4D-5B† row in Table 12 is a 5B variant trained on OmniWorld-DROID, MultiCamVideo, TartanAir, and Kubric, excluding Syn4D, MVS-Synth, and DL3DV. With the same source condition and 25 steps, its errors (ATE 0.914×10^{-2}, rotation 0.106°) remain below those of Vista4D. It uses step 29,000 and a dense track condition without persistent identities. It also differs from HOLO4D-14B in backbone and track condition, so this comparison is not a controlled data ablation. The Pi3 columns use source-aligned camera recovery. All HOLO4D variants have lower Pi3 translation error than Vista4D; only the 14B variant with VGGT-Ω at 25 steps also has lower Pi3 rotation error.

Compatibility with a smaller backbone. The geometric interface also applies to Wan2.2-TI2V-5B with a variant of the track condition (Table 12). At 704×1280 with 50 denoising steps on DAVIS, replacing 4RC with the two-stage backend and GMOS reduces ATE from 1.464×10^{-2} to 0.608×10^{-2}, and on the nine-clip Syn4D comparison its source-aligned rotation error rises from 0.515° with ground-truth geometry to 0.885° with 4RC estimates (Table 14). Both trends match the 14B model.

Sampling time of the 5B model. On one H200 with 50 denoising steps and 49 frames, HOLO4D-5B generates a 704×1280 video in 202.0 s on average, versus 417.4 s for HOLO4D-14B at 384×672 and 495.0 s for Vista4D at 384×672 (plus 6.2 s of point-cloud rendering), averaged over the same 100 Vista4D trajectory pairs; peak GPU memory is 76.3, 43.2, and 49.9 GiB, respectively. For these two configurations, the measured sampling time of the 5B model is about $2.1\times$ lower than that of the 14B model; because backbone, resolution, and track condition all differ, this is not a speedup under matched conditions. These are complete sampling times with precomputed source reconstruction from the 100-pair evaluation runs, not the matched two-trajectory profiling in Table 24, so the 14B totals differ between the two measurements.

GMOS within the same reconstruction backend. The shared trajectory representation supports foreground/background fusion without modifying the generator: for the two-stage backend, background points are held fixed in 3D while foreground points keep their tracked motion, and a foreground mask decides which is which. Our default 4RC condition uses no explicit fusion. Table 13 compares two foreground masks within the same static/dynamic fusion of the two-stage backend: GMOS masks and masks derived from track motion. Both rows use the step-19,500 checkpoint, the same VGGT-Ω and CoWTracker caches with first-frame queries, 50 denoising steps at 384×672, CFG 5, shift 5, and seed 42; the saved sampling settings are otherwise identical. In both rows, background identities are fixed at their query-frame positions and foreground identities keep their tracked motion, so the comparison isolates the segmentation, not fusion versus no fusion. GMOS lowers the VGGT-Ω errors (ATE by 27%, RPE-T by 15%, RPE-R by 12%) and the Pi3 rotation error by 3%, while the Pi3 translation error is essentially unchanged. The GMOS row is the 50-step VGGT-Ω row of Table 12. Figure 12 illustrates the fusion on one DAVIS sequence.

Table 11: **VBench across reconstruction backends on DAVIS.** α: 4RC; β: two-stage backend with GMOS. 5B rows use Wan2.2-TI2V-5B at 704×1280. Columns follow Table 2.

Method	Steps	Aest. ↑	Imag. ↑	Subj. ↑	Backg. ↑	Temp. ↑
Vista4D$^\alpha$	50	0.478	0.534	<u>0.904</u>	0.926	**0.256**
HOLO4D-14B$^\alpha$	50	<u>0.493</u>	**0.601**	**0.910**	**0.931**	<u>0.253</u>
HOLO4D-14B$^\beta$	50	0.478	<u>0.589</u>	0.891	0.919	0.252
HOLO4D-14B$^\beta$	25	0.471	0.580	0.883	0.910	0.249
HOLO4D-5B$^\alpha$	50	**0.501**	0.559	0.894	<u>0.928</u>	0.253
HOLO4D-5B$^\beta$	50	0.489	0.554	0.881	0.920	0.252

Table 12: **Source geometry and sampling steps on DAVIS.** α: 4RC; β: two-stage backend with GMOS. 5B rows use Wan2.2-TI2V-5B at 704×1280 (384×672 for †). †Trained without Syn4D, MVS-Synth, and DL3DV, with a dense track condition without persistent identities. ATE is scaled by 10^2; RPE-T and TransErr^2 by 10^3. Rotations are in degrees.

Method	Steps	VGGT-Ω			Pi3	
		ATE ↓	RPE-T ↓	RPE-R ↓	TransErr^2 ↓	RotErr ↓
Vista4D$^{\alpha}$	50	3.576	3.608	0.132	6.201	<u>2.070</u>
HOLO4D-14B$^{\alpha}$	50	1.657	2.415	0.102	<u>3.206</u>	2.332
HOLO4D-14B$^{\beta}$	50	<u>0.719</u>	<u>1.299</u>	<u>0.101</u>	3.359	2.284
HOLO4D-14B$^{\beta}$	25	0.729	**1.095**	**0.077**	3.625	**1.785**
HOLO4D-5B$^{\alpha}$	50	1.464	2.952	0.116	**3.195**	2.275
HOLO4D-5B$^{\beta}$	50	**0.608**	1.302	0.105	3.425	2.199
HOLO4D-5B$^{\beta,\dagger}$	25	0.914	1.363	0.106	3.604	2.270

Table 13: **Segmentation for static/dynamic track fusion on DAVIS.** Two-stage backend with the same first-frame tracks, 384×672, 50 steps; only the foreground mask differs. ATE: $\times 10^2$; RPE-T and TransErr^2: $\times 10^3$; rotations: degrees.

Mask	ATE ↓	RPE-T ↓	RPE-R ↓	TransErr^2 ↓	RotErr ↓
Motion-derived	0.984	1.527	0.114	**3.351**	2.364
GMOS	**0.719**	**1.299**	**0.101**	3.359	**2.284**

E Camera Adherence on Vista4D Trajectories

We additionally evaluate on the 110 native Vista4D trajectories with both generated-only VGGT-Ω recovery and Vista4D's source-aligned Pi3 protocol (Table 6). Vista4D uses the benchmark's released Pi3 reconstruction, masks, and rendered inputs, whereas HOLO4D uses 4RC or the two-stage backend. This preserves Vista4D's native preprocessing and compares complete pipelines; it does not isolate the video generator as the source of performance differences. For the qualitative comparisons in Figures 3 and 16, baselines use their released sampling settings and the benchmark's released Pi3 geometry where a point cloud is required; Vista4D videos are its official outputs. The VGGT-Ω and Pi3 evaluators recover cameras for scoring and are separate from the source reconstruction used for generation. With 4RC source geometry, HOLO4D has lower RPE-T under VGGT-Ω and lower translation error under Pi3, but higher rotation error under both evaluators. VGGT-Ω evaluation follows the normalization in Appendix D. ATE and RPE-T have 110 valid values for HOLO4D with 4RC and 109 each for Vista4D and the two-stage condition; RPE-R uses all 110. These are per-method valid-metric counts, not a common-subset evaluation. Scale normalization requires non-degenerate requested and recovered camera paths; a completed video can therefore lack a valid translation metric. Vista4D's Pi3 protocol reconstructs source and generated frames jointly, then aligns the source cameras by $\mathrm{Sim}(3)$.

The 4RC condition in Table 6 uses its reconstructed trajectories without mask-guided fusion and aligns the geometry to the released camera coordinate system with a per-video scalar depth factor. An alternative pointmap $\mathrm{Sim}(3)$ alignment of the same checkpoint and 4RC geometry reduces VGGT-Ω RPE-R from $1.142°$ to $0.348°$, but increases Pi3 RotErr from $14.98°$ to $17.19°$. Thus the source-condition comparison includes alignment effects, and the two evaluators disagree on this preprocessing change.

Additional geometry comparisons. Section 4.4 analyzes the ground-truth and 4RC comparisons in Table 14. All rows use nine held-out Syn4D `flying_group` clips, with source camera 0 and target camera 1. Pi3 [28] recovers cameras jointly from source and generated frames. In the additional Pi3-source group, HOLO4D constructs tracks from estimated depth and cameras using ground-truth 2D correspondences and visibility, whereas Vista4D uses its native Pi3 reconstruction. This partially oracle condition is distinct from both the full ground-truth and 4RC conditions. The 5B model is Wan2.2-TI2V-5B with a dense identity-target track condition, generated at 384×672; its source-aligned rotation error with Pi3 geometry is $0.740°$.

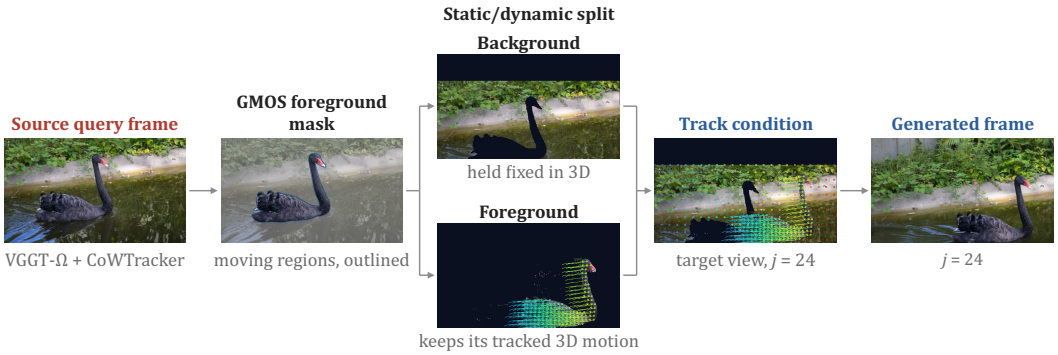

Figure 12: **GMOS in the two-stage backend.** VGGT-Ω supplies depth and cameras and CoWTracker supplies tracks. GMOS segments the moving regions in the query frame; trajectories outside the mask are held at their query-frame 3D positions, and trajectories inside keep their tracked motion. Both sets form one track condition, shown projected into the target view at $j = 24$. Background regions occluded by the foreground in the query frame have no trajectories. DAVIS *blackswan*, final checkpoint, 50 denoising steps.

Table 14: **Camera control with GT and estimated source geometry on Syn4D.** Nine held-out clips; Pi3 camera recovery. ATE: $\times 10^2$; RPE-T and TransErr^2: $\times 10^3$; rotations: degrees. Bold/underline: best/second best per geometry group. Protocols: Appendix E.

Method	Geometry	ATE ↓	RPE-T ↓	RPE-R ↓	TransErr^2 ↓	RotErr ↓
Vista4D	GT	2.233	12.12	<u>0.079</u>	13.36	<u>0.372</u>
HOLO4D-14B	GT	**1.299**	**8.121**	**0.051**	**1.881**	**0.242**
HOLO4D-5B	GT	<u>1.957</u>	<u>11.19</u>	0.080	<u>5.023</u>	0.515
Vista4D	Pi3	2.348	<u>10.15</u>	<u>0.067</u>	19.26	<u>0.651</u>
HOLO4D-14B	Pi3 + GT 2D	**1.330**	**8.158**	**0.052**	**5.715**	**0.534**
HOLO4D-5B	Pi3 + GT 2D	<u>1.996</u>	10.40	0.084	<u>13.50</u>	0.740
HOLO4D-14B	4RC	**1.203**	**8.280**	**0.055**	**24.15**	<u>0.931</u>
HOLO4D-5B	4RC	<u>2.263</u>	<u>12.63</u>	<u>0.087</u>	<u>34.46</u>	**0.885**

On Vista4D's native trajectories, source replacement changes camera estimates, geometry, and coverage together. Separating generator sensitivity from reconstruction error would require running both methods with the same estimated geometry on these trajectories.

Conditioning signals in early checkpoints. To ask whether richer geometric conditions repair rotation, we compare four separately trained 1.3B checkpoints that add rendered RGB-D, tracks, and multi-frame rendering to Plücker camera rays (Table 15). The Pi3 translation error decreases monotonically from 0.987 to 0.685 as conditions are added, whereas rotation error does not decrease; rendered RGB-D raises it by 2.33° (95% confidence interval $[0.18°, 4.73°]$ over the 20 pairs). These early checkpoints barely follow the requested rotation, so the experiment shows only that added geometric conditions improve translation more readily than rotation; it does not measure the rotation accuracy of the final model. All four Wan2.1-T2V-1.3B checkpoints are evaluated at step 3,000, using the same 4RC reconstructions, $288 \times 512 \times 49$ outputs, and 50 denoising steps. The 20 pairs come from ten sources, five selected for rotation failures, with metrics as in Table 6.

F Novel-View Evaluation on iPhone/DyCheck

We use the five iPhone/DyCheck scenes and source–target split of Vista4D [11]. One synchronized camera supplies the source video and a held-out camera supplies the target trajectory and reference RGB. Target references and masks are center-cropped to 16:9 and resized to 288×512. Predictions are resized to the same canvas and evaluated over 49 frames; EX-4D's square outputs are resized directly, without an additional center crop. Masked metrics use the intersection of valid and co-visible pixels, not a foreground-object mask. The two HOLO4D rows share the same checkpoint and use 50 denoising steps, CFG 5, shift 5, and seed 42. For both source conditions, source depth, cameras, and tracks are calibrated per scene to the source iPhone LiDAR scale.

Table 15: **Conditioning in early 1.3B checkpoints.** Rotation gain is the fitted ratio of generated to requested rotation. ATE is scaled by 10^2; RPE-T by 10^3. Rotations are in degrees.

Condition	VGGT-Ω			Pi3, source-aligned		Rot. gain ↑
	ATE ↓	RPE-T ↓	RPE-R ↓	TransErr^2 ↓	RotErr ↓	
Plücker only	**15.03**	60.92	**1.423**	0.9866	**31.26**	**0.0218**
+ rendered RGB-D	16.21	<u>41.69</u>	<u>1.565</u>	0.7625	33.59	0.0027
+ rendered RGB-D + tracks	<u>16.14</u>	45.22	1.642	<u>0.7117</u>	<u>33.25</u>	0.0096
+ multi-frame RGB-D + tracks	16.39	**34.69**	1.635	**0.6854**	33.29	<u>0.0149</u>

EX-4D generates at its native 512×512 resolution with 25 denoising steps and uses the dataset's source-view depth. The VGGT-Ω source condition uses CoWTracker tracks. Neither iPhone source condition uses GMOS. All RGB rows, including EX-4D, are rescored from stored outputs with one evaluator. The iPhone columns of Table 20 report depth reconstructed by 4RC or VGGT-Ω from both the generated and the held-out target RGB; the target reconstruction is a pseudo-reference, not sensor ground truth. All iPhone depth rows, baselines included, use the same scorer, crop, valid co-visible mask, and confidence threshold, and are equal-weight means over the five scenes.

G Conditioning Comparisons

Stage-1 design study. The variants in Table 4 share the backbone initialization, training data, optimizer, and $15{,}000$ steps at 144×256. We evaluate them on DAVIS at 288×512 with 4RC source geometry, 25 denoising steps, and VGGT-Ω camera recovery. The rendered-feature variant encodes rendered RGB together with validity and source/target motion masks, then sums the resulting features with the track features before injection, without adding attention tokens.

Additional conditioning variants. Table 16 compares camera-ray, rendered-video, and trajectory conditioning. The upper block reports variants with different training schedules and evaluation sets; the lower block compares rendered streams on a common evaluation subset. In the lower block, adding rendered streams reduces translation errors but increases rotation error. This trade-off complements the controlled stage-1 comparison in Table 4. All variants use a 14B backbone with source-video and camera conditioning; cameras are recovered with VGGT-Ω. In the upper block, the camera-only model uses 10k training steps, track-payload variants use 15k, and rendered variants use a different evaluation suite. The reported translation/rotation counts are $54/100$ for the upper block and $36/63$ for the lower block, so magnitudes should not be compared across blocks.

Table 16: **Track and rendered conditioning on DAVIS.** PCD denotes rendered point clouds. The unmatched upper block is not ranked; lower-block bold and underline mark **best** and <u>second-best</u> results. ATE and RPE-T are scaled by 10^2 and 10^3; rotations are in degrees.

Condition	ATE ↓	RPE-T ↓	RPE-R ↓
Conditioning variants (training not matched)			
Plücker rays only	10.65	7.32	0.373
PCD(RGB-D) + valid mask, no tracks	10.36	9.16	0.236
PCD(RGB-D) + 3D tracks	2.35	2.38	0.086
Tracks without per-track color	2.33	2.13	0.094
3D tracks only (HOLO4D)	1.62	2.08	0.065
Rendered-stream variants (common-success camera subset)			
3D tracks only (HOLO4D)	2.04	3.27	**0.059**
+ PCD render stream, mask off	**1.32**	2.43	0.076
+ PCD render stream, mask on	<u>1.50</u>	**1.96**	0.076
PCD render streams only, no tracks	**1.32**	<u>2.37</u>	<u>0.071</u>

H Additional Novel-View Synthesis Evaluation

Rendered and trajectory conditioning. Table 17 compares five conditioning variants on five DyCheck sequences. These checkpoints use an earlier architecture with a rendered-conditioning

branch, not the model of the main paper. Both source-backend panels use 49 frames at 288×512, 50 denoising steps, CFG 1, shift 5, and seed 42. Adding tracks improves on the RGB-D tier under both backends; the relative ordering of single- and multi-frame variants depends on the metric and backend. Because neither panel contains a track-only checkpoint, this study does not test removal of the rendered branch.

Table 17: **Earlier conditioning variants on DyCheck.** Source geometry: (a) 4RC; (b) two-stage backend. "m" denotes masked metrics; bold and underline mark **best** and second-best results per panel.

(a) Default 4RC source geometry

Configuration	mPSNR ↑	mSSIM ↑	mLPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Plücker rays only	11.23	0.442	0.630	11.36	0.281	0.717
+ RGB render	11.91	0.466	0.611	11.95	0.314	0.725
+ depth + valid mask	11.93	0.453	0.640	11.93	0.296	0.758
+ 3D tracks (single-frame)	13.58	0.477	**0.490**	13.20	0.319	**0.568**
+ multi-frame static cache	**13.65**	**0.479**	0.497	**13.28**	**0.322**	0.581

(b) Two-stage source geometry

Configuration	mPSNR ↑	mSSIM ↑	mLPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Plücker rays only	11.25	0.443	0.631	11.36	0.282	0.720
+ RGB render	11.92	0.466	0.613	11.94	0.314	0.726
+ depth + valid mask	12.02	0.454	0.641	11.97	0.296	0.759
+ 3D tracks (single-frame)	**13.81**	0.480	**0.482**	**13.40**	0.325	**0.554**
+ multi-frame static cache	13.31	**0.485**	0.510	13.18	**0.330**	0.592

Optimization-based reconstruction. Table 18 compares with ViDAR [26] on the same five scenes. ViDAR is optimized per scene using only the first 49 source frames, with outputs evaluated at 288×512. With the two-stage backend, HOLO4D has better PSNR and LPIPS, while ViDAR has better SSIM; the same ordering holds for masked metrics. ViDAR takes on average 5 h 18 min per scene on an A6000. This distinguishes scene-specific optimization from amortized generation, but is not a hardware-matched speed comparison.

Each scene is optimized once, and the first 49 target frames are evaluated after center cropping and resizing to 288×512. This short-window comparison has less temporal and viewpoint coverage than ViDAR's full-sequence benchmark. We do not compare with CAT4D [57], whose code and checkpoints were not released at the time of our experiments.

I Synthetic Evaluation and Upstream Robustness

Ground-truth source geometry. The Syn4D evaluation with ground-truth source geometry uses five held-out source clips and 32 source–target pairs with exact target RGB and depth. Rendered-geometry baselines use ground-truth source depth and cameras to reproject source RGB; HOLO4D uses ground-truth 3D tracks, and ReCamMaster uses source RGB and cameras. No method receives target RGB, depth, or masks. Target-view image metrics compare directly against rendered target RGB. Syn4D also contributes training data for HOLO4D, whereas baseline checkpoints are not adapted to it. The held-out real-capture results provide a complementary test. HOLO4D uses 25 denoising steps at 384×672 for the ground-truth-source comparison and the reconstruction-backend and track-corruption studies (Tables 3 and 21). Cameras are recovered with VGGT-Ω in both Syn4D comparisons. For these comparisons and the scaffold studies, ATE is the mean of per-pair trajectory RMSEs after $\mathrm{Sim}(3)$ alignment, including rotation, translation, and scale. This differs from the first-frame-anchored, scale-normalized ATE used on DAVIS and the native Vista4D trajectories (Appendices D and E); ATE values should therefore be compared within each evaluation setting. Image metrics and FVD use ground-truth target videos as references, including the upstream-replacement and track-corruption studies on the same 32 ground-truth-source pairs. FVD in Tables 3 and 19 uses the same I3D scorer at $288 \times 512 \times 49$; the two tables still evaluate different pairs, so their values should not be compared with each other. The FVD column of Table 21 comes from an earlier scoring at 256×256 and is comparable only within that table.

Figure 13 shows two of these pairs.

Table 18: **Novel-view synthesis compared with ViDAR.** Under Vista4D's iPhone protocol, ViDAR is optimized per scene on the first 49 source frames, center-cropped and resized to 288×512. We evaluate 49 target frames, unlike ViDAR's full-sequence benchmark. α: 4RC; β: two-stage backend. "m" denotes masked metrics. Bold and underline mark **best** and second-best results.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	mPSNR ↑	mSSIM ↑	mLPIPS ↓
ViDAR [26]	12.90	**0.385**	0.734	13.01	**0.525**	0.653
HOLO4D-14B$^{\alpha}$	12.68	0.326	0.509	13.22	0.484	0.435
HOLO4D-14B$^{\beta}$	**13.20**	0.348	**0.465**	**14.03**	0.507	**0.379**

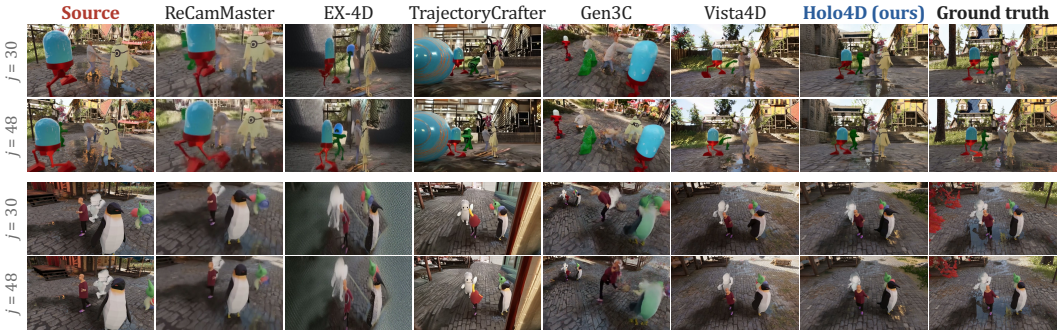

Figure 13: **Syn4D with ground-truth source geometry.** Two held-out pairs with large viewpoint changes; the right column is the rendered target view. Rendered-geometry baselines reproject ground-truth source depth, HOLO4D receives ground-truth 3D trajectories, and ReCamMaster uses only the cameras (Table 3). EX-4D generates square frames, shown as a central crop. HOLO4D places the moving objects as in the ground truth; all methods differ from it in unobserved regions.

FVD reference videos. Using source videos as FVD references, as in Vista4D [11], measures distributional similarity to the inputs and does not establish adherence to the requested target views. A model that returns the source videos unchanged would obtain zero FVD, up to numerical precision under identical preprocessing, even when the requested cameras differ. We therefore report FVD only on Syn4D against ground-truth target videos. FVD remains a distribution-level video-quality metric; camera adherence is evaluated separately using recovered camera trajectories. We assess video quality with VBench (Table 2) and target-referenced FVD; CLIP-based scores are not reported.

Estimated source geometry. Table 19 uses 32 pairs from 16 held-out `flying_group` scenes, with camera pairs $0 \to 7$ and $2 \to 6$. Geometry-consuming methods receive 4RC source estimates. HOLO4D uses 50 denoising steps at 384×672 and has the lowest camera errors and LPIPS; ReCamMaster has the highest PSNR and SSIM, while Vista4D has the lowest FVD. FVD compares all methods against the same 32 target videos at 288×512 over 49 frames; the small sample size makes it a supplementary distributional measure. This test set differs from the ground-truth-source experiment, so differences in their means do not isolate reconstruction error. All methods use the same neutral prompt.

Table 19: **Novel-view synthesis on Syn4D with 4RC source geometry.** Geometry-conditioned methods receive 4RC source geometry in their native form; ReCamMaster uses no geometry. FVD uses the paired target videos. ATE and RPE-T are scaled by 10^2 and 10^3; rotations are in degrees. Bold/underline: best/second best.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	ATE ↓	RPE-T ↓	RPE-R ↓	FVD ↓
ReCamMaster	**12.40**	**0.277**	0.761	11.69	63.79	0.776	1908
TrajectoryCrafter	9.418	0.176	0.725	9.636	39.98	0.522	3073
EX-4D	9.865	0.163	0.863	18.76	89.83	2.438	2625
Gen3C	11.72	0.216	0.686	9.336	52.28	0.395	1693
Vista4D	10.98	0.158	0.668	9.972	41.56	0.419	**1525**
HOLO4D-14B	11.70	0.180	**0.667**	**3.267**	**32.09**	**0.272**	1711

Output depth. On the Syn4D test with ground-truth source geometry, depth reconstructed from generated RGB with either VGGT-Ω or 4RC is compared against exact target depth. On iPhone/DyCheck,

Table 20: **Depth reconstructed from generated videos.** Syn4D uses exact target depth; iPhone uses reconstructed target-video depth as a pseudo-reference. α: 4RC; β: two-stage source backend, without GMOS. Evaluation and aggregation are specified in Appendices F and I.

	Syn4D (exact depth)				iPhone (pseudo-reference)			
	VGGT-Ω		4RC		VGGT-Ω		4RC	
Method	**AbsRel** $\downarrow$	$\delta_1 \uparrow$	**AbsRel** $\downarrow$	$\delta_1 \uparrow$	**AbsRel** $\downarrow$	$\delta_1 \uparrow$	**AbsRel** $\downarrow$	$\delta_1 \uparrow$
ReCamMaster	0.762	0.311	0.738	0.298	0.152	0.765	0.121	0.835
TrajectoryCrafter	0.673	0.367	0.713	0.337	0.109	0.848	0.098	0.881
EX-4D	0.673	0.298	0.766	0.232	0.109	0.823	0.112	0.877
Gen3C	0.740	0.337	0.736	0.326	0.098	0.862	0.112	0.895
Vista4D	0.628	0.509	0.655	0.455	0.092	0.845	0.081	0.888
HOLO4D-14B (GT source)	**0.423**	**0.683**	**0.413**	**0.635**	–	–	–	–
HOLO4D-14B$^{\alpha}$	–	–	–	–	0.094	0.843	0.077	0.895
HOLO4D-14B$^{\beta}$	–	–	–	–	**0.076**	**0.914**	**0.064**	**0.933**

Table 21: **Source geometry and track noise on Syn4D.** The same generator receives estimated source geometry from four reconstruction backends (top; no GMOS), ground-truth tracks (middle), or ground-truth tracks with Gaussian noise whose standard deviation is α times that of the source tracks (bottom). 32 held-out pairs, 25 denoising steps. ATE and RPE-T are scaled by 10^2 and 10^3; rotations are in degrees. FVD uses an earlier 256×256 scoring and is comparable only within this table. Bold/underline rank estimated backends.

Source geometry	**ATE** $\downarrow$	**RPE-T** $\downarrow$	**RPE-R** $\downarrow$	**PSNR** $\uparrow$	**SSIM** $\uparrow$	**LPIPS** $\downarrow$	**FVD** $\downarrow$
VGGT-Ω + Open-D4RT	1.858	11.22	0.151	13.10	0.158	0.169	1337
V-DPM	1.660	8.904	**0.079**	14.96	0.268	0.151	1252
4RC	1.353	**7.148**	0.087	14.98	0.278	**0.131**	**1168**
VGGT-Ω + CoWTracker	**1.205**	7.688	0.107	**15.09**	**0.294**	0.137	1216
Ground-truth tracks	1.065	6.154	0.073	16.52	0.441	0.119	1098
GT tracks, $\alpha = 0.1$	3.033	22.53	0.221	13.44	0.171	0.172	1375
GT tracks, $\alpha = 0.5$	2.568	33.71	0.540	13.49	0.111	0.238	1739
GT tracks, $\alpha = 1.0$	2.228	31.88	0.574	13.37	0.100	0.241	1838
GT tracks, $\alpha = 4.0$	2.163	33.02	0.604	13.26	0.093	0.242	1783

each reconstructor is applied to both generated and held-out target RGB; the target reconstruction serves as a pseudo-depth reference. For both Syn4D depth evaluators, each 49-frame prediction is aligned to exact target depth by one robust median scale over the full video, with no per-frame alignment. Metrics are weighted by the number of valid pixels when aggregated over source–target pairs. Valid pixels have finite positive predicted and reference depths and pass the evaluator's confidence threshold at the fifth percentile. In Table 20, columns identify the output-depth evaluator; the iPhone HOLO4D row labels identify the source reconstructor used for generation. Both source conditions are evaluated with both output reconstructors. We define δ_1 as the fraction of pixels with $\max(\hat{d}/d, d/\hat{d}) < 1.25$.

Reconstruction backends. We hold the trained checkpoint, target trajectories, sampler, and seed fixed and replace the source backend with 4RC, V-DPM [8], VGGT-Ω + CoWTracker, or VGGT-Ω + Open-D4RT [7]. In the last two cases, VGGT-Ω supplies cameras and depth and the tracker supplies point trajectories; these Syn4D conditions use no GMOS fusion. Table 21 shows that the shared interface accepts all four backends without retraining, with different quality trade-offs. Figure 14 shows two pairs under each backend.

Track corruption. We add Gaussian noise to ground-truth tracks while keeping the other inputs fixed. Per-axis noise standard deviation is the source-track standard deviation multiplied by α. Rotation error, SSIM, and LPIPS degrade as corruption increases (Table 21). ATE jumps at $\alpha = 0.1$ but does not grow further under independent $\mathrm{Sim}(3)$ trajectory alignment, so it should not be used alone to characterize this sensitivity.



Figure 14: **Changing the source reconstructor without retraining.** The same checkpoint receives ground-truth trajectories or trajectories from four reconstruction backends, with target cameras, sampler, and seed fixed (Table 21). Geometric errors change the generated layout and appearance, as seen with V-DPM and Open-D4RT.

J Conditioning Cost, Runtime, and Memory

Source geometry and query-frame appearance are reusable across target trajectories. Target projection, depth, validity, scattering, and camera features must be recomputed. Training uses offline geometry annotations but still evaluates the learned adapters.

Channel-wise geometric conditioning. For hidden features $\mathbf{h} \in \mathbb{R}^{L\times D}$ and track features $\mathbf{T} \in \mathbb{R}^{L\times 156}$, each injector projects $[\mathbf{h}; \mathbf{T}] \in \mathbb{R}^{L\times(D+156)}$ back to $\mathbb{R}^{L\times D}$. Self-attention therefore keeps the same sequence length, avoiding Vista4D's additional rendered-video tokens [11]. Source-video tokens, adapters, and per-block projections still incur cost; fewer tokens do not imply a proportional reduction in total memory.

Table 22: **Warm inference runtime on one H200.** Times are mean ± standard deviation over four videos, including reconstruction, proxy rendering, and generation but excluding model loading and video writing.

Stage	Time (s)	Share
4RC feed-forward	1.46 ± 0.02	1.5%
Static cache + rendering	7.54 ± 0.30	7.8%
Dynamic render + fusion	1.43 ± 0.31	1.5%
3D-track encoding	0.06 ± 0.00	0.1%
Denoising + decoding	86.50 ± 0.26	89.2%
Warm-pipeline total	$\mathbf{96.99 \pm 0.67}$	**100%**

H200 stage-wise profile. Table 22 profiles step 19,500 on one H200 NVL at $288 \times 512 \times 49$, with 25 steps and CFG 5. Four videos after one warm-up average 96.99 ± 0.67 s, excluding model loading and MP4 writing. The 1.46 s 4RC entry is fresh reconstruction timed by its internal profiler, not a cache read. The harness also performs proxy rendering (8.97 s for the static cache and dynamic render rows), although the generator uses only track conditioning. Preprocessing totals 10.49 s; denoising and VAE decoding take 86.50 s.

Source reconstruction cost. Table 23 uses separate backend profiles. The warm A6000 runs average ten 49-frame ADT clips after one warm-up, excluding model loading. Unlike the internally timed H200 stage, the 8.8 s 4RC measurement includes video decoding, native preprocessing, and inference. Cold runs include loading and query dense tracks at the first and last frames. Warm memory is PyTorch peak allocation; cold memory is `nvidia-smi` device occupancy. The different inputs, hardware, and timing boundaries preclude a matched ranking.

Matched sampling time. Table 24 compares two native Vista4D trajectories at $384 \times 672 \times 49$, using 50 steps on one H200. Each video has $N = 13{,}104$ latent tokens (13 latent frames of 24×42 patches); Vista4D attends over source, rendered, and target videos, HOLO4D over source and target. Sampling-loop averages include classifier-free guidance but exclude source reconstruction and point-cloud

Table 23: **Reconstruction cost per 49-frame source.** Warm/cold exclude/include model loading. Open-D4RT supplies tracks only.

Backend	Mode	GPU	Time (s)	GiB
4RC	warm	A6000	8.8	16.0
VGGT-Ω + CoWTracker	warm	A6000	11.8	25.2
VGGT-Ω + CoWTracker	cold	A6000	66.6	25.1
Open-D4RT	cold	A6000	168.4	5.9
V-DPM	cold	H100	139.4	53.5

Table 24: **Sampling cost on one H200.** Both 14B models use $384 \times 672 \times 49$ and 50 denoising steps. $N = 13{,}104$ tokens per video.

Metric	Vista4D	HOLO4D
Tokens per DiT layer ↓	$3N$	$\mathbf{2N}$
Step time (s) ↓	9.89	**7.39**
Total time (s) ↓	581.9	**570.9**
Peak memory (GiB) ↓	49.86	**49.14**

rendering: 7.39 s per step for HOLO4D versus 9.89 s for Vista4D. Including loading, conditioning, decoding, and writing, full-run times are 570.9 and 581.9 s. The remaining stages jointly account for approximately 201 and 87 s; their individual contributions were not profiled. Peak memory is similar (49.14 and 49.86 GiB). Training-step cost was not measured.

K Additional Qualitative Results

Figure 15 shows HOLO4D on six further trajectories from Vista4D's benchmark and six from DAVIS, covering lateral, orbit, crane, dolly, and zoom motions; overlaid trajectories indicate the requested image-space paths for visual comparison with the generated motion. These examples complement the DAVIS and held-out iPhone comparisons in the main paper.

Figure 16 compares all methods on two further trajectories in the layout of Figure 3.

Robot videos. Figure 17 applies HOLO4D-14B, trained without robot videos, to DROID manipulation episodes and humanoid-robot clips without fine-tuning, using a scaffold reconstructed from each source video. Figure 18 shows four further robot-video rollouts using a continuation model trained with OmniWorld-DROID (Appendix B).

Remaining failure modes. Fine dynamic structures such as hands can remain inconsistent across views, and large disoccluded regions are completed by the generative prior without geometric support, so their appearance can be implausible. The conditioning does not remove these failures; we expect stronger base models to reduce them while keeping the same interface.

Coverage of the initial scaffold. With first-frame queries, objects that enter the source video later may have no corresponding trajectories. The generator still receives their appearance in the source video, but must infer their target-view placement without direct geometric support. Figure 19 illustrates this case for a person entering the DAVIS *india* sequence. Because changing the camera can change when a person becomes visible, this example illustrates missing scaffold coverage rather than a measured timing error.

L Robot Policy Learning with Fast-WAM

Fast-WAM [65] couples a Wan2.2-TI2V-5B video expert with an action expert. Both arms share the Wan2.2-TI2V-5B weights, VAE, action-expert initialization, and cached text features. The HOLO4D arm replaces 30 video self-attention blocks (300 tensors) with those from a 5B image-to-video variant at step 29,500, warm-started from a video-to-video HOLO4D-5B model. This is distinct from the step-29,000 no-identity variant in Table 12; its pretraining mixture includes OmniWorld-DROID robot videos. Track, camera, and domain modules are discarded because Fast-WAM lacks them. Both arms train the full model, including the transferred blocks, for 20,000 steps on the four LIBERO [86] suites with identical data order, normalization, seed, and optimization settings. The official release is a separate reference. All checkpoints are evaluated with the Fast-WAM closed-loop evaluator: LIBERO uses 10 tasks per suite and 50 trials per task (2,000 rollouts per checkpoint); LIBERO-plus [87] uses all 10,030 perturbed tasks with one trial each. Per-suite results are in Table 25 and are discussed in Section 4.5. On LIBERO-plus, HOLO4D improves the goal suite (+6.2) but drops 4.8 points on the spatial suite. All runs use a single seed.

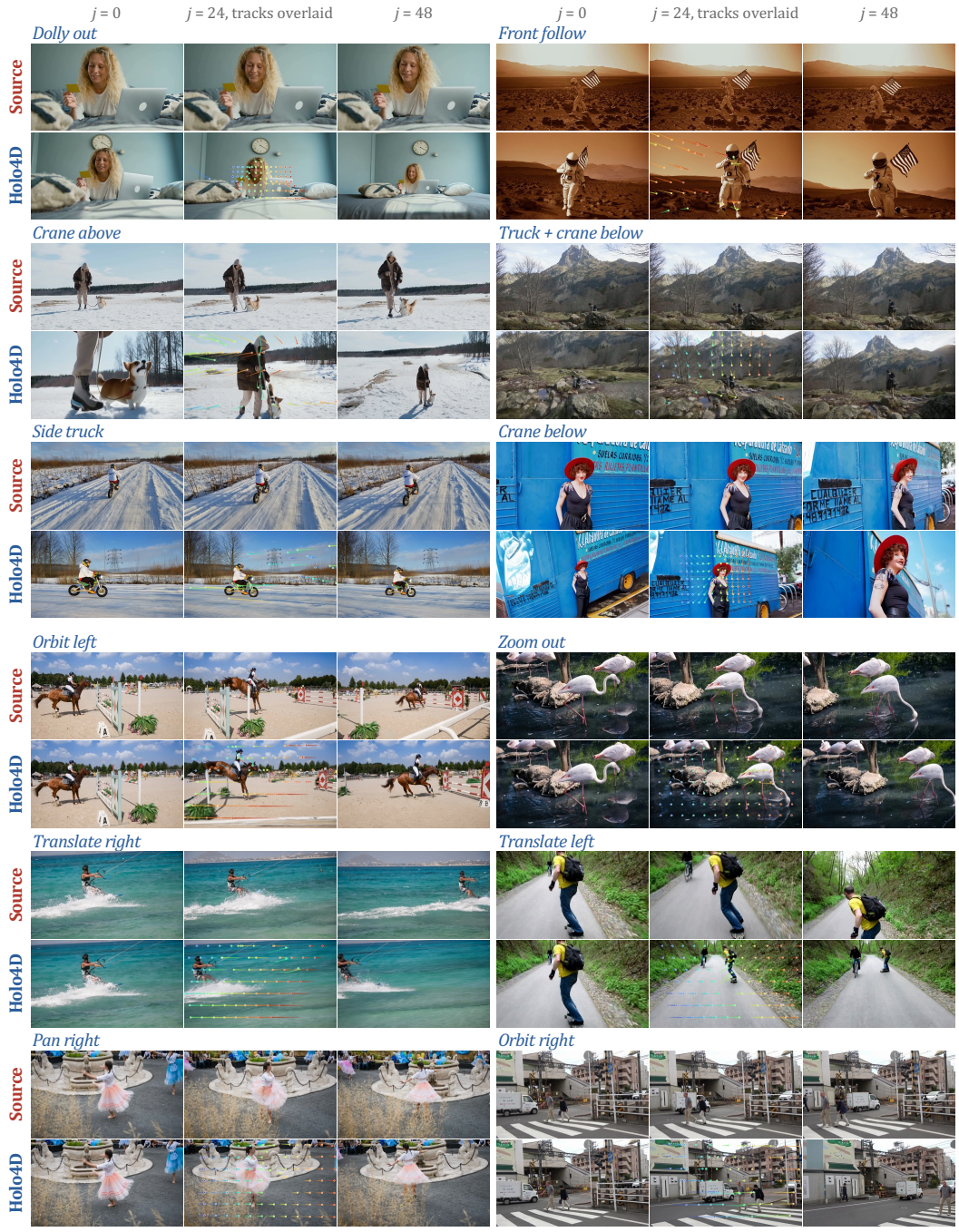

Figure 15: **Additional video reshooting examples.** Source and generated frames for six trajectories from Vista4D's benchmark (top) and six from our DAVIS evaluation set (bottom), named above each example. The middle generated frame ($j = 24$) overlays a subset of the projected trajectories, with trails covering the previous 12 frames. All examples use the two-stage backend and 50 denoising steps; none appears in the other figures.

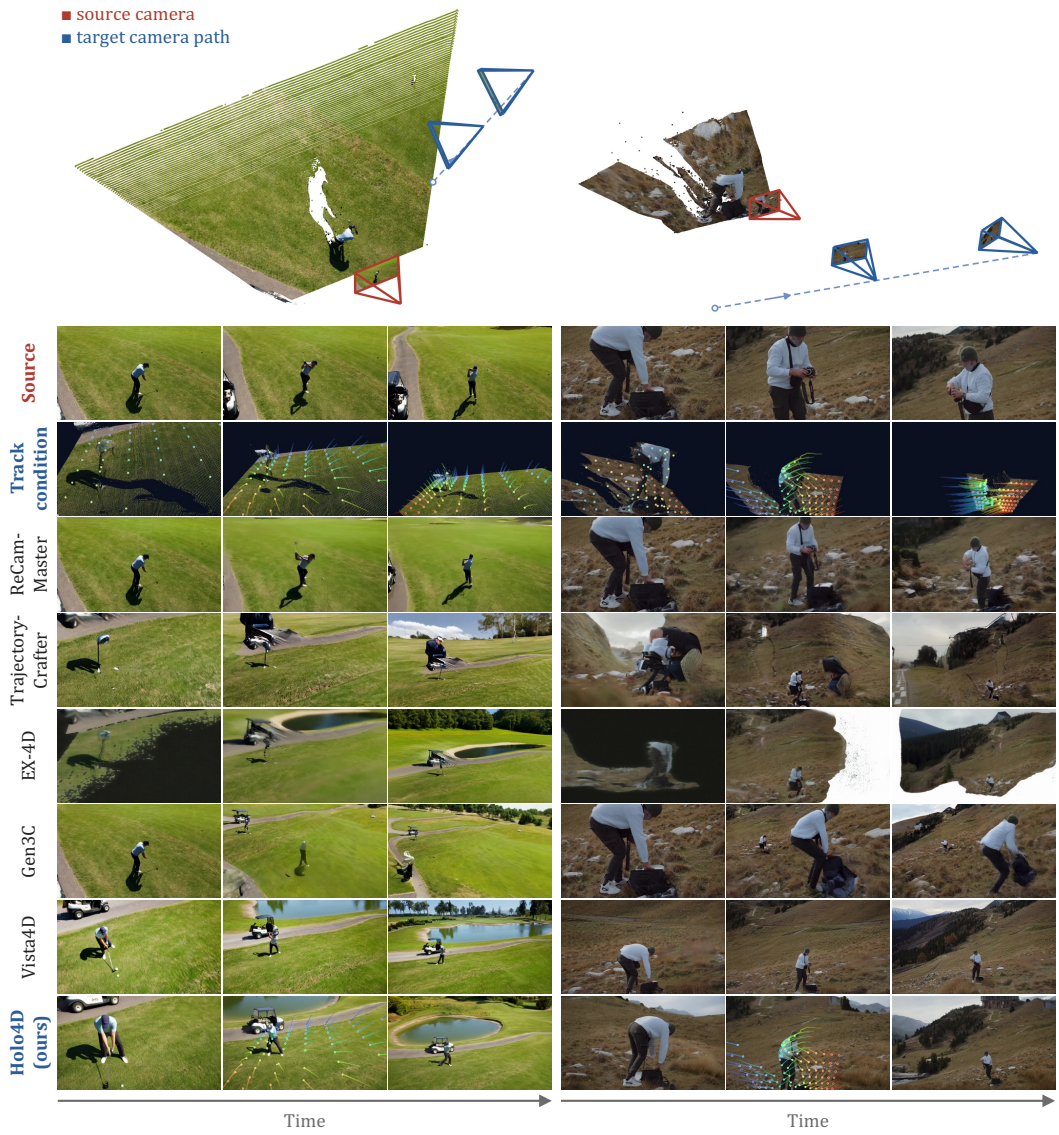

Figure 16: **Comparison with all baselines on two further Vista4D trajectories.** Left: a dolly-out from a golfer that reveals a golf cart; right: a rising, zooming-out crane shot of a man on a hillside. Layout as in Figure 3. TrajectoryCrafter, EX-4D, and Gen3C render the cart incomplete or distort the people, EX-4D leaves large holes, and ReCamMaster stays close to the source framing; Vista4D and HOLO4D both follow the requested paths. HOLO4D-14B uses 4RC source reconstruction and 50 denoising steps; baselines use the same settings as in Figure 3.

M Broader Impacts

Controllable novel-view video synthesis can support content creation, immersive media, simulation, and the construction of multi-view robot data. It can also fabricate plausible but incorrect views of real events. Source-video and geometric conditioning constrain generation but do not guarantee factual fidelity, especially in disoccluded regions. Track validity is an observation-support signal, not a forensic test. Deployment should distinguish generated views from recorded footage and respect the base model's license and safety requirements.



Figure 17: **Zero-shot novel-view synthesis of robot videos.** Each example shows the source video (top), projected 3D trajectories with source colors (middle; dark regions have no support, and trails show a subset over the preceding 12 frames), and the generated view (bottom). Top: DROID [85] episodes with 4RC reconstruction; bottom: humanoid clips with the two-stage backend. HOLO4D-14B uses no robot training data and no fine-tuning; 50 denoising steps.

N Existing Assets and Licenses

The code (training and inference), pretrained models, the predictions of all evaluated methods, and the evaluation code are released at `https://github.com/NIRVANALAN/Holo4D`. External assets retain their original licenses and terms. Table 26 lists the licenses stated by their official repositories and release cards, distinguishing code, data, and model weights. Third-party footage and component assets retain their own terms. Training datasets and baseline backbones are listed in Tables 9 and 10. Camera-adherence evaluation uses DAVIS [80]; real novel-view evaluation uses iPhone/DyCheck [81]. Pretrained components include the Wan 2.1 backbone [70], the umT5-XXL text encoder [93], DepthAnything 3 [4], and 4RC [6]. Reconstruction and preprocessing also use



Figure 18: **More long-video rollouts.** From top: three DROID [85] episodes of 196, 214, and 246 frames (5, 5, and 6 windows) reshot to static target views, and a 94-frame humanoid clip (2 windows) reshot along a close crane-above path. Six evenly spaced frames per video. Window 1 uses HOLO4D-14B with 4RC (DROID) or the two-stage backend (humanoid); later windows use the I2V continuation model, trained with OmniWorld-DROID, and 4RC; 50 steps (Appendix B).



Figure 19: **Content outside the initial scaffold.** The marked person enters the source view late and has no first-frame query points, so generated views must complete them without trajectory support (no target-view ground truth is available).

Table 25: **Fast-WAM details.** Closed-loop success rates (%) per suite on LIBERO (500 trials per suite) and LIBERO-plus (one trial per task); *All*: suite mean (LIBERO), task-weighted mean (LIBERO-plus). Official: released Fast-WAM checkpoint; Ctrl.: Wan2.2-initialized control; Ours: HOLO4D self-attention initialization.

	LIBERO			LIBERO-plus			
Suite	**Official**	**Ctrl.**	**Ours**	**Tasks**	**Official**	**Ctrl.**	**Ours**
Spatial	97.40	97.00	**97.60**	2,402	54.58	**59.87**	55.08
Object	**99.40**	99.20	99.20	2,518	**69.46**	62.43	65.45
Goal	**96.40**	95.20	96.00	2,591	37.32	38.67	**44.89**
LIBERO-10	**94.00**	91.00	93.00	2,519	42.83	43.03	**43.35**
All	**96.80**	95.60	96.45	10,030	50.91	50.81	**52.10**

VGGT-Ω [5], CoWTracker [18], GMOS [83], V-DPM [8], and Open-D4RT [7]. Evaluation uses Pi3 [28], VBench [84], held-out Syn4D scenes, and Vista4D's released trajectory benchmark [11]. The 5B models use Wan2.2-TI2V-5B, the long-video continuation model uses Wan2.1-I2V-14B, and both additionally train on OmniWorld-DROID [92, 85]. Robot reshooting uses DROID [85] and publicly available online videos of Figure and Tesla Optimus humanoid robots, shown for qualitative illustration only; policy experiments use Fast-WAM [65], LIBERO [86], and LIBERO-plus [87]. SAM3 [94] supplies precomputed MultiCamVideo foreground masks.

Table 26: **External asset licenses.** Asset names link to official license notices or release cards. Code licenses do not automatically cover separate weights or source footage.

Asset	License and scope
Training and evaluation data	
Syn4D	CC BY 4.0; underlying assets retain their individual terms.
MultiCamVideo, DyCheck iPhone	Apache 2.0 (dataset releases).
DL3DV-10K	CC BY-NC 4.0 and DL3DV-10K Terms of Use.
MVS-Synth	Custom terms: research and educational use only.
Kubric, RealEstate10K	Apache 2.0 (code); rendered assets/source videos have separate terms.
TartanAir	CC BY 4.0 (official dataset terms).
DROID	CC BY 4.0 (dataset).
OmniWorld-DROID	CC BY-NC-SA 4.0 (OmniWorld release); DROID attribution also applies.
Vista4D trajectory benchmark	Apache 2.0 (release); source-video terms remain separate.
DAVIS; Figure/Optimus videos	Original source terms; no blanket footage license asserted here.
LIBERO; LIBERO-plus	MIT (LIBERO code; LIBERO-plus asset card).
Generator, reconstruction, and preprocessing	
Wan2.1, Wan2.2-TI2V-5B, umT5-XXL	Apache 2.0.
4RC	S-Lab License 1.0 (noncommercial).
VGGT-Ω, CoWTracker	FAIR Noncommercial Research License.
GMOS, SAM2, Open-D4RT	Apache 2.0; dependencies retain their own terms.
Pi3 code; Pi3 weights	BSD 3-Clause (code); BSD 2-Clause (model card).
V-DPM	MIT (code); separate checkpoint terms not verified.
Depth Anything 3 Giant 1.1	CC BY-NC 4.0 (weights).
SAM3	SAM License (custom code and model terms).
Baselines and evaluation software	
ReCamMaster, ViDAR, Fast-WAM	MIT (code); separate model/component terms apply.
TrajectoryCrafter / CogVideoX	Custom academic/research/education terms and CogVideoX License.
EX-4D, Vista4D	Apache 2.0 (code); Wan/component terms apply.
GEN3C	Apache 2.0 (code); NVIDIA Open Model License (weights).
VBench, DINO, InternVideo/ViCLIP	Apache 2.0 (repositories); auxiliary model notices remain separate.
CLIP; LPIPS	MIT (CLIP); BSD 2-Clause (LPIPS).
IQA-PyTorch / MUSIQ	PolyForm Noncommercial 1.0.0 (current code); installed-version and weight terms require separate verification.
I3D and aesthetic-head weights	Exact cached-weight license notices not verified.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and the contribution list at the end of Section 1 claim a holistic scaffold of persistent 3D trajectories that carry source appearance, channel-wise conditioning of a pretrained video diffusion model on these trajectories, and evaluations against reshooting baselines, reconstructor substitutions, trajectory perturbations, and robot videos. These claims are supported by Tables 1 to 6 and Section 4.5; the policy-learning result is reported as mixed (Table 7).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 states that HOLO4D remains sensitive to geometric error, incomplete source coverage, and difficult camera trajectories. Tables 5 and 21 quantify the sensitivity to trajectory noise; Appendix K describes remaining failure modes (fine dynamic structures, large disoccluded regions) and missing coverage for content that enters after the query frame (Figure 19); Section 4.5 states that the long-video continuation model is not zero-shot and that the policy-learning results are mixed and single-seed.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper presents an empirical method and does not introduce formal theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 defines the trajectory scaffold, its projection and encoding, and the channel-wise conditioning of the generator; Section 3.4 gives the backbone, training schedule, learning rates, and batch sizes; Appendices B and C give module dimensions, augmentation, long-video inference, dataset weights, and annotation sources; Appendices D to F and I give the evaluation sets, trajectories, and camera-recovery protocols.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We publicly release the code (training and inference), pretrained models, the predictions of all evaluated methods, and the evaluation code, with instructions to reproduce the main results, at `https://github.com/NIRVANALAN/Holo4D` (Appendix N). All training datasets (Table 9) and evaluation benchmarks are publicly available, and the data sources and evaluation protocols are described in Appendices C to F and I.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 describes baselines, evaluation sets, reconstruction backends, and inference settings; Section 3.4 and Appendices B and C give the optimizer (AdamW), learning rates, batch sizes, schedules, resolutions, data mixture, and augmentation; per-benchmark protocols are in Appendices D to F and I.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The main video-generation comparisons report point estimates, without repeated-training confidence intervals. Evaluation uses fixed benchmark pairs; valid camera-metric counts can differ by method (Appendix E). The early 1.3B conditioning study reports a confidence interval over pairs, runtime profiling reports variation across videos (Table 22), and policy learning uses a single seed (Appendix L). These do not quantify uncertainty for the main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B reports the training compute: both stages run on NVIDIA GH200 GPUs for about 24 hours each, with stage 2 on 32 GPUs using FSDP. Appendix J reports inference hardware, time, and memory per stage and per reconstruction backend (Tables 22 to 24).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly released datasets and models and publicly available robot videos under their original terms (Appendix N), involves no new human-subject data collection, and discusses misuse risks (Appendix M).

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix M discusses positive uses (content creation, immersive media, simulation, and robot data) and the risk that generated views are mistaken for recorded footage, together with mitigations.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: HOLO4D fine-tunes a publicly released video diffusion model (Wan2.1) and conditions on a given source video; it does not introduce a new high-risk pretrained model or a scraped dataset. Released checkpoints will follow the base model's license and usage terms, and Appendix M discusses the risk that generated views are mistaken for recorded footage.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Every dataset, pretrained model, reconstruction method, and benchmark is cited at first use; Appendices C and N list them and state that they are used under their original licenses and terms. The humanoid-robot clips are publicly available online videos shown for qualitative illustration only.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code, checkpoints, predictions, and evaluation code are publicly released at `https://github.com/NIRVANALAN/Holo4D` (Appendix N) with documentation of the environment, training and inference commands, and data layout, under the base model's license.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work involves no crowdsourcing or newly recruited human participants. It uses existing datasets and publicly available robot videos (Appendices C and N).

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not part of the core methodology of HOLO4D. The text encoder used by the diffusion backbone (umT5-XXL) is a standard component inherited from the Wan 2.1 base model and serves only as a frozen text-conditioning module rather than as an original methodological contribution.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.