LEARNING THE 4D WORLD THROUGH POINT TRAJECTORY FORECASTING

Byungwoo Jeon[1*] **Sunghwan Hong**[2*] **Dohyeon Kim**[3] **Jonghoon Lee**[1]
Jungwoo Park[1] **Inhee Lee**[3] **Doosan Baek**[6] **Jinil Kim**[6] **Taehyun Lim**[6]
Hanbyul Joo[3,5] **Marc Pollefeys**[2] **Saining Xie**[4†] **Jinwoo Shin**[1,5†]
[1]KAIST [2]ETH Zürich [3]Seoul National University [4]New York University
[5]RLWRLD [6] Physics Sim Lab

Project Page **VPM-Data** **VP-WorldBench**

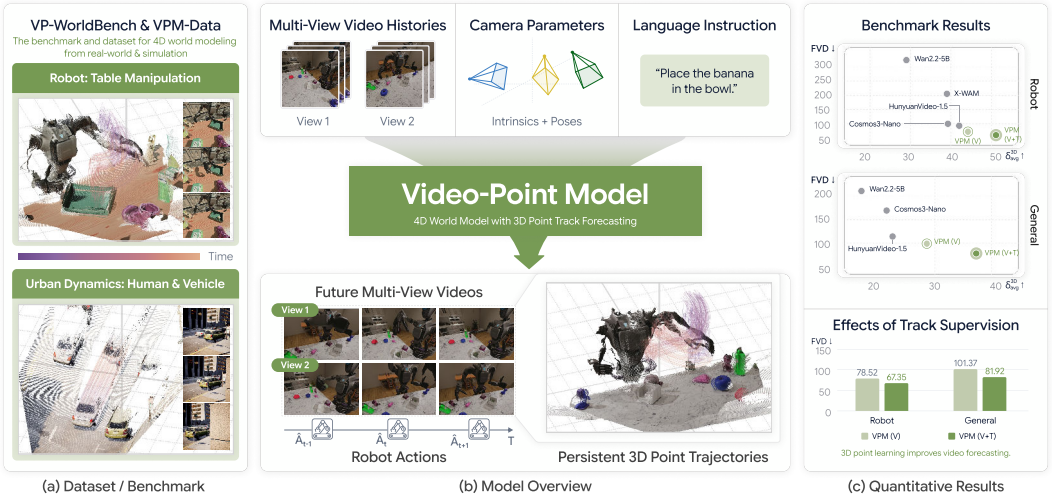


Figure 1: **Teaser.** We introduce simulated and real-world multi-view videos paired with 3D point tracks (left), jointly forecast future videos and point trajectories (center), and show that point supervision improves visual forecasting (right).

ABSTRACT

Reliable multi-view forecasting requires videos from different cameras to describe the same evolving 3D scene. To this end, we introduce Video-Point Model (VPM), a world model that conditions on multi-view video histories and a language instruction to jointly forecast future videos from arbitrary viewpoints and the 3D trajectories of persistent scene points. We extend a pretrained world model with camera-conditioned multi-view processing that integrates complementary observations across cameras, and add a trajectory head that predicts how physical points move through time. Each trajectory tracks a single point across all views, and supervising these trajectories alongside video generation therefore anchors appearance changes in every camera to a shared description of 3D motion. To support training and evaluation, we build VPM-Data, a simulation corpus pairing synchronized multi-view videos with persistent point trajectories, and VP-WorldBench, a forecasting benchmark spanning simulated and real environments. We further extend VPM to robot control by adding robot actions as a token modality in the generator. Action tokens share the point-supervised backbone and are denoised jointly with future videos, thereby grounding the policy in the same 3D-aware representation. On VP-WorldBench and existing 4D generation benchmarks, point supervision improves both video fidelity and 3D motion accuracy over a video-only counterpart, and in simulated and real-robot experiments, it also raises the manipulation success of the action-enabled model.

[*]Equal contribution.
[†]Equal advising.

1 INTRODUCTION

World models aim to anticipate how scenes evolve from past observations. Recent models pursue this goal by predicting future videos and, for robot interactions, the associated actions (Agarwal et al., 2026). However, visual plausibility alone does not establish whether these forecasts describe coherent 3D motion. Multi-view histories provide richer evidence for predicting this motion by revealing scene structure and ongoing dynamics that may be obscured in individual cameras (Rajič et al., 2025). To exploit this evidence, a model must relate observations across cameras to a common prediction of scene evolution. Its forecasts from arbitrary viewpoints should therefore depict the same future event, despite differences in perspective and visibility.

To connect these predictions, we build on geometric supervision (Lee et al., 2026b) for video modeling, including pointmap prediction and trajectory-derived representation alignment (Liu et al., 2026; Yang et al., 2026). Specifically, we investigate jointly forecasting future images and persistent 3D point trajectories. Because each trajectory follows the same scene point over time, it provides a common description of motion across cameras. Learning this motion alongside image generation encourages the model to relate appearance changes to the movement of identifiable points. Unlike conventional tracking, both predictions extend beyond the observed history, requiring the model to anticipate correspondences without access to future frames. Our hypothesis is that learning where scene points will move helps the visual predictor produce more faithful forecasts.

Accordingly, we introduce **VPM**, a multi-view video–point world model built on Cosmos 3 (Agarwal et al., 2026). Given posed video histories and a language instruction, VPM predicts future videos from arbitrary viewpoints together with the trajectories of points selected from observed frames. To integrate information across cameras, we process their video tokens in a shared sequence, using ray-based embeddings to encode camera geometry and a view axis in positional encoding to distinguish camera streams. A trajectory forecasting head is optimized jointly with video generation, allowing explicit point-motion supervision to shape the representation used for visual prediction.

To support this learning objective, we construct **VPM-Data**, a simulation corpus covering tabletop manipulation and urban dynamics. Each sequence pairs synchronized multi-view videos with persistent point trajectories, camera geometry, visibility annotations, and language descriptions, providing aligned supervision for future appearance and motion. We complement this training corpus with **VP-WorldBench**, an evaluation suite spanning simulated and real environments. Its simulation component assesses visual and geometric forecasts on the same held-out events, allowing image quality and motion accuracy to be examined together.

We evaluate VPM on our benchmark and existing benchmarks, assessing future-video quality and 3D trajectory accuracy. Controlled ablations demonstrate that point supervision improves visual forecasting. We further investigate whether this benefit extends to robot control through an action-enabled variant that jointly learns video prediction, point trajectories, and robot commands. Across simulated benchmarks and real-robot experiments, point supervision consistently improves task success rates of the action-enabled model. Our contributions are threefold:

- We introduce VPM, a camera-conditioned model that jointly forecasts multi-view videos and persistent 3D point trajectories, using explicit point-motion supervision to improve visual prediction from arbitrary viewpoints.
- We develop VPM-Data and VP-WorldBench, providing aligned visual and geometric training data and a forecasting evaluation suite spanning simulated manipulation, urban dynamics, and real-world recordings.
- Through extensive experiments and controlled ablations, we demonstrate the benefits of point supervision for multi-view forecasting and assess its contribution to robot action learning in an action-enabled extension.

2 RELATED WORK

Geometry- and Motion-Aware Video Prediction. Auxiliary objectives improve video generation by supervising internal features with 2D point tracking (Jeong et al., 2025; Lee et al., 2026a) or optical flow (Chefer et al., 2025), or by aligning them with video and 3D foundation models (Zhang

et al., 2025; Wu et al., 2026; Han et al., 2026). For multi-view prediction (Balice et al., 2026), SV4D (Xie et al., 2025) synthesizes novel views of observed videos, Geometry-aware 4D Video Generation (Liu et al., 2026) forecasts RGB and pointmaps, and X-WAM (Guo et al., 2026) adds actions to multi-view RGB-D forecasting, while GEM-4D (Zhou et al., 2026b), MECo-WAM (Zhang et al., 2026c), and 4D-WAM (Yang et al., 2026) distill geometry or trajectory fields into video and video–action representations. Unlike these single-view or per-frame targets, our persistent 3D trajectories retain point identity across views and time. Closest to ours, MVTrack4Gen (Lee et al., 2026b) supervises novel-view generation with multi-view point tracking; we differ from this in that we *forecast* beyond observed frames, lift supervision from 2D to metric 3D, and extend it to action learning.

Video World Models and Action Learning. Video foundation models such as HunyuanVideo-1.5 (Wu et al., 2025) and Wan2.2-TI2V-5B (Team et al., 2025) provide pretrained representations for visual prediction. VideoVLA (Shen et al., 2026) and Cosmos 3 (Agarwal et al., 2026) jointly predict future observations and actions, while SpatialVAM (Li et al., 2026b) pairs multi-view RGB prediction with spatial action heatmaps. We instead supervise multi-view forecasting with future 3D point motion and examine whether it also benefits action learning.

3D Motion Forecasting and Benchmarks. Point trackers estimate correspondences within observed videos (Karaev et al., 2025), and TAPIP3D (Zhang et al., 2026a) and MVTracker (Rajič et al., 2025) extend tracking to 3D from monocular and multi-view input. MotionForesight (Bharadhwaj & Jangir, 2026) and MolmoMotion (Zhang et al., 2026b) instead forecast motion beyond observed frames, the latter also supporting video generation and robot learning. Among benchmarks, TAPVid-MV (Koppula et al., 2026) provides multi-view tracking annotations and PointMotionBench (Zhang et al., 2026b) targets language-conditioned motion prediction; ours adds paired visual and geometric supervision for jointly evaluating future appearance and point motion.

3 Method

3.1 Problem formulation

Let $\mathcal{V}_{\text{in}}$ and $\mathcal{V}_{\text{out}}$ denote the input and output camera sets. For camera v at time τ, let $\mathbf{O}_\tau^v \in \mathbb{R}^{H\times W\times 3}$, $\mathbf{K}^v$, and $\mathbf{T}_\tau^v \in \mathrm{SE}(3)$ denote its RGB frame, intrinsics, and camera-to-world pose. Given the observed histories $\{\mathbf{O}^v_{t-h'+1:t}\}_{v\in\mathcal{V}_{\text{in}}}$, the camera parameters of the observed and requested future views, and a language instruction ℓ, the model predicts the next h frames at arbitrary output viewpoints and the future trajectories of N points selected from the observed frames. Each point n is specified by a query $(t_n, v_n, \mathbf{u}_n)$ within the history, and $\mathbf{P}_{\tau,n} \in \mathbb{R}^3$ denotes its persistent 3D position. Trajectories are expressed in a fixed reference frame defined by camera v_{ref} at the first observed timestep $t_0 = t - h' + 1$. Our architecture overview is shown in Figure 2.

3.2 Preliminaries: Mixture-of-Transformers for World Modeling

We build on Cosmos 3 (Agarwal et al., 2026), whose Mixture-of-Transformers backbone comprises a reasoner and a generator. The two towers are initialized from a pretrained vision-language model and maintain separate layer parameters, interacting through a joint attention operation. In the forecasting configuration, the reasoner processes instruction tokens with causal attention restricted to its own sequence, while generator tokens attend to both sequences.

The generator is trained with rectified flow matching (Liu et al., 2022; Lipman et al., 2022). Given clean future-video tokens $\mathbf{x}_0$ and Gaussian noise $\boldsymbol{\epsilon}$, we construct $\mathbf{x}_\sigma = (1-\sigma)\mathbf{x}_0 + \sigma\boldsymbol{\epsilon}$ at noise level $\sigma \in [0,1]$. The denoiser v_θ predicts $\boldsymbol{\epsilon} - \mathbf{x}_0$, conditioned on the noise level and context c:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{\mathbf{x}_0,c,\boldsymbol{\epsilon},\sigma}\left[\|v_\theta(\mathbf{x}_\sigma,\sigma,c) - (\boldsymbol{\epsilon}-\mathbf{x}_0)\|_2^2\right]. \tag{1}$$

Here, c contains the observed history and conditioning information. Clean conditioning tokens are excluded from the generation loss.

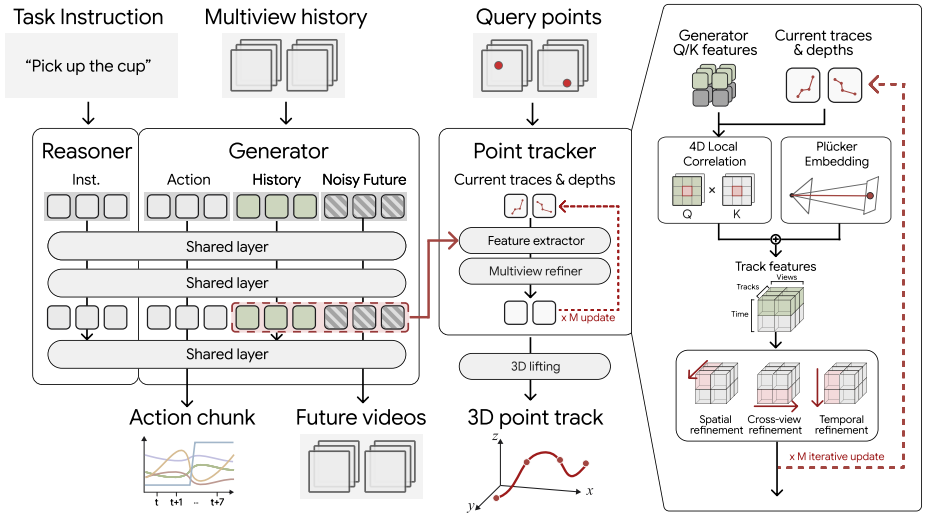


Figure 2: **Overview of VPM.** From posed multi-view histories and a task instruction, VPM forecasts future videos and the trajectories of points queried from observed frames. The 3D point tracker iteratively refines point positions and depths before lifting them to 3D. Joint point supervision shapes the generator's representation, and the action-enabled extension also outputs robot action chunks.

3.3 Camera-Conditioned Multi-View Prediction

We pack latent video tokens across cameras into a single sequence on a shared temporal grid. Observed history tokens provide context, while noisy future tokens are associated with the requested output cameras. To distinguish camera geometry from view identity, we introduce per-token camera embeddings and an additional view axis in rotary positional encoding.

Camera embeddings. For each video token i, we compute the Plücker ray (Sitzmann et al., 2021; He et al., 2024) of its image region using the corresponding intrinsics and pose. Expressed in the fixed reference frame, the ray is $\mathbf{r}_i = (\mathbf{d}_i, \mathbf{o}_i \times \mathbf{d}_i)$, where $\mathbf{o}_i$ and $\mathbf{d}_i$ are its origin and unit direction. A lightweight encoder produces $\mathbf{e}_i = \phi(\mathbf{r}_i)$, which conditions the backbone through additive features and camera-dependent rotations of attention queries and keys:

$$\begin{aligned} \mathbf{h}_i &\leftarrow \mathbf{h}_i + \mathbf{W}\mathbf{e}_i, \\ \mathbf{q}_i &\leftarrow \mathbf{R}\big(\psi(\mathbf{e}_i)\big)\mathbf{q}_i, \qquad \mathbf{k}_i \leftarrow \mathbf{R}\big(\psi(\mathbf{e}_i)\big)\mathbf{k}_i. \end{aligned} \tag{2}$$

Here, $\mathbf{W}$ is a learned projection and $\mathbf{R}(\cdot)$ applies per-head rotary phases predicted by ψ. These query and key rotations are applied in each transformer layer (Park et al., 2026), allowing attention to depend on the camera geometry associated with its tokens.

4D RoPE. Tokens from different cameras can share the same temporal and spatial coordinates. Standard 3D rotary positional encoding therefore does not distinguish their view identities. We add a fourth axis, assigning $p_{\text{view},i} = v$ to tokens from camera v. The rotary phase becomes

$$\theta_{i,k} = p_{t,i}\omega_k^t + p_{h,i}\omega_k^h + p_{w,i}\omega_k^w + p_{\text{view},i}\omega_k^{\text{view}}, \tag{3}$$

where ω^t, ω^h, and ω^w are the pretrained frequencies, and ω^{view} is learnable. Initializing the view frequencies to zero recovers the original 3D positional encoding at initialization. This formulation preserves the temporal and spatial indexing within each view while accommodating different numbers of camera streams.

3.4 Joint Video and Trajectory Learning

We connect point-motion supervision to video prediction in two ways. A trajectory head forecasts point locations from intermediate generator features, while a correspondence objective directly supervises the attention features from which those trajectories are inferred. Both objectives provide geometric learning signals to the shared video representation.

Trajectory forecasting head. The head operates on query and key features from an intermediate self-attention layer, reshaped into per-view feature grids $\mathbf{F}_q^v$ and $\mathbf{F}_k^v$ over the latent frames. For each query $(t_n, v_n, \mathbf{u}_n)$, it samples a source feature and computes correlations with key features around the current point estimate in each view and timestep. A transformer with attention factorized over space, time, and views (Koo et al., 2026) refines the estimates over M iterations. At each latent timestep, the head predicts a 2D position $\widehat{\mathbf{u}}_{\tau,n}^v$, depth $\widehat{d}_{\tau,n}^v$, and confidence $\widehat{c}_{\tau,n}^v$ for every view.

We unproject each estimate into the fixed reference frame. Let $\mathbf{G}_\tau^v = (\mathbf{T}_{t_0}^{v_{\text{ref}}})^{-1}\mathbf{T}_\tau^v$ be the camera-to-reference transform, with rotation $\mathbf{R}_\tau^v$ and translation $\mathbf{b}_\tau^v$. The corresponding 3D estimate is

$$\widehat{\mathbf{P}}_{\tau,n}^v = \mathbf{R}_\tau^v \left(\widehat{d}_{\tau,n}^v (\mathbf{K}^v)^{-1} \begin{bmatrix} \widehat{\mathbf{u}}_{\tau,n}^v \\ 1 \end{bmatrix} \right) + \mathbf{b}_\tau^v. \tag{4}$$

During training, future-frame features are obtained from partially denoised tokens rather than clean future observations. The head therefore learns to infer motion from the model's predictive features.

Trajectory supervision. Extending the multi-view tracking objective (Lee et al., 2026b), we supervise the iterative predictions with

$$\mathcal{L}_{\text{track}} = \lambda_{\text{seq}}\mathcal{L}_{\text{seq}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{conf}}\mathcal{L}_{\text{conf}}. \tag{5}$$

$\mathcal{L}_{\text{seq}}$ and $\mathcal{L}_{\text{depth}}$ apply a truncated Huber loss to 2D positions with their frame-to-frame displacements and to log-depth, respectively; truncation limits the influence of large errors from unstable refinements on heavily noised features. $\mathcal{L}_{\text{conf}}$ is a binary cross-entropy whose target indicates whether the predicted position lies within a threshold of the ground truth. All terms are summed over the M refinement iterations and averaged over visible points. The 2D and depth targets are projected from the same ground-truth 3D trajectories using known camera parameters, keeping supervision geometrically consistent across views.

Cross-view correspondence supervision. To supervise the features used for trajectory refinement directly, we treat correspondence as classification over the spatial cells of a target frame. Let $\mathbf{C}_{i,j}^{v_1,v_2}$ denote the attention map between latent frame i in view v_1 and frame j in view v_2, obtained by normalizing query–key similarities over target cells and averaging across attention heads. We minimize

$$\mathcal{L}_{\text{corr}} = -\frac{1}{|\Omega|} \sum_{(n,v_2,j)\in\Omega} \log \mathbf{C}_{t_n,j}^{v_n,v_2} \left[\mathbf{u}_n, \mathbf{u}_{j,n}^{v_2}\right], \tag{6}$$

where Ω contains query–target pairs for which the ground-truth correspondence is visible, and the bracket selects the attention weight at that correspondence. This objective encourages the generator's attention features to associate the same physical point across views and time, complementing supervision of the head's predicted positions.

3.5 Training and Inference

Training. We first adapt the generation pathway, camera embeddings, and view frequencies using $\mathcal{L}_{\text{FM}}$ on all multi-view sources, including those without trajectory annotations. Next, we train only the trajectory head and view frequencies with $\mathcal{L}_{\text{track}}$, freezing the remaining parameters. Finally, we jointly optimize the video backbone and trajectory head with

$$\mathcal{L}_{\text{forecast}} = \mathcal{L}_{\text{FM}} + \lambda_{\text{track}}\mathcal{L}_{\text{track}} + \lambda_{\text{corr}}\mathcal{L}_{\text{corr}}. \tag{7}$$

Inference. Conditioned on the observed histories, camera parameters, and instruction, we denoise future-video tokens from Gaussian noise at $\sigma = 1$ to 0, keeping history tokens clean. Trajectories are decoded from the predicted clip's backbone features and transformed into the reference frame.

4 Dataset and Benchmark

We construct **VPM-Data**, 10K training clips pairing synchronized multi-view videos with persistent 3D trajectories for tabletop manipulation and urban dynamics, and **VP-WorldBench**, an evaluation benchmark spanning tabletop, indoor, and urban scenes in simulation and the real world (Table 1).

Table 1: **Overview of VPM-Data and VP-WorldBench.**

Domain	Samples	Views	fps	Frames	3D tracks/clip	Hard
VPM-Data						
Tabletop manipulation	8,000	3	20	81–181	768–2,048	-
Urban dynamics	2,000	4	15	81–181	256	-
VP-WorldBench (Sim)						
Tabletop manipulation	500	3	20	81–680	768–1,536	27
Urban dynamics	500	4	15	81–181	256	95
VP-WorldBench (Real)						
Tabletop manipulation	100	2	15	139–325	768	-
Indoor scenes	100	48	15	150	512	-

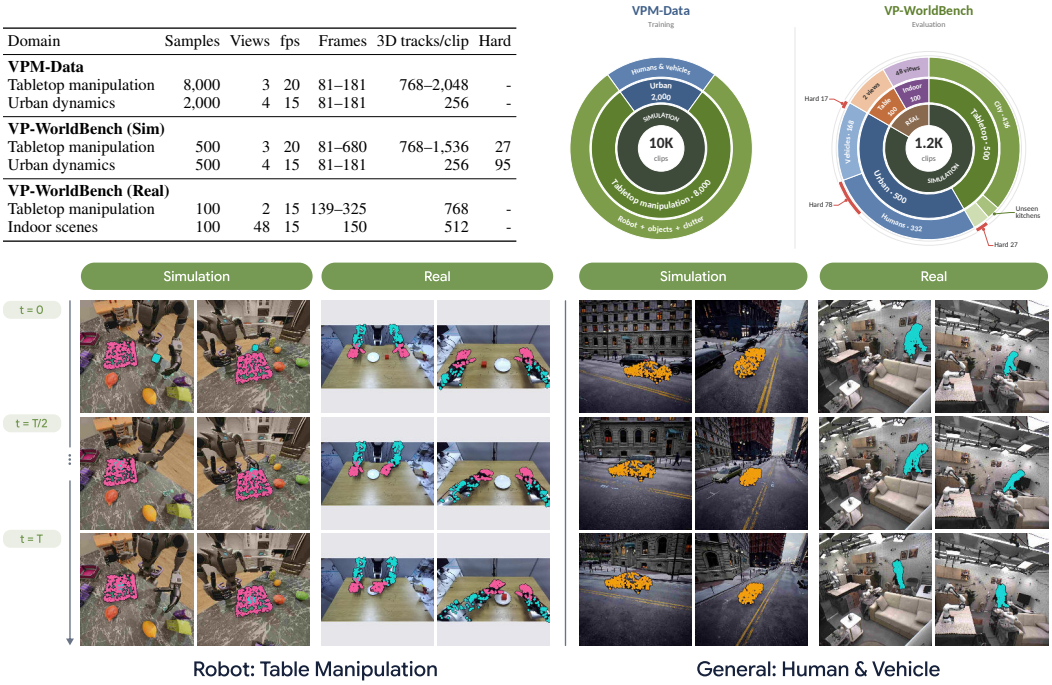


Figure 3: **Qualitative examples of VPM-Data and VP-WorldBench.**

4.1 Training Dataset

Generation and capture. Tabletop clips replay physics-simulated pick-and-place episodes of a Fourier GR-1 T2 robot in Unreal Engine 5, with randomized clutter, surface appearance, scene placement, and camera poses. Distractor interactions are validated and recorded before rendering. Urban clips capture pedestrians and vehicles from a running traffic and crowd simulation, varying their appearance, camera intrinsics, and camera paths. In both domains, images and geometric annotations are exported from the same simulation timestep to maintain alignment across views.

Annotations and validation. We annotate 256 surface points per object and 512 additional points on the robot's arms and hands. Rigid transforms or triangle identities and barycentric coordinates propagate these points through motion and occlusion. Each clip also provides metric depth, camera parameters, point-validity and per-view visibility masks, object visibility states, and a vision-language-model-generated caption. Clips undergo checks for geometric alignment, annotation completeness, depth validity, camera motion, target visibility, and caption consistency, followed by a final human audit. Generation and validation details are provided in Appendix C.

4.2 VP-WorldBench

Evaluation splits. The simulation benchmark covers three generalization regimes: 436 new tabletop episodes at varied locations in city C_1, whose scenery appears in VPM-Data only as static background; 64 tabletop episodes in two unseen kitchens; and urban clips of moving agents in C_1, whose traffic and crowd dynamics are unseen, as urban training uses a separate city C_2. Each domain includes a hard subset in which individual cameras lose sight of the target for at least 30 consecutive frames while at least one camera sees it at every frame. In tabletop hard clips, all three cameras meet this threshold and the target moves while hidden; in urban hard clips, all four cameras lose and re-acquire the target, with two or three meeting the threshold. Details are in Appendix D.

Real-world benchmarks. For robot manipulation, we collect 100 teleoperated episodes and sample 3D point tracks on the robot surface from recorded robot states (Appendix B.2). For general scenes, we record 100 ten-second sequences of people moving, handling objects, and interacting with furniture in a furnished home-like space with 48 synchronized RGB cameras. Each sequence pairs multi-view videos with a 4D reconstruction for evaluating visual forecasting and geometric ac-

Table 2: **Results on VP-WorldBench.** Numbers in parentheses denote clips per split. V and T denote video and point-trajectory supervision.

		Simulation								Real-world							
		Robot (500)				General (500)				Robot (100)				General (100)			
Method	MV	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑
HV-1.5	✗	98.34	20.21	0.023	41.62	115.63	11.45	0.061	23.10	426.76	16.04	0.102	36.17	105.83	20.00	0.014	21.35
Wan2.2-5B	✗	315.13	15.84	0.108	29.51	205.73	10.43	0.076	18.00	550.55	14.58	0.159	29.65	115.52	16.97	0.030	16.82
X-WAM	✓	203.83	19.97	0.038	38.80	-	-	-	-	492.31	17.63	0.091	50.86	-	-	-	-
Cosmos3-N	✗	104.74	20.05	0.040	39.03	167.02	10.99	0.062	22.12	326.11	16.37	0.099	41.50	66.24	20.89	0.015	20.47
VPM (V)	✓	78.52	21.41	0.022	43.62	101.37	16.95	0.057	28.71	296.83	18.42	0.074	50.83	62.85	20.96	0.010	26.54
VPM (V+T)	✓	**67.35**	**22.08**	**0.016**	**50.05**	**81.92**	**19.16**	**0.050**	**36.80**	**255.19**	**20.57**	**0.064**	**64.75**	**56.48**	**22.10**	**0.007**	**33.92**

Table 3: **Results of 4D generation on ReCamMaster and CMU Panoptic.**

		ReCamMaster (100)		CMU Panoptic (100)			
Method	MV	FVD ↓	PSNR ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑
HV-1.5	✗	183.48	14.03	352.15	21.60	0.041	31.67
Wan2.2-5B	✗	457.16	12.10	258.06	18.65	0.061	27.88
Cosmos3-N	✗	125.70	13.99	162.36	20.72	0.060	30.12
VPM (V)	✓	103.12	15.09	150.63	20.85	0.037	34.72
VPM (V+T)	✓	**79.63**	**17.06**	**128.64**	**23.24**	**0.029**	**42.03**

Table 4: **Results of 4D generation on RoboCasa Kitchen.**

		RGB			Geometry	
Method	MV	PSNR↑	SSIM↑	LPIPS↓	MEt3R↓	δ^{3D}_{avg} ↑
DreamZero	✗	20.47	0.7612	0.1651	0.061	30.83
Robot4DGen	✓	22.05	0.8035	0.1078	0.025	46.96
X-WAM	✓	22.41	0.8729	0.0571	0.042	42.10
VPM (V)	✓	22.61	0.8752	0.0558	0.033	46.94
VPM (V+T)	✓	**22.83**	**0.8801**	**0.0537**	**0.020**	**53.70**

curacy, and passes the same quality checks as the training data. Ground-truth 3D tracks are obtained by tracking points in 2D across all views, triangulating them per frame, and keeping only tracks that are depth-consistent across views. Details are in Appendix D.7.

5 Extension: Robot Control and Joint Action Learning

To test whether point supervision also benefits control, we extend VPM with robot action generation. The action-enabled model keeps the multi-view video and trajectory objectives and additionally learns robot commands from action-annotated interactions. Following Cosmos 3, we represent a future action sequence $\mathbf{A}_t \in \mathbb{R}^{h_A \times d_A}$, with horizon h_A and dimension d_A, as an additional token modality that shares the predictive backbone with video tokens. Action learning can thus draw on representations trained with point-motion supervision.

Clean action tokens $\mathbf{a}_0$ are corrupted as $\mathbf{a}_{\sigma_a} = (1-\sigma_a)\mathbf{a}_0 + \sigma_a \boldsymbol{\epsilon}_a$, where $\boldsymbol{\epsilon}_a$ is Gaussian noise and σ_a is sampled independently of the video noise level. The shared denoiser's action output v^a_θ is trained jointly with the forecasting objective in the final training stage:

$$\mathcal{L}_{\text{robot}} = \mathcal{L}_{\text{forecast}} + \lambda_{\text{act}} \underbrace{\mathbb{E}\left[\left\|v^a_\theta(\mathbf{a}_{\sigma_a}, \sigma_a, \mathbf{x}_\sigma, c) - (\boldsymbol{\epsilon}_a - \mathbf{a}_0)\right\|_2^2\right]}_{\mathcal{L}_{\text{act}}}. \tag{8}$$

At inference, video and action tokens are initialized from noise and denoised jointly, conditioned only on the observations and instruction.

6 Experiments

We validate the effectiveness of VPM and its components by investigating the following questions:

- Can VPM improve the generation quality of multi-view videos? (Table 2, 3, 4)
- Does point supervision benefit robot control? (Table 5)
- Can explicitly learning future point motion improve world models? (Figure 4, Table 6)

6.1 Experimental Setup

Evaluation setup. We evaluate VPM on three aspects: the fidelity of generated videos, their geometric consistency, and the performance of robot policies built on the action-enabled model. For a fair comparison, no method, including VPM, receives camera parameters for future frames; thus, all methods must infer camera motion from the observed history. Additional results with camera conditioning can be found in the Appendix. For video fidelity, we report FVD and PSNR, which

Table 5: **Robot manipulation results.** A denotes action. △ denotes methods that observe multiple cameras without camera conditioning, either as separate input images or tiled into a single frame. ✓ denotes camera-conditioned multi-view generation.

Method	MV	Backbone	Simulation			Real-world		
			LIBERO	LIBERO-Plus	RoboCasa	*PnP Banana*	*PnP Cup*	*PnP Cube*
VLA-based Methods								
$\pi_{0.5}$	△	-	96.8	68.0	62.1	41.7	47.9	12.5
GR00T N1.5	△	-	86.6	66.4	65.6	-	-	-
GR00T N1.7	△	-	-	-	70.0	31.3	27.1	4.2
WAM-based Methods								
Cosmos3-Nano	△	-	96.8	81.6	72.0	50.0	54.2	41.7
DreamZero	△	Wan2.1-14B	-	-	-	54.2	68.8	45.8
X-WAM	✓	Wan2.2-5B	-	-	73.8	14.6	27.1	4.2
VPM (V+A)	✓	Cosmos3-Nano	96.8	82.0	73.4	58.3	64.6	43.8
VPM (V+T+A)	✓	Cosmos3-Nano	**97.2**	**84.6**	**76.2**	**79.2**	**70.8**	**54.2**

Table 6: **Modality-wise ablation on VP-WorldBench.**

Method	Simulation								Real-world							
	Robot (500)				General (500)				Robot (100)				General (100)			
	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑
Cosmos3-Nano	104.74	20.05	0.040	39.03	167.02	10.99	0.062	22.12	326.11	16.37	0.099	41.50	66.24	20.89	0.015	20.47
VPM (V)	78.52	21.41	0.022	43.62	101.37	16.95	0.057	28.71	296.83	18.42	0.074	50.83	62.85	20.96	0.010	26.54
VPM (V+A)	74.29	21.59	0.021	44.27	98.08	17.42	0.054	30.20	291.05	18.87	0.069	53.21	61.31	21.24	0.009	28.06
VPM (V+T)	67.35	22.08	0.016	50.05	81.92	19.16	0.050	36.80	255.19	20.57	0.064	64.75	56.48	22.10	0.007	33.92
VPM (V+T+A)	**58.97**	**22.43**	**0.015**	**53.14**	**71.23**	**20.02**	**0.048**	**38.57**	**234.59**	**21.10**	**0.058**	**66.94**	**55.48**	**22.85**	**0.006**	**37.03**

measure distribution-level and pixel-level fidelity, respectively. For geometric consistency, we use cross-view MEt3R (Asim et al., 2025) and extract 3D point tracks from the generated videos with MVTracker (Rajič et al., 2025) and compare them with ground-truth trajectories, reporting the average fraction of points within depth-adaptive thresholds (δ^{3D}_{avg}). For robot manipulation, we report task success rates (%). Results for the trajectory head's own forecasts are provided in Appendix E.4.

Baselines. We compare VPM with recent video world models, which are fine-tuned on the same data as ours for fair comparison, and world-action models: HunyuanVideo-1.5 (Wu et al., 2025), Wan2.2-5B (Team et al., 2025), Cosmos3-Nano (Agarwal et al., 2026), DreamZero (Ye et al., 2026), Robot4DGen (Liu et al., 2026), and X-WAM (Guo et al., 2026). Single-view models generate each requested view independently. We additionally include VLA models for real-world robot manipulation experiments: $\pi_{0.5}$ (Intelligence et al., 2025) and GR00T N-series (NVIDIA et al., 2025).

6.2 VP-WorldBench Results

Setup. We first evaluate VPM on VP-WorldBench, which covers the same domains as VPM-Data, robot manipulation and general scene dynamics, using held-out episodes and environments along with real-world recordings. Video fidelity metrics are computed for each generated view and averaged across views. We compare two variants: VPM (V), trained only on multi-view videos, and VPM (V+T), our default model, trained on both multi-view videos and 3D point trajectories. Both variants use the same training budget, so the comparison isolates the effect of point supervision.

Results. As shown in Table 2, VPM (V+T) achieves the best performance on every split and metric, generating videos with higher fidelity and stronger geometric consistency in both simulated and real-world scenes. Even without point supervision, VPM (V) improves over its Cosmos3-Nano backbone, showing the benefit of multi-view processing. Adding point supervision under the same training budget further improves both fidelity and 3D motion accuracy on all splits, supporting our hypothesis that learning where scene points will move yields more faithful multi-view forecasts.

6.3 Zero-Shot 4D Generation Results

Setup. To assess out-of-domain generalization, we further evaluate on three datasets: ReCamMaster (Bai et al., 2025), CMU Panoptic (Joo et al., 2015), and RoboCasa Kitchen (Nasiriany et al., 2026).

Results. As shown in Table 3, VPM (V+T) achieves the best results on both ReCamMaster and CMU Panoptic. Compared with VPM (V), point supervision improves both video fidelity and 3D motion accuracy, indicating that the generated videos depict more coherent 3D motion. The same

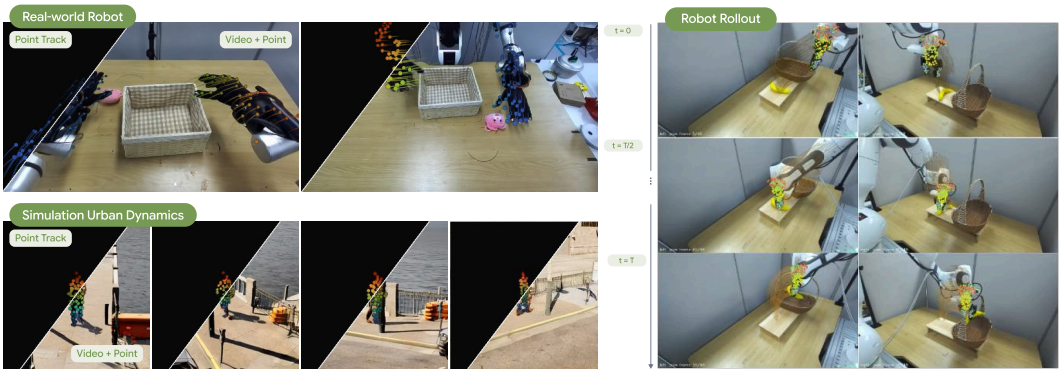


Figure 4: **Predicted 3D point trajectories guide generation and action. Left:** Predicted trajectories alone and overlaid on generated videos. **Right:** Zero-shot real-robot rollout obtained by converting predicted trajectories into commands with inverse kinematics.

trend holds on RoboCasa Kitchen (Table 4), where VPM (V+T) outperforms all baselines. Notably, VPM (V) trails the geometry-aware Robot4DGen on multi-view consistency, whereas VPM (V+T) surpasses it, suggesting that point supervision is key to the 3D consistency of generated videos.

6.4 Robot Manipulation Results

Setup. We test whether point supervision also benefits robot control using the action-enabled extension of VPM (Section 5). Simulation experiments follow each benchmark's standard protocol with three synchronized views: one wrist camera and one or two third-person cameras on either side of the robot. In the real world, we use the DROID platform (a Franka Research 3 arm with a Robotiq 2F-85 gripper) and train only on an 89K-trajectory subset of DROID dataset. We evaluate three pick-and-place tasks of varying difficulty, with 24 trials each. We provide more details in Appendix A.3.

Results. As shown in Table 5, VPM (V+T+A) achieves the best success rates on all simulation benchmarks, outperforming both VLA policies and world-action models. This advantage carries over to the real world (Table 5), where VPM (V+T+A) achieves the highest success rate on every task. In both simulated and real-world settings, VPM (V+T+A) consistently improves over VPM (V+A), which is trained without point supervision. These results indicate that forecasting persistent 3D point trajectories provides a useful learning signal for robot actions.

6.5 Analysis and Ablation Studies

Modality-wise ablation. We train VPM with combinations of video (V), point trajectories (T), and actions (A). In Table 6, adding trajectories yields the largest gains in video fidelity and 3D motion accuracy on every split, while adding actions yields modest improvements. Training with all three performs best on all splits and metrics, indicating that action supervision complements point supervision. Together with Table 5, where point supervision raises success rates on every benchmark, these results show point trajectories benefit both generation and action. We next examine why.

Why 3D point trajectory forecasting helps multi-view video generation. The trajectory head predicts point motion from the generator's intermediate features, which the remaining blocks process into future frames, so point supervision directly shapes the representation from which videos are generated. Figure 4 illustrates this coupling: the robot hands and the pedestrian closely follow the predicted trajectories overlaid on the generated frames. Since each trajectory follows a single physical point, generation that follows them remains consistent across views.

Why 3D point trajectory forecasting helps action prediction. 3D point forecasting supervision consistently improves the action-enabled model (Table 5), which we attribute to three properties. First, on robot data, trajectories are computed from joint states by forward kinematics, describing the same motion as actions but grounded in 3D and in every view, which teaches the backbone to connect visual features with where the robot will move. Second, they are dense, constraining the robot's full

future configuration with hundreds of points per timestep rather than one low-dimensional action. Third, object trajectories, where available, specify where manipulated objects should go, encoding the task's intended outcome. Action tokens, sharing this backbone, thus draw on a representation that already encodes the robot's future motion. Supporting this, converting predicted trajectories into commands via inverse kinematics, without action tokens, completes unseen real-robot tasks zero-shot (Figure 4). Though not yet consistent enough for quantitative evaluation, these rollouts show that forecast trajectories alone carry actionable information (Appendix A.4).

7 Conclusion and Limitations

We presented VPM, a multi-view world model that jointly forecasts future videos and persistent 3D point trajectories from posed video histories and language instructions. With VPM-Data and VP-WorldBench, our experiments show that supervising where scene points will move improves visual forecasting on simulated and real benchmarks and carries over to robot action learning. Limitations remain. Our point supervision covers only the robot and annotated objects rather than the full scene; skeletal annotations also omit independent cloth and hair motion. VPM assumes known camera intrinsics and poses at inference, and its cost grows with the number of views, as all camera streams share one token sequence. Future work includes deriving trajectory supervision from real multi-view videos at scale, extending it to denser, non-rigid content, and relaxing the need for calibrated cameras.

Acknowledgments

This work was partly supported by Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST); RS-2024-00509279, Global AI Frontier Lab), RLWRLD Inc., and Physics Sim Lab, Inc.; the Swiss AI Initiative through a grant from the Swiss National Supercomputing Centre (CSCS) under project ID ab036 on the Alps infrastructure; the Swiss National Science Foundation (Advanced Grant 216260: "Beyond Frozen Worlds: Capturing Functional 3D Digital Twins from the Real World"); the European Union's Horizon Europe research and innovation programme under grant agreement No. 101214398 (ELLIOT); the Horizon Europe programme under grant agreement No. 101298421 (GRAIL); and the ETH AI Center through an ETH AI Center postdoc fellowship awarded to Sunghwan Hong.

The authors thank Seonghu Jeon and Seoyeon Byun for their proofreading and assistance with improving the visualizations in the manuscript; Junbeom Kim, Minha Lee, Seong Hyeon Park, and Minsu Cho for their insightful discussions; and Beomjun Kim, Seunghoon Sim, Jaewon Youm, and Heeseung Kwon for providing feedback on model implementation and real-robot experiments.

References

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

Honggyu An, Jin Hyeon Kim, Seonghoon Park, Jaewoo Jung, Jisang Han, Sunghwan Hong, and Seungryong Kim. Cross-view completion models are zero-shot correspondence estimators. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 1103–1115, 2025.

Mohammad Asim, Christopher Wewer, Thomas Wimmer, Bernt Schiele, and Jan Eric Lenssen. Met3r: Measuring multi-view consistency in generated images. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6034–6044. IEEE, 2025.

Jianhong Bai, Menghan Xia, Xiao Fu, Xintao Wang, Lianrui Mu, Jinwen Cao, Zuozhu Liu, Haoji Hu, Xiang Bai, Pengfei Wan, et al. Recammaster: Camera-controlled generative rendering from a single video. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14834–14844. IEEE, 2025.

Matteo Balice, Yanik Kunzi, Chenyangguang Zhang, Matteo Matteucci, Marc Pollefeys, and Sunghwan Hong. No pose, no problem in 4d: Feed-forward dynamic gaussians from unposed multi-view videos. *arXiv preprint arXiv:2605.22190*, 2026.

Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Fan Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, et al. Introducing hot3d: An egocentric dataset for 3d hand and object tracking. *arXiv preprint arXiv:2406.09598*, 2024.

Homanga Bharadhwaj and Yash Jangir. Motionforesight: Re-purposing video models for future 3d scene-flow prediction. *arXiv preprint arXiv:2607.16192*, 2026.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 4724–4733. IEEE, 2017.

Hila Chefer, Uriel Singer, Amit Zohar, Yuval Kirstain, Adam Polyak, Yaniv Taigman, Lior Wolf, and Shelly Sheynin. Videojam: Joint appearance-motion representations for enhanced motion generation in video models. *arXiv preprint arXiv:2502.02492*, 2025.

Zhiang Chen, Nahyuk Lee, Boyang Sun, Taein Kwon, Marc Pollefeys, Zuria Bauer, and Sunghwan Hong. Prose: Training-free egocentric scene registration with vision-language models. *arXiv preprint arXiv:2606.16569*, 2026.

Seokju Cho, Sunghwan Hong, Sangryul Jeon, Yunsung Lee, Kwanghoon Sohn, and Seungryong Kim. Cats: Cost aggregation transformers for visual correspondence. *Advances in Neural Information Processing Systems*, 34:9011–9023, 2021.

Seokju Cho, Sunghwan Hong, and Seungryong Kim. Cats++: Boosting cost aggregation with convolutions and transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (6):7174–7194, 2022.

Yufan Deng, Zilin Pan, Hongyu Zhang, Xiaojie Li, Ruoqing Hu, Yufei Ding, Yiming Zou, Yan Zeng, and Daquan Zhou. Rethinking video generation model for the embodied world. In *Forty-third International Conference on Machine Learning*, 2026.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

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 *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8503–8513. IEEE, 2025.

Stephanie Fu, Mark Hamilton, Laura E Brandt, Axel Feldmann, Zhoutong Zhang, and William Freeman. Featup: A model-agnostic framework for features at any resolution. In *International Conference on Learning Representations*, volume 2024, pp. 45324–45350, 2024.

Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19383–19400, 2024.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4d world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.

Jisang Han, Honggyu An, Jaewoo Jung, Takuya Narihira, Junyoung Seo, Kazumi Fukuda, Chaehyun Kim, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. Dˆ 2ust3r: Enhancing 3d reconstruction with 4d pointmaps for dynamic scenes. *arXiv e-prints*, pp. arXiv–2504, 2025.

Jisang Han, Seonghu Jeon, Jaewoo Jung, René Zurbrügg, Honggyu An, Tifanny Portela, Marco Hutter, Marc Pollefeys, Seungryong Kim, and Sunghwan Hong. Geometric action model for robot policy learning. *arXiv preprint arXiv:2606.17046*, 2026.

Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024.

Sunghwan Hong and Seungryong Kim. Deep matching prior: Test-time optimization for dense correspondence. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9907–9917, 2021.

Sunghwan Hong, Seokju Cho, Seungryong Kim, and Stephen Lin. Unifying feature and cost aggregation with transformers for semantic and visual correspondence. *arXiv preprint arXiv:2403.11120*, 2024a.

Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jisang Han, Jiaolong Yang, Chong Luo, and Seungryong Kim. Pf3plat: Pose-free feed-forward 3d gaussian splatting. *arXiv preprint arXiv:2410.22128*, 2024b.

Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jiaolong Yang, Seungryong Kim, and Chong Luo. Unifying correspondence pose and nerf for generalized pose-free novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20196–20206, 2024c.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Byungwoo Jeon, Dongyoung Kim, Huiwon Jang, Insoo Kim, and Jinwoo Shin. Spatialboost: Enhancing visual representation through language-guided reasoning. In *European Conference on Computer Vision*, pp. 390–408. Springer, 2026.

Hyeonho Jeong, Chun-Hao P Huang, Jong Chul Ye, Niloy J Mitra, and Duygu Ceylan. Track4gen: Teaching video diffusion models to track points improves video generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7276–7287. IEEE, 2025.

Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE international conference on computer vision*, pp. 3334–3342, 2015.

Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labeling real videos. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1–10. IEEE, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023.

Jahyeok Koo, Inès Hyeonsu Kim, Mungyeom Kim, Junghyun Park, Seohyeon Park, Jaeyeong Kim, Jung Yi, Seokju Cho, and Seungryong Kim. Mv-tap: Tracking any point in multi-view videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20932–20941, 2026.

Skanda Koppula, Frano Rajic, Abdullah Faiz Ur Rahman, Yi Yang, Ignacio Rocco, Jeet Thakwani, Rishabh Kabra, Andrew Zisserman, Joao Carreira, Siyu Tang, et al. Tapvid-mv: A benchmark for tracking any point in 3d across multiple views. *arXiv preprint arXiv:2609.01899*, 2026.

JoungBin Lee, Jaewoo Jung, Jisang Han, Takuya Narihira, Kazumi Fukuda, Junyoung Seo, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. 3d scene prompting for scene-consistent camera-controllable video generation. In *International Conference on Learning Representations*, volume 2026, pp. 73753–73777, 2026a.

JoungBin Lee, Jaewoo Jung, Jongmin Lee, Tongmin Kim, Hyunsung Kim, Takuya Narihira, Kazumi Fukuda, Jahyeok Koo, Jisang Han, Yuki Mitsufuji, et al. Mvtrack4gen: Multi-view point tracking as geometric supervision for 4d video generation. *arXiv preprint arXiv:2606.26087*, 2026b.

Nahyuk Lee, Zhiang Chen, Marc Pollefeys, and Sunghwan Hong. Tora: Topological representation alignment for 3d shape assembly. *arXiv preprint arXiv:2604.04050*, 2026c.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pp. 71–91. Springer, 2024.

Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xiaohan Yu, Lin Gu, and Gim Hee Lee. Revisiting photometric ambiguity for accurate gaussian-splatting surface reconstruction. In *International Conference on Machine Learning*. PMLR, 2026a.

Peiyan Li, Yixiang Chen, Yuan Xu, Jiabing Yang, Xiangnan Wu, Jun Guo, Nan Sun, Long Qian, Xinghang Li, Xin Xiao, et al. Spatial-aware multi-view video diffusion as a data-efficient robot policy. *arXiv preprint arXiv:2604.03181*, 2026b.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024.

Jongbin Lim, Taeyun Ha, Mingi Choi, Jisoo Kim, Byungjun Kim, Subin Jeon, and Hanbyul Joo. Hrdexdb: A paired human-robot dataset for cross-embodiment dexterous grasping. *arXiv preprint arXiv:2604.14944*, 2026.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Zeyi Liu, Shuang Li, Eric Cousineau, Siyuan Feng, Benjamin Burchfiel, and Shuran Song. Geometry-aware 4d video generation for robot manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 140751–140773, 2026.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Byeongjun Park, Byung-Hoon Kim, Hyungjin Chung, and Jong Chul Ye. Redirector: Creating any-length video retakes with rotary camera encoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11163–11173, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

Frano Rajič, Haofei Xu, Marko Mihajlovic, Siyuan Li, Irem Demir, Emircan Gündoğdu, Lei Ke, Sergey Prokudin, Marc Pollefeys, and Siyu Tang. Multi-view 3d point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 59–68, 2025.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025.

Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. Videovla: Video generators can be generalizable robot manipulators. *Advances in neural information processing systems*, 38:95597–95621, 2026.

Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in neural information processing systems*, 34:19313–19325, 2021.

Wan Team, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2(3):4, 2025.

Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

Joachim Tesch, Giorgio Becherini, Prerana Achar, Anastasios Yiannakidis, Muhammed Kocabas, Priyanka Patel, and Michael Black. Bedlam2. 0: Synthetic humans and cameras in motion. *Advances in Neural Information Processing Systems*, 38, 2026.

Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details. *Advances in Neural Information Processing Systems*, 38:35928–35959, 2026.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709. IEEE, 2024.

Bing Wu, Chang Zou, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Jack Peng, Jianbing Wu, Jiangfeng Xiong, Jie Jiang, et al. Hunyuanvideo 1.5 technical report. *arXiv preprint arXiv:2511.18870*, 2025.

Haoyu Wu, Diankun Wu, Tianyu He, Junliang Guo, Yang Ye, Yueqi Duan, and Jiang Bian. Geometry forcing: Marrying video diffusion and 3d representation for consistent world modeling. In *International Conference on Learning Representations*, volume 2026, pp. 119186–119209, 2026.

Yiming Xie, Chun-Han Yao, Vikram Voleti, Huaizu Jiang, and Varun Jampani. Sv4d: Dynamic 3d content generation with multi-frame and multi-view consistency. In *International Conference on Learning Representations*, volume 2025, pp. 33421–33441, 2025.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

Lishan Yang, Wenxuan Song, Xi Wang, Pingyue Sheng, Zheng Fang, Ziyang Zhou, Junjie He, Haodong Yan, Jiayi Chen, Nan Sun, et al. 4d-wam: Infusing spatiotemporal awareness into world action models through trajectory fields. *arXiv preprint arXiv:2608.08023*, 2026.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Yuanwen Yue, Damien Robert, Jianyuan Wang, Sunghwan Hong, Jan Dirk Wegner, Christian Rupprecht, and Konrad Schindler. Litept: Lighter yet stronger point transformer. *arXiv preprint arXiv:2512.13689*, 2025.

Bowei Zhang, Lei Ke, Adam Harley, and Katerina Fragkiadaki. Tapip3d: Tracking any point in persistent 3d geometry. *Advances in Neural Information Processing Systems*, 38:135284–135303, 2026a.

Jianing Zhang, Chenhao Zheng, Yajun Yang, Max Argus, Rustin Soraki, Winson Han, Taira Anderson, Chun-Liang Li, Shuo Liu, Jiafei Duan, et al. Molmomotion: Forecasting point trajectories in 3d with language instruction. *arXiv preprint arXiv:2606.18558*, 2026b.

Jianjun Zhang, Jian Zhu, Taiyi Su, Chong Ma, Zitai Huang, Yi Xu, and Hanli Wang. Learning 4d geometric priors for inference-efficient world action models. *arXiv preprint arXiv:2607.05468*, 2026c.

X Zhang, J Liao, S Zhang, F Meng, X Wan, J Yan, and Y Cheng. Videorepa: Learning physics for video generation through relational alignment with foundation models (2025). *arXiv preprint arXiv:2505.23656*, 2025.

Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, December 2000. MSR-TR-98-71, Updated March 25, 1999.

Fengzhe Zhou, Jiannan Huang, Jialuo Li, Deva Ramanan, and Humphrey Shi. Pai-bench: A comprehensive benchmark for physical ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21522–21536, 2026a.

Kaichen Zhou, Yuzhen Chen, Fangneng Zhan, Hang Hua, Grace Chen, Xinhai Chang, Ao Qu, Yilun Du, Zhuang Liu, Paul Pu Liang, et al. Gem-4d: Geometry-enhanced video world models for robot manipulation. *arXiv preprint arXiv:2605.22882*, 2026b.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5738–5746. IEEE, 2019.

Appendix Contents

A Implementation Details **17**
- A.1 Training Details . . . 17
- A.2 Evaluation Details . . . 17
- A.3 Robot Manipulation Tasks . . . 19
- A.4 Details of Robot Manipulation through 3D Point Trajectory . . . 20

B Dataset Curation **21**
- B.1 Data Statistics . . . 21
- B.2 3D Point Extraction Pipeline from Robot Videos . . . 21

C Training Data Generation **23**
- C.1 Capture Settings and Environments . . . 23
- C.2 Tabletop Generation . . . 23
- C.3 Urban Capture . . . 23
- C.4 Geometric and Language Annotations . . . 24
- C.5 Visibility-Controlled Clips . . . 24
- C.6 Validation and Rendering . . . 25

D VP-WorldBench: Construction and Statistics **26**
- D.1 Composition and Capture Settings . . . 26
- D.2 Tabletop Capture . . . 26
- D.3 Urban Capture . . . 27
- D.4 Ground-Truth Annotations . . . 27
- D.5 Visibility-Controlled Subsets . . . 27
- D.6 Validation and Limitations . . . 28
- D.7 Real-World Benchmark Details . . . 29
- D.8 Relation to Existing Resources . . . 30

E Additional Experimental Results **31**
- E.1 Detailed Experiments on VP-WorldBench . . . 31
- E.2 Detailed Experiments on 4D Generation . . . 32
- E.3 Comparison with Per-Frame Geometry Supervision . . . 33
- E.4 Performance of 3D Point Trajectory Head . . . 33
- E.5 Performance on Single-View World Modeling . . . 34
- E.6 Additional Results on Modality-Wise Ablation Study . . . 35
- E.7 Analysis on Multi-View Video Generation . . . 35
- E.8 Clock-Time Analysis . . . 35
- E.9 Data Scalability . . . 36

F Detailed Ablation Studies **37**
- F.1 Ablation Study for Camera Embeddings . . . 37
- F.2 Ablation Study for Objectives . . . 37
- F.3 Ablation Study for Training Stages . . . 38
- F.4 Ablation Study for Layers . . . 38
- F.5 Ablation Study for Refinement . . . 38
- F.6 Ablation Study for Training Data . . . 39

G Qualitative Results **40**
- G.1 Multi-View Video and Point Trajectory Prediction . . . 40
- G.2 Predicted vs. Ground-Truth Videos . . . 42
- G.3 Example Rollouts of Robot Manipulation Tasks . . . 43

A Implementation Details

A.1 Training Details

We adopt Cosmos3-Nano (Agarwal et al., 2026) as the backbone and follow its Mixture-of-Transformers architecture (Liang et al., 2024). We train VPM with the three-stage pipeline described in Section 3.5.

In Stage 1, we train the generation pathway, camera embeddings, and view frequencies with $\mathcal{L}_{\text{FM}}$ on all multi-view sources, including those without trajectory annotations. In Stage 2, we freeze the remaining parameters and train only the trajectory head and view frequencies with $\mathcal{L}_{\text{track}}$. In Stage 3, we jointly optimize the video backbone and the trajectory head with $\mathcal{L}_{\text{forecast}}$. The reasoner is frozen throughout all stages. For video supervision, we treat the camera pose of each view as an optional condition. For instance, views from VPM-Data, where camera control is important, are conditioned on their future camera poses, whereas wrist-camera views from the robot datasets, whose future poses depend on the actions being predicted, are not conditioned on any pose. This also allows VPM to forecast without future camera parameters, as required by our main evaluation protocol. Detailed hyperparameters are provided in Table 7.

The trajectory head operates on query and key features from the 18th self-attention layer of the generator and performs $M = 4$ refinement iterations. During training, we sample $N = 512$ query points per clip from the observed history. For the action-enabled variants, we apply $\mathcal{L}_{\text{act}}$ in Stage 3 on action-annotated interactions. Following Cosmos 3 (Agarwal et al., 2026), unified actions are represented as 9-dimensional vectors (Zhou et al., 2019). VPM (V) and VPM (V+A) are trained without $\mathcal{L}_{\text{track}}$ and $\mathcal{L}_{\text{corr}}$. All training is conducted on 32 NVIDIA GH200 Grace Hopper GPUs.

Table 7: **Hyperparameters for each training stage.**

Hyperparameter	Stage 1	Stage 2	Stage 3
Trainable modules	Generator, Camera emb.	Traj. head	Generator, Traj. head
Objective	$\mathcal{L}_{\text{FM}}$	$\mathcal{L}_{\text{track}}$	$\mathcal{L}_{\text{forecast}}$ ($+\lambda_{\text{act}}\mathcal{L}_{\text{act}}$)
Optimizer		AdamW	
Learning rate	1e-4	1e-4	2e-5
Tokens	60,000	60,000	24,000
Training iterations	5,000	2,000	5,000
Warm-up steps		100	
Video resolution		256–480px	
Maximum frames		181	
History frames (h')		1-16	
Query points (N)	-	512	512
Refinement iterations (M)	-	4	4
λ_{seq} / λ_{depth} / λ_{conf}	-	1 / 1 / 1	1 / 1 / 1
λ_{track}	-	-	0.01
λ_{corr}	-	-	0.01
λ_{act}	-	-	0.1

A.2 Evaluation Details

Evaluation protocol. For Image-to-Video (I2V) evaluation, each model receives a single frame. For Video-to-Video (V2V) evaluation, each model receives $h' = 16$ frames of multi-view history. The model then predicts the next $h = 64$ frames and the output videos are resized to 480p. We use each model's default number of denoising steps. To evaluate single-view baselines, *i.e.*, HunyuanVideo-1.5, Wan2.2-5B, and Cosmos3-Nano, we generate videos from each view independently.

Video fidelity. We compute FVD (Unterthiner et al., 2018) using I3D (Carreira & Zisserman, 2017) features and PSNR between generated and ground-truth frames. Both metrics are computed independently for each output view and averaged across views.

Geometric consistency of multi-view videos. We measure geometric consistency with cross-view MEt3R and $\delta^{\text{3D}}_{\text{avg}}$. MEt3R (Asim et al., 2025) scores the 3D consistency of two images I_1 and I_2 without ground truth, comparing their DINO features (Oquab et al., 2023) in a common view. Chosen over different geometric foundation models (Wang et al., 2024; Han et al., 2025), MASt3R (Leroy et al., 2024) regresses point maps of both images in the camera frame of I_1, and upsampled features

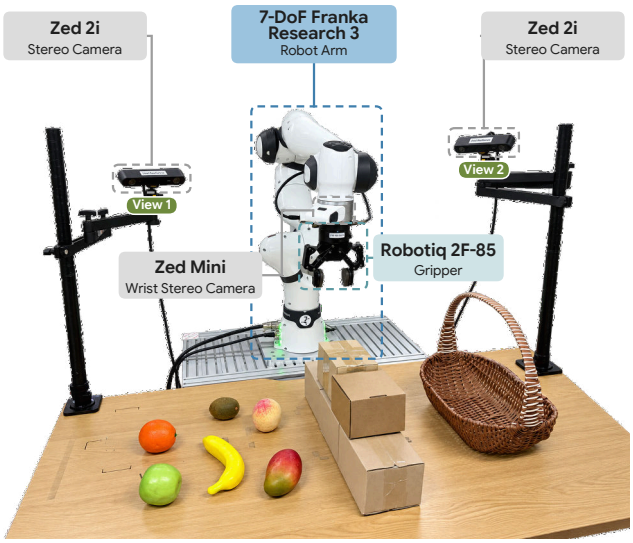


Figure 5: **Real-world hardware platform.** A 7-DoF Franka Research 3 robot arm equipped with a Robotiq 2F-85 gripper and a wrist-mounted Zed Mini stereo camera. Two external Zed 2i stereo cameras (view 1 and view 2) observe the workspace from different viewpoints, providing synchronized multi-view observations.

through FeatUp (Fu et al., 2024) attached to these points are rendered into the view of I_1, yielding $\hat{F}_1$ and $\hat{F}_2$. The score averages their cosine dissimilarity over the co-visible pixels Ω_{12}, symmetrized over the two orderings:

$$S(I_1, I_2) = \frac{1}{|\Omega_{12}|} \sum_{p \in \Omega_{12}} \big[1 - \cos(\hat{F}_1(p), \hat{F}_2(p))\big], \quad \text{MEt3R}(I_1, I_2) = \tfrac{1}{2}\big[S(I_1, I_2) + S(I_2, I_1)\big]. \tag{9}$$

However, MEt3R is typically applied to consecutive frames of a monocular video, assuming large overlap between the two images. On multi-view videos, raw MEt3R fails to rank videos by quality: real view pairs with little overlap already score high, while generated views drifting apart go unpenalized, since a pair without overlap scores 0. We thus introduce cross-view MEt3R, measuring the deviation of a generated video's consistency from that of the ground truth. For each clip s in a split $\mathcal{S}$, we average MEt3R over all pairs of output views at timesteps $\mathcal{T} = \{t + 4k\}_{k=1}^{h/4}$, obtaining $m(s)$, and report

$$\text{Cross-view MEt3R} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \big| m^{\text{gen}}(s) - m^{\text{gt}}(s) \big|. \tag{10}$$

Both videos are evaluated with the official implementation on 256×256 center crops.

For $\delta^{\text{3D}}_{\text{avg}}$, we track query points sampled from the observed history with MVTracker (Rajič et al., 2025), using the generated videos and depth maps from MoGe-2 (Wang et al., 2026), expressing the trajectories in the fixed reference frame. With $e_{\tau,n} = \|\widehat{\mathbf{P}}_{\tau,n} - \mathbf{P}_{\tau,n}\|_2$ denoting the error of point n at future timestep τ,

$$\delta^{\text{3D}}_{\text{avg}} = \frac{1}{|\mathcal{D}|} \sum_{\delta \in \mathcal{D}} \frac{1}{Nh} \sum_{n=1}^{N} \sum_{\tau=t+1}^{t+h} \mathbb{1}\left[e_{\tau,n} < \delta \, \frac{d_{\tau,n}}{f} \right], \tag{11}$$

where the pixel thresholds $\mathcal{D} = \{1, 2, 4, 8, 16\}$ are converted into 3D distances at the ground-truth depth $d_{\tau,n}$ with focal length f, following TAPVid-MV (Koppula et al., 2026).

Neither metric reaches its ideal value even for a perfect generator. For $\delta^{\text{3D}}_{\text{avg}}$, we report an upper bound, evaluating the ground-truth videos with the same pipeline. For cross-view MEt3R, a generator as consistent as the ground truth still scores above 0 when forecasting a different future, as MEt3R fluctuates over time with the content. We estimate this floor from the ground-truth videos as

$$\text{floor} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{1}{\sqrt{2}} \big| \bar{x}_s^{(1)} - \bar{x}_s^{(2)} \big|, \tag{12}$$

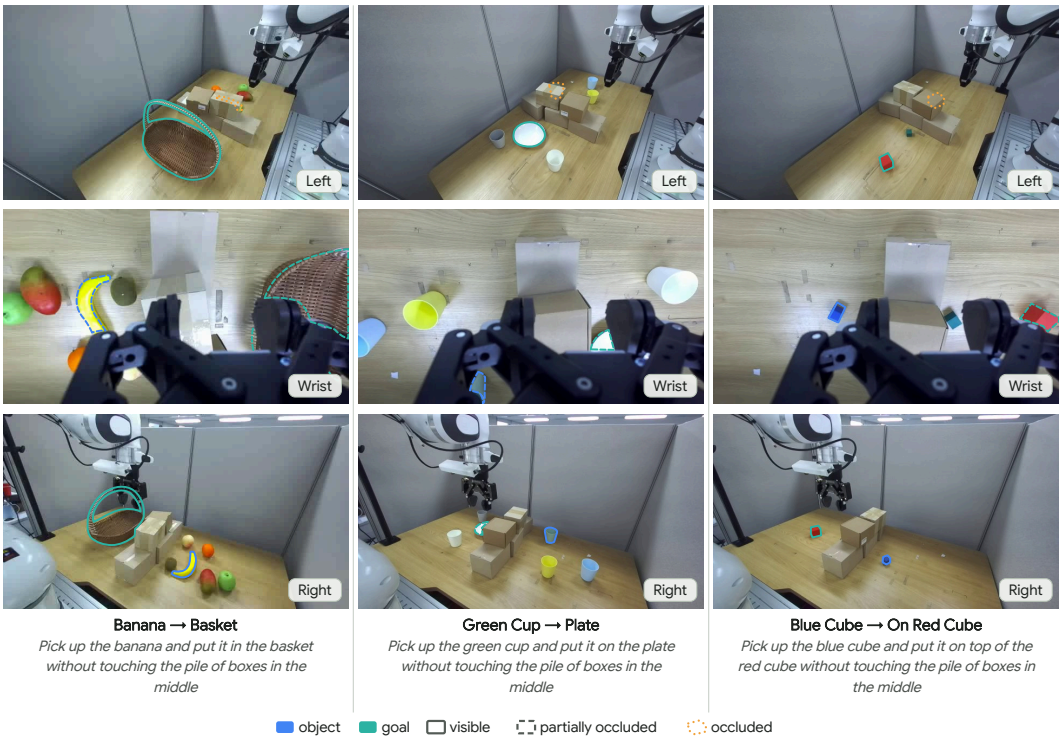


Figure 6: **Examples of the robot manipulation tasks.**

where $\bar{x}_s^{(1)}$ and $\bar{x}_s^{(2)}$ average the per-timestep MEt3R of clip s, relative to the split average, over the first and second halves of $\mathcal{T}$, with $1/\sqrt{2}$ compensating for the half-length averages.

Robot manipulation. For LIBERO (Liu et al., 2023), we evaluate on the four standard suites, *i.e.*, LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long, each consisting of 10 tasks, with 20 trials per task. For LIBERO-Plus (Fei et al., 2025), we evaluate the policy trained on LIBERO in a zero-shot manner on 10,030 tasks spanning seven perturbation dimensions with one trial per task. For RoboCasa Kitchen (Nasiriany et al., 2026), we evaluate on 24 manipulation tasks with 200 trials per task across five evaluation scenes (10 trials per scene), and average the success rates over three random seeds. For real-world tasks, we use two exo cameras and one wrist camera, and conduct 24 trials per task.

A.3 Robot Manipulation Tasks

We evaluate all policies on three pick-and-place tasks with a Franka Research 3 arm and the DROID camera setup (left and right exterior cameras and a wrist camera), as shown in Figure 5. All tasks share a tabletop layout with a pile of boxes in the middle, and the object and its goal are placed on opposite sides of the pile so that no single camera shows the whole task (Figure 6). The robot puts a banana into a basket among other fruits, places a green cup on a plate among cups of other colors, and stacks a small blue cube on top of a red cube next to a green distractor cube, which additionally requires precise placement in height. Solving these tasks requires fusing the three views into a consistent 3D understanding of the scene, moving across the pile without touching it, and grounding the instruction among same-category distractors.

Each trial is scored as 1 if the object is placed at the goal without touching the pile of boxes, 0.5 if the pick-and-place succeeds but the robot hits the pile on the way, and 0 otherwise. The success rate of a task is the average score over its 24 trials, which is why some entries in Table 5 correspond to half-integer counts.

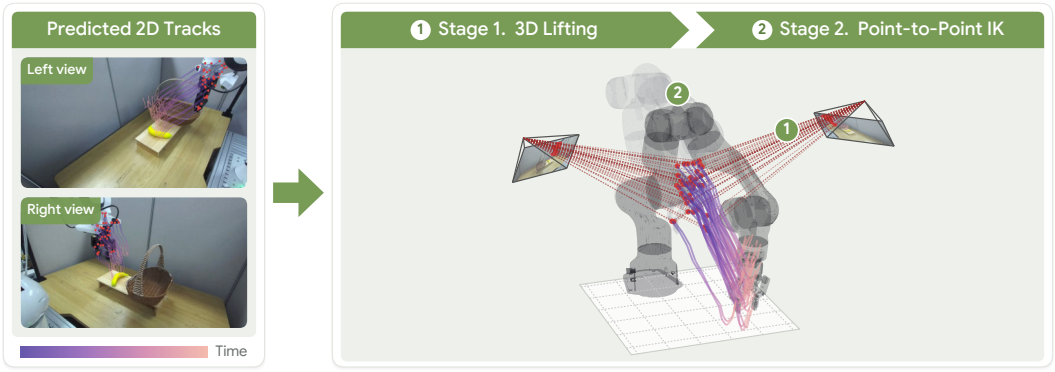


Figure 7: **From predicted point trajectories to joint-space actions.** Predicted 2D trajectories from both views are lifted to 3D (Stage 1) and converted into a joint trajectory by fitting the robot's kinematic model to the lifted points (Stage 2).

A.4 Details of Robot Manipulation through 3D Point Trajectory

To examine whether the forecast trajectories alone carry actionable information, we bypass the action tokens and convert the predicted trajectories directly into robot commands with a training-free two-stage procedure (Figure 7 and Algorithm 1). We sample $N{=}128$ query points on the robot surface from its kinematic model at the current joint state and project them into both exterior views. Their trajectories are read from the trajectory head at the third of four denoising steps ($\sigma \approx 0.83$). We choose this step because $\sigma \approx 0.83$ is the mode of the noise-level distribution used to train the trajectory head, where its predictions are most reliable. In the first stage, each trajectory is lifted to 3D by triangulating the two views, weighting each observation by its predicted confidence $\widehat{c}^{v}_{\tau,n}$. In the second stage, since every query point is attached to a known robot link, we obtain the joint configuration at each future frame by fitting the robot's forward kinematics (FK) to the lifted points with a robust loss, regularized toward the previous frame for a smooth trajectory. Each point is weighted by $w_{\tau,n}$, its predicted confidence averaged over views, so that unreliable tracks contribute less to the fit. The resulting joint trajectory, together with the gripper opening recovered from the finger points, is executed in a receding-horizon manner, without action labels or any robot-specific training.

Algorithm 1 Zero-shot joint-space control from predicted point trajectories

Require: calibrated exterior views $\mathcal{V}$, current joint state $\mathbf{q}_t$, robot kinematics FK, read-out noise level $\sigma^\star$

Sample N query points on the robot surface at $\mathbf{q}_t$ and project them into each view
Read 2D tracks $\{\widehat{\mathbf{u}}^{v}_{\tau,n}\}$ and confidences $\{\widehat{c}^{v}_{\tau,n}\}$ from the forward pass at noise level $\sigma^\star$
for $\tau = t+1, \ldots, t+h$ **do**
  {Stage 1: 3D lifting}
  $\widehat{\mathbf{P}}_{\tau,n} \leftarrow \textsc{Triangulate}\big(\{\widehat{\mathbf{u}}^{v}_{\tau,n}, \widehat{c}^{v}_{\tau,n}\}_{v\in\mathcal{V}}\big)$ for all n
  $w_{\tau,n} \leftarrow \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} \widehat{c}^{v}_{\tau,n}$ for all n
  {Stage 2: point-to-point IK}
  $\mathbf{q}_\tau \leftarrow \arg\min_{\mathbf{q}} \sum_n w_{\tau,n}\, \rho\big(\|\mathrm{FK}_n(\mathbf{q}) - \widehat{\mathbf{P}}_{\tau,n}\|\big) + \lambda\|\mathbf{q} - \mathbf{q}_{\tau-1}\|^2$
end for
Execute the first steps of $\{\mathbf{q}_\tau\}$ and re-plan from the new observation

B Dataset Curation

To train VPM, we construct 600K samples from three sources: (i) VPM-Data, our multi-view videos with persistent 3D point trajectories from simulation, (ii) robot manipulation video datasets annotated with kinematics-grounded 3D point trajectories, and (iii) open-source video datasets from diverse domains, *e.g.*, MolmoMotion-1M (Zhang et al., 2026b) and Ego-Exo4D (Grauman et al., 2024). We provide detailed statistics in Section B.1, and elaborate on the construction of VPM-Data in Section C and robot data in Section B.2, respectively.

B.1 Data Statistics

We collect 600K samples in total, as summarized in Table 8. VPM-Data contains 10,000 multi-view samples, consisting of 8,000 tabletop manipulation clips and 2,000 urban dynamics clips. The robot datasets contain 91K episodes from DROID (Khazatsky et al., 2024) and HRDexDB (Lim et al., 2026). The remaining clips are collected and sampled from open-source datasets.

Table 8: **Statistics of the training data.**

Source	Domain	Real/Sim	Samples	Views	3D traj.	Action
VPM-Data	Tabletop manipulation	Sim	8,000	3	✓	✗
VPM-Data	Urban dynamics	Sim	2,000	4	✓	✗
DROID	Tabletop manipulation	Real	89,312	3	✓	✓
HRDexDB	Tabletop manipulation	Real	1,601	2	✓	✓
MolmoMotion-1M	Mixed domain	Real	480,000	1-2	✓	✗
Ego-Exo4D	Human activities	Real	24,000	1-4	✓	✗
Total			604,913			

B.2 3D Point Extraction Pipeline from Robot Videos

We describe our pipeline for extracting the 3D point trajectories used to train our model. Our objective is to recover metric, temporally consistent 3D trajectories of action-relevant geometry throughout each manipulation episode. Rather than relying on noisy pseudo-labels from visual point trackers, we construct these trajectories directly from known robot and object geometry together with recorded kinematic states. In particular, robot CAD models provide accurate surface geometry, which can be transformed according to the robot state at each timestep to obtain geometrically grounded 3D point trajectories. We detail this procedure below.

3D Point Extraction. We begin with the CAD or visual meshes of each robot link and manipulated object. We assume calibrated cameras and a known transformation between the robot base and each camera coordinate frame. For each mesh, we sample surface points using area-weighted triangle sampling. Sampling is performed once in the mesh-local coordinate system with a fixed random seed, and the same points are reused throughout the entire episode. By subsequently transforming these fixed local points according to the corresponding link or object pose at each timestep, we obtain temporally consistent 3D point trajectories without resampling noise.

We allocate a fixed point budget to each action-relevant component: we sample 256 3D points from the whole robot body. In cases where the 3D points of the target object are available, such as in simulation settings, we also sample 256 points from the object being manipulated. For articulated components composed of multiple meshes, the budget is distributed approximately in proportion to mesh surface area. Let $\mathbf{p}_n^\ell$ denote a point sampled on the local CAD geometry of link or object ℓ. Its world-space location at timestep τ is obtained as

$$\mathbf{p}_{\tau,n}^w = \mathbf{R}_{\ell,\tau}^w \, \mathbf{p}_n^\ell + \mathbf{b}_{\ell,\tau}^w,$$

where $(\mathbf{R}_{\ell,\tau}^w, \mathbf{b}_{\ell,\tau}^w)$ denotes the pose of the corresponding link or object. Because $\mathbf{p}_n^\ell$ is fixed, point n retains the same physical identity over the complete trajectory. For these sampled points, we apply forward kinematics of the robot via saved robot states, and obtain their 3D positions per time step. For simulated datasets, we replay the recorded simulator state and directly query the per-frame poses of robot links, objects, and cameras. The sampled CAD points can therefore be transformed

using the simulator's underlying geometry without requiring visual pose estimation. For real-robot datasets, we load the corresponding robot model and visual meshes, and manipulate the robot CAD via saved robot states and forward kinematics.

Camera-frame trajectories and projection. Using the extracted 3D points, we first construct point trajectories in a common world frame and subsequently transform them into each camera coordinate system. For camera v, the position $\mathbf{p}_{\tau,n}^{v}$ of point n at timestep τ is given by

$$\mathbf{p}_{\tau,n}^{v} = \mathbf{R}_{\tau}^{v\leftarrow w}\,\mathbf{p}_{\tau,n}^{w} + \mathbf{b}_{\tau}^{v\leftarrow w},$$

where $(\mathbf{R}_{\tau}^{v\leftarrow w}, \mathbf{b}_{\tau}^{v\leftarrow w})$ denotes the world-to-camera extrinsics of camera v at timestep τ. Because the same point identities are preserved under these transformations, this representation provides explicit point correspondences (Yue et al., 2025; Chen et al., 2026; Lee et al., 2026c) across both time and viewpoints.

Data Filtering for Robot Data. Although the real-robot datasets provide camera calibration, even relatively small errors, such as errors in camera extrinsics or in the robot-base calibration can create substantial image-space misalignment between the projected CAD model and the observed robot. We therefore explicitly validate the geometric alignment of each real episode before retaining it.

We evaluate calibration quality using agreement between the projected CAD geometry and an independently obtained image-space robot mask. We first use the recorded joint configuration and camera calibration to solve the robot kinematics and render the corresponding CAD model into every frame, producing a binary projected CAD mask M_t^{CAD}.

Rather than evaluating all frames equally, we select the frame $\tau^\star$ in which the projected robot occupies the largest image area, defined as

$$\tau^\star = \arg\max_{\tau} \left|M_{\tau}^{\mathrm{CAD}}\right|.$$

This frame provides the strongest calibration signal because a larger visible robot region reduces sensitivity to small masks, truncation, and weak overlap statistics.

For the selected frame $\tau^\star$, we obtain an image-derived robot mask $M_{\tau^\star}^{\mathrm{SAM}}$ using SAM2 (Ravi et al., 2025) and compare it against the rendered CAD mask. We use CAD-mask recall, defined as

$$\mathrm{Recall}_{\mathrm{CAD}} = \frac{\left|M_{\tau^\star}^{\mathrm{CAD}} \cap M_{\tau^\star}^{\mathrm{SAM}}\right|}{\left|M_{\tau^\star}^{\mathrm{CAD}}\right|},$$

which measures the fraction of geometry predicted by the calibrated robot model that is supported by the observed image mask. We retain episodes with a recall of at least 0.9 and discard those below this threshold. Low recall indicates substantial disagreement between the kinematically projected CAD model and the image evidence, suggesting that the resulting 3D supervision may be unreliable.

We find recall preferable to symmetric mask-overlap metrics, as the SAM2 mask may include regions that do not belong to the robot or may merge the robot with nearby objects. Our primary requirement is instead that the geometry used to construct the 3D supervision projects consistently onto the robot observed in the image.

Output representation. The resulting sample contains synchronized multi-view RGB observations together with persistent 3D surface-point trajectories for the robot arm, gripper or hand, and manipulated objects. We additionally retain the recorded joint states and all camera metadata required to reproduce the trajectories and their image-space projections, including intrinsics, distortion parameters, and per-frame extrinsics. Consequently, each point index refers to the same physical surface location across both time and camera viewpoints.

C TRAINING DATA GENERATION

We use separate pipelines for tabletop manipulation and urban capture, with a shared annotation format. This section describes training-data generation; benchmark construction is detailed in Section D.

C.1 CAPTURE SETTINGS AND ENVIRONMENTS

Table 9 summarizes the capture settings; corpus composition is reported in Table 1. Tabletop training uses city environment C_1 as static scenery, whereas urban training captures live agents in C_2. The benchmark additionally uses unseen kitchen environments H for manipulation and moving agents in C_1 for urban evaluation. The latter environment is therefore familiar as scenery, but its traffic and crowd dynamics are absent from training.

Table 9: **Training-data capture settings.** Both domains use 1280×1280 images and distortion-free pinhole cameras.

Setting	Tabletop	Urban
Motion source	Recorded manipulation	Live traffic and crowds
Views / frame rate	3 / 20 fps	4 / 15 fps
Clip length	81–181 frames	81, 100, 121, 144, or 181 frames
Intrinsics	Authored	Sampled per camera and clip
Camera motion	Static or planned hard-clip paths	Static, arc, translation, random polyline, or pan–tilt
Visibility	Rendered depth, 3 cm tolerance	Surface-point ray casting
Reference camera	`cam00`	`cam01`

C.2 TABLETOP GENERATION

Motion and scene composition. We replay physics-simulated pick-and-place episodes of a Fourier GR-1 T2 robot using recorded joint configurations and object transforms. Each clip contains 1–4 targets or receptacles, augmented with 5–8 non-contact and 0–2 contact distractors drawn from 306 props with annotated resting poses. All distractors avoid the manipulated objects' swept volumes. Contact distractors additionally overlap the robot's swept volume, with at least 25% of a $3\times3\times3$ grid inside their bounding box lying within that volume. Non-contact props alternate between the robot's left and right.

Physics validation. Distractors are simulated at 60 Hz, with three substeps per video frame. They first settle for up to 60 frames, requiring translation below 0.1 cm and rotation below 0.5° for five consecutive frames. Placements with falling props are rejected. During replay, contact distractors must reach 1 cm contact depth for at least two frames and move by at least 1 cm. We allow up to five placement attempts, then bake the accepted per-frame transforms for rendering.

Camera and background placement. Camera poses are perturbed in robot-local coordinates: distance is scaled by $[0.94, 1.06]$, azimuth by $\pm4^\circ$, and elevation by $\pm3^\circ$. Cameras are re-aimed at the work surface while remaining at least 5 cm above it and 3 cm clear of the robot. The complete scene and camera rig are then relocated into C_1, accepting only placements where the entire camera path remains clear of scene geometry.

C.3 URBAN CAPTURE

The camera rig is placed at one of 13 surveyed locations in C_2. After the traffic and crowd simulation settles, we select a visible pedestrian or vehicle satisfying the motion criteria, with a separate quota of stationary targets. Targets use 12 vehicle models and six pedestrian body meshes; procedural combinations of clothing, hair, and footwear provide additional appearance variation.

Each camera independently samples a horizontal field of view between 25° and 90° and a trajectory type from Table 9. Cameras aim at the target except during scheduled departures. Images and

Table 10: **Hard-clip criteria for the training pipelines.** An invisible interval comprises frames labeled occluded or out of frustum. Benchmark definitions and measured statistics are reported separately in Section D.

Criterion	Tabletop	Urban
Invisible interval	≥ 30 consecutive frames per camera	≥ 30 per camera, followed by re-acquisition
Collective coverage	At least one visible camera per frame	At least one visible camera per frame
Motion in interval	Target displacement ≥ 5 cm	Target moving in $\geq 40\%$ of interval frames
Additional condition	≥ 20 occluded frames summed over cameras	Two-phase departure schedule

annotations are exported from the same simulation timestep across all views. The scheduler balances the five clip-length buckets, with hard clips restricted to 181 frames.

C.4 Geometric and Language Annotations

Persistent surface points. We sample 256 points per annotated object. Tabletop samples are cached per mesh and propagated using recorded transforms; urban samples are initialized per acquired target. On deforming meshes, component and triangle identities with barycentric coordinates recover the same surface points from the deformed vertices at each timestep. Tabletop clips additionally contain 512 robot points, split equally between the arms. Allocation is proportional to link surface area with a minimum of four points per link, reserving half of each arm's budget for the hand.

Visibility. Tabletop visibility uses rendered depth with a 3 cm tolerance. Urban visibility tracks up to 32 in-frustum surface points per camera and frame, excluding the target itself from collision tests and allowing a 15 cm margin. The target is visible if at least three points are unoccluded, out of frustum if none is in frame, and occluded otherwise. Invalid surface reconstructions count as occluded. Urban clips retain separate bounding-box and traced visibility arrays: the former supports the visibility acceptance check, while the latter controls departures, re-acquisition, and hard labels.

Annotation format. Camera files store intrinsics and per-frame 4×4 world-to-camera matrices. Point arrays use reference-camera coordinates in metres under the OpenCV axis convention. Object trajectories have shape $(T, 256N_{\text{obj}}, 3)$ for a T-frame clip with N_{obj} annotated objects, including contact distractors, and tabletop robot trajectories have shape $(T, 512, 3)$. Metadata records object index ranges, roles, point validity, per-view visibility, and hard-label measurements. Each clip contains per-camera RGB videos, `depth.zip`, `cameras.json`, `points_target.npz`, `visibility.json`, `meta.json`, and `caption.txt`.

Captions. Motivated by SpatialBoost (Jeon et al., 2026), we generate captions per clip through a vision-language model and filter them. Captions must agree with recorded motion, omit numerical claims and camera descriptions, and satisfy the single-sentence format. Failures are regenerated with the validation feedback for up to two rounds; unresolved failures are excluded.

C.5 Visibility-Controlled Clips

Hard clips introduce incomplete individual-camera observations while retaining target coverage across the camera set. Table 10 states the training-label criteria; labels are verified from measured visibility rather than assigned from the camera schedule alone.

Tabletop construction. One camera starts 8–12 cm above the work surface. Non-contact occluders are placed along camera–target segments at relative positions 0.40–0.85; when placement fails, cameras may rotate away by up to 75°. Paths are limited to 2.5 cm translation and 2° rotation per frame, with 30 cm inter-camera separation, 20° yaw separation, and 15 cm forward lens clearance. Invisible-window starts are searched within ±60 frames at two-frame intervals. Collision tracks to 32 target points verify the schedule during physics replay. Failed attempts revert to ordinary camera jitter and receive a normal label.

Urban construction. All four cameras depart and re-acquire the target in two phases. A new phase starts only while a camera from the preceding phase remains traced-visible; capture aborts if collective visibility is lost. Hard clips use the 181-frame bucket, with a scheduler target of 10–20% of the urban training corpus. Failed hard attempts are discarded and replaced.

C.6 Validation and Rendering

Clips are checked against the criteria in Table 11. Failed samples are replaced, and samples with unavailable validation inputs are held rather than accepted. The training profile records departure–re-acquisition as a label; the stricter benchmark profile additionally enforces it as an acceptance condition.

Table 11: **Configured acceptance checks.** Departure–re-acquisition is recorded but not required by the training profile.

Check	Threshold
Target visibility	Minimum visible ratio ≥ 0.1 over all cameras
Survival and motion	Target present throughout; travel ≥ 1.0 m per 81 frames, with a 0.5 m floor
Track completeness	No track breaks; completeness ≥ 0.99
Depth validity	Sky, near, and constant-value pixel ratios each ≤ 0.25
Camera clearance	At most two cameras with blocked paths; view-block ratio ≤ 0.6
Caption	Present, at least 20 characters, and passes format checks
Departure cycle	At least one cycle per scheduled camera, with ≥ 8 frames out and ≥ 8 frames visible
Camera smoothness	Jolt ≤ 40 cm; step ≤ 160 cm; rotation ≤ 0.08 of horizontal field of view per frame; vertical vibration ≤ 4 cm peak-to-peak at 1.3–3.0 Hz
Motion persistence	Target speed ≥ 40 cm/s in at least 40% of frames

For urban surface tracks, reprojection checks require at least 99% of points to agree within one pixel, with no track breaks. Tabletop rendering uses two spatial samples per pixel and 32 warm-up frames; urban capture uses a warm-up and settling schedule. RGB is encoded as H.264, and optical-axis depth is stored as 16-bit PNG in centimetres, valid from 1 cm to 200 m. Per-camera valid-depth ratios are recorded. Images and annotations use the same evaluated scene state.

Tabletop randomization uses per-clip seeds, and validated transforms are replayed during rendering. Urban seeds are recorded, but the live traffic and crowd simulation does not guarantee bit-exact regeneration.

D VP-WORLDBENCH: CONSTRUCTION AND STATISTICS

This appendix documents the simulation component of VP-WorldBench, which pairs synchronized multi-view videos with metric surface-point trajectories, camera geometry, and per-view visibility. Its capture pipelines are distinct from the training-data pipelines in Section C. Real-world benchmark details are reserved for Section D.7.

D.1 COMPOSITION AND CAPTURE SETTINGS

The simulation benchmark contains 500 tabletop and 500 urban clips, with 661,071 images and 716,544 persistent point trajectories (Table 12). Images and annotations correspond to the same simulation timesteps, and point identities remain defined during occlusion. Reported statistics are measured from the released files.

Table 12: **Simulation benchmark composition and capture settings.** Timesteps are counted once per synchronized camera set; images count all views. Trajectory counts sum persistent point identities over clips. All images are 1280×1280, with distortion-free pinhole cameras.

Property	Tabletop	Urban
Clips (hard)	500 (27)	500 (95)
Views / frame rate	3 / 20 fps	4 / 15 fps
Clip length	81–680 frames	81–181 frames
Synchronized timesteps	129,441	68,187
RGB images	388,323	272,748
3D point trajectories	588,544	128,000
Horizontal field of view	93.0°, 100.5°, 67.0°	25.0°–89.9°
Camera translation	None within clips	Median 10.7 cm/frame
Reference camera	`cam00`	`cam01`
Median target path length	0.867 m	19.0 m

Tabletop clips cover city environment C_1 (436 clips) and two kitchens, k_1 (28) and k_2 (36). They contain 1,299 tracked objects: 693 manipulation targets and 606 receptacles. Of the 500 clips, 392 retain their source-episode length and 108 are windowed to 81–181 frames. Urban clips contain 332 pedestrians and 168 vehicles spanning 55 target models, with at most 11 clips per model. Their lengths are 81, 100, 121, 144, and 181 frames, containing 79, 79, 83, 82, and 177 clips, respectively. All 95 hard urban clips belong to the 181-frame group. The relationship of these environments to training is described in Section C.1.

D.2 TABLETOP CAPTURE

Scene composition. Each clip replays a physics-simulated Fourier GR-1 T2 pick-and-place episode in Unreal Engine. Source sequences contain 54 robot joint tracks, 1–4 replay objects, a work surface, three cameras, and swept-volume geometry for the robot and objects. Seeded randomization varies camera poses, surface materials, and clutter. Distractors avoid the moving objects' swept volumes, with a quota of 0–2 props intersecting the robot's swept volume. Props settle and interact with the robot under physics, after which their transforms are recorded for rendering. Released clips contain 1–12 distractors (median 8); contacts and subsequent motion are retained.

Cameras and rendering. The three cameras retain their authored intrinsics and receive robot-local pose perturbations: distance scaled by $[0.94, 1.06]$, azimuth by $\pm4^\circ$, and elevation by $\pm3^\circ$. They are aimed at the manipulation region. Camera positions remain fixed within each benchmark clip; hard clips additionally use the rotations described in Section D.5. Per-frame world transforms are read back for all objects, 36 robot links, and cameras, including their parent transforms. RGB and depth are rendered at these timestamps with two spatial samples per frame. All clips use a fixed AgX tone curve and recorded lighting settings; lighting is not randomized in this benchmark.

D.3 URBAN CAPTURE

Agents and synchronization. Urban clips are captured in an open-source driving simulator built on Unreal Engine, using six transformed city tiles with 60 additional building models and fixed noon weather. Vehicles follow the traffic manager and pedestrians follow navigation destinations, amid approximately 30 background vehicles and 40 pedestrians. A target is acquired once its instance occupies at least 300 pixels in a camera; capture starts after 15 frames of motion. The simulator advances at $1/15$ s intervals and waits for every sensor before the next tick. Each view contains pixel-aligned RGB, depth, and instance-segmentation sensors, with motion blur and depth of field disabled.

Camera configuration. Each camera independently samples its horizontal field of view from 25° to 90°. Initial positions are selected from 32 ray-tested azimuths around the target, with distance chosen to control its image coverage. Cameras follow static, arc, translation, random-polyline, or pan–tilt paths and aim at the target except during scheduled departures. Path control limits translation to 150 cm per frame, changes in the translation step to 20 cm, and rotation to 0.06 times the horizontal field of view per frame. A two-stage height filter with a 0.5 s time constant stabilizes camera aiming without modifying the target's recorded trajectory.

D.4 GROUND-TRUTH ANNOTATIONS

Persistent points. We sample 256 area-weighted surface points per object once per mesh and preserve their identities across frames and clips. Tabletop points follow recorded rigid transforms. The robot has 512 additional points, divided equally between its arms and allocated across 18 links per side, with at least four points per link and half of each arm's budget reserved for the hand and fingers. For urban targets, points store triangle indices, barycentric coordinates, and skinning weights. Per-frame bone transforms recover their positions through linear blend skinning, with rigid components propagated from their attachment transforms. These annotations follow skeletal deformation, not simulated cloth or hair.

Coordinates and depth. Stored trajectories use the reference camera's frame at each timestep, in metres under the OpenCV convention ($+X$ right, $+Y$ down, $+Z$ forward). Per-frame 4×4 world-to-camera matrices and intrinsics are stored for every view, allowing conversion to world coordinates. Motion-based acceptance checks use world-space target motion. Depth is stored as 16-bit PNG in centimetres, with far limits of 200 m for kitchens, 650 m for tabletop city scenes, and 655 m for urban scenes.

Visibility labels. For tabletop points projected inside the image, visibility requires camera-space depth $z \le d + \epsilon$, where d is rendered depth and $\epsilon = 3$ cm. An object is visible when more than 10% of its in-frustum points pass this test; labels also distinguish occlusion, departure from the frustum, and absence. Urban clips contain two target-visibility arrays. `visibility` uses bounding-box projection and requires at least 300 target-instance pixels for visibility. `visibility_strict` instead uses projected surface points with the same instance-pixel criterion. The bounding-box array supports the visibility acceptance check, while the strict array determines departures, re-acquisition, and hard labels.

Language annotations. Each clip receives an English caption from a vision-language model. Tabletop captioning uses the recorded object identities, motion, active hand, and scene context; its style checks are advisory. Urban captions are generated from representative video frames after selection, and failing captions are regenerated. All 500 urban captions pass the final style checks.

D.5 VISIBILITY-CONTROLLED SUBSETS

Hard subsets introduce sustained visibility losses in individual views while preserving target visibility across the camera set. Table 13 summarizes their domain-specific criteria and measured composition.

Tabletop construction. One camera is placed 8–12 cm above the work surface and 55–70 cm from the target region to induce occlusion by arms and clutter. The other two rotate away in staggered windows, overlapping only while the low camera is predicted to retain visibility. Rotation is limited to 0.08 times the camera's half field of view per frame. Planned visibility is estimated from 32 target

Table 13: **Hard-subset criteria and composition.** An invisible interval includes occluded and out-of-frustum frames. Urban collective coverage is evaluated on the strict array.

Property	Tabletop	Urban
Hard clips	27	95
Cameras with $\geq$ 30 consecutive invisible frames	All 3	2 or 3[†]
Target motion in interval	Displacement ≥ 5 cm	Moving fraction ≥ 0.4
Additional occlusion requirement	≥ 20 frames summed over views	None
Frames with no visible camera	0	0 (strict)

[†]All four urban cameras complete departure and re-acquisition cycles. Of the 95 hard urban clips, 31 have two and 64 have three cameras meeting the 30-frame threshold.

surface points; the camera paths are then recorded and the hard label recomputed from rendered depth. Failed hard attempts are retried once before reverting to normal clips. Of 100 attempts, 27 satisfy the hard criteria.

Urban construction. All four cameras depart and re-acquire the target in two phases. The second phase begins only while a camera from the preceding phase is strict-visible, and capture aborts if collective visibility is lost. These clips use 181 frames. Normal clips may also include departures: each camera is scheduled with probability 0.3, with at most two departures per clip and none in the 81-frame group.

Measured coverage. Table 14 reports target visibility across cameras. No hard tabletop frame is invisible to all views. The urban bounding-box array contains one such hard frame, whereas the strict array used for labeling contains none. These measurements characterize the temporal availability of direct target observations across the camera set.

Table 14: **Percentage of frames by the number of cameras seeing the target.** Urban columns use bounding-box visibility; strict-array collective coverage is reported separately.

	Tabletop ($V = 3$)		Urban ($V = 4$)	
Visible cameras	Normal	Hard	Normal	Hard
4	N/A	N/A	71.81	36.48
3	89.67	28.60	21.63	24.41
2	9.65	48.49	5.78	29.34
1	0.67	22.91	0.67	9.77
0	0.012	0.000	0.12	0.01[†]
Timesteps	121,406	8,035	50,992	17,195

[†]One urban hard frame has no visible camera under bounding-box visibility; zero of 17,195 hard frames do so under strict visibility.

D.6 Validation and Limitations

Acceptance checks. Tabletop clips are validated after rendering for point–depth agreement, camera-pose consistency, depth validity, target visibility, object-trajectory clearance, hard-label agreement, camera and pose-array completeness, sweep-geometry alignment, and caption availability. Target, arm, and hand point–depth checks require median error no greater than 3 cm. Urban clips follow the thresholds in Table 15. Failed clips are replaced and their rejection reasons retained; checks with missing inputs do not pass.

Annotation quality and selection. Urban surface reconstruction is checked against rendered depth and instance silhouettes, with 90th-percentile reprojection errors of 0.00 px for pedestrians and 1.00 px for vehicles. Transparent and unsupported surface sections are excluded from this validation. All urban clips have complete tracks without breaks. Of 793 judged urban clips, 526 pass the strict profile and 500 are selected. The hard subset contains 78 pedestrian and 17 vehicle clips, reflecting a lower acceptance rate for fast-moving vehicles.

Table 15: **Urban benchmark acceptance thresholds.** The caption check is applied after selection.

Check	Threshold
Target visibility	Minimum visible ratio ≥ 0.1 over cameras without scheduled departures
Survival and motion	Target present throughout; travel ≥ 1.0 m per 81 frames, with a 0.5 m floor
Track completeness	Non-empty points, no track breaks, completeness ≥ 0.99
Depth validity	Sky, near, and constant-value pixel ratios individually ≤ 0.25
Camera clearance	At most two cameras with blocked paths; view-block ratio ≤ 0.6
Caption	Present, at least 20 characters, and passes style checks
Departure cycle	At least one cycle per scheduled camera, with ≥ 8 frames out and ≥ 8 frames visible
Camera smoothness	Jolt ≤ 40 cm; step ≤ 160 cm; rotation ≤ 0.08 times horizontal field of view per frame; vertical vibration ≤ 4 cm peak-to-peak at 1.3–3.0 Hz
Motion persistence	Speed ≥ 40 cm/s in at least 40% of frames

Limitations and reproducibility. Simulation depth does not reproduce real sensor noise and has invalid sky and out-of-range regions. Camera counts, frame rates, and clip lengths differ by domain. The hard urban subset over-represents pedestrians, and skeletal point annotations omit independent cloth and hair motion. Rendering profiles are fixed rather than calibrated to real camera responses. Seeds are recorded, but physics settling and live traffic prevent bit-exact regeneration from seeds alone; recorded per-frame poses provide the reference for replay.

D.7 Real-World Benchmark Details

Real-world human benchmark. Our real-world benchmark split comprises 100 newly recorded 10-second sequences featuring three participants in total. Each sequence includes synchronized multi-view RGB videos, a sequence of reconstructed 3D surfaces, and camera parameters. To capture realistic everyday activities, we constructed a $23.1\,\mathrm{m}^2$ home-like space furnished with a sink, cabinets, drawers, a refrigerator, a sofa, tables, and household objects. Participants move through the space, lift and carry objects, open drawers, and set objects down, producing changes in both human pose and scene configuration.

Capture system. We capture RGB videos using 48 FLIR industrial cameras at 30 fps and a resolution of 2048×1536 pixels. The camera array combines wide-angle lenses for broad spatial coverage with standard-field-of-view lenses for capturing detail. We synchronize image capture across all cameras using an external hardware trigger. Before recording, we calibrate camera intrinsics using Zhang's method (Zhang, 2000) and estimate extrinsics using 16 calibration boards distributed throughout the room. Specifically, perspective-n-point estimates initialize the relative camera-board poses, which are linked through a pose graph in a common metric coordinate system. We further refine the calibration through feature matching and bundle adjustment using COLMAP (Schönberger & Frahm, 2016). The calibrated camera parameters are used for image undistortion, 3D reconstruction, and as model inputs.

Reconstruction and filtering. We undistort the images using the calibrated camera parameters and reconstruct each timestamp independently with AmbiSuR (Li et al., 2026a), using depth priors from Depth Anything 3 (Lin et al., 2025). For all sequences, we optimize the Gaussians for 2,000 iterations per timestamp before exporting the surfaces. To reduce holes in the reconstructed surfaces, we extract surface points by back-projecting rendered depth maps rather than directly exporting Gaussian as surface points.

To reduce floating artifacts, we further filter the reconstructed 3D points. For each point, we count the views in which the rendered depth at its projected image location agrees with its camera-axis depth within $5\,\mathrm{mm}$. We retain only points supported by at least two views and having at least five neighbors within a $10\,\mathrm{mm}$ radius. Finally, to make downstream tasks computationally tractable, we downsample the filtered points using a $1\,\mathrm{mm}$ voxel grid, retaining the point with the greatest view support in each occupied voxel.

D.8 Relation to Existing Resources

Related resources emphasize complementary annotations and tasks. BEDLAM 2.0 (Tesch et al., 2026) provides human-body geometry, while ReCamMaster's Multi-Cam Video dataset (Bai et al., 2025) supports multi-view re-rendering with camera trajectories. TAPVid-MV (Koppula et al., 2026) evaluates multi-view point tracking, and PointMotionBench (Zhang et al., 2026b) evaluates language-conditioned point-motion forecasting. Our simulation benchmark pairs future multi-view appearance with metric surface-point trajectories, camera geometry, depth, visibility, and language on the same clips.

E Additional Experimental Results

E.1 Detailed Experiments on VP-WorldBench

In this section, we discuss the detailed results, varying the evaluation setup under VP-WorldBench. Specifically, in addition to the all-view Image-to-Video (I2V) setup in Table 2, we evaluate VP-WorldBench under the following setups:

- **All-view I2V** (Table 16): We report performance using all views for each split.
- **2-view I2V** (Table 17): We use two input views instead of all.
- **All-view V2V** (Table 18): We condition on input videos instead of images.
- **Trajectory head** (Table 19): We directly measure the performance of VPM trajectory head.

Table 16: **Results on VP-WorldBench (I2V with all-view).** V and T denote video and point-trajectory supervision. Best results are in **bold**. The upper bound is obtained by applying the same evaluation pipeline to the ground-truth videos.

		Simulation								Real-world							
		Robot (500)				General (500)				Robot (100)				General (100)			
Method	MV	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑
HV-1.5	✗	98.34	20.21	0.023	41.62	115.63	11.45	0.061	23.10	426.76	16.04	0.102	36.17	105.83	20.00	0.014	21.35
Wan2.2-5B	✗	315.13	15.84	0.108	29.51	205.73	10.43	0.076	18.00	550.55	14.58	0.159	29.65	115.52	16.97	0.030	16.82
X-WAM	✓	203.83	19.97	0.038	38.80	-	-	-	-	492.31	17.63	0.091	50.86	-	-	-	-
Cosmos3-N	✗	104.74	20.05	0.040	39.03	167.02	10.99	0.062	22.12	326.11	16.37	0.099	41.50	66.24	20.89	0.015	20.47
VPM (V)	✓	78.52	21.41	0.022	43.62	101.37	16.95	0.057	28.71	296.83	18.42	0.074	50.83	62.85	20.96	0.010	26.54
VPM (V+T)	✓	**67.35**	**22.08**	**0.016**	**50.05**	**81.92**	**19.16**	**0.050**	**36.80**	**255.19**	**20.57**	**0.064**	**64.75**	**56.48**	**22.10**	**0.007**	**33.92**
+ camera cond.	✓	65.34	22.75	0.014	55.05	79.33	19.39	0.046	40.04	244.84	21.03	0.062	66.75	49.77	22.45	0.005	38.46
Upper bound	–	–	–	0.004	73.27	–	–	0.029	61.87	–	–	0.010	80.40	–	–	0.003	57.98

In Table 16, we provide a full table for Table 2, reporting the performance with the camera condition in which the target camera trajectory is given for the forecasting, and the upper bound. We evaluate I2V performance with all-view from each split and follow the same evaluation protocols as Table 2.

Table 17: **Results on VP-WorldBench (I2V with 2-view).** V and T denote video and point-trajectory supervision. Best results are in **bold**. The upper bound is obtained by applying the same evaluation pipeline to the ground-truth videos.

		Simulation								Real-world							
		Robot (500)				General (500)				Robot (100)				General (100)			
Method	MV	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑
HV-1.5	✗	85.91	20.72	0.021	39.22	141.01	11.66	0.062	20.46	426.76	16.04	0.102	36.17	117.21	20.17	0.016	19.54
Wan2.2-5B	✗	294.52	16.42	0.089	27.62	180.34	9.96	0.070	17.33	550.55	14.58	0.159	29.65	151.03	16.76	0.033	16.20
X-WAM	✓	154.26	20.03	0.033	36.73	-	-	-	-	492.31	17.63	0.091	50.86	-	-	-	-
Cosmos3-N	✗	86.05	20.17	0.037	34.66	202.55	10.06	0.058	19.99	326.11	16.37	0.099	41.50	88.21	20.08	0.018	19.48
VPM (V)	✓	85.51	20.95	0.022	40.08	102.95	17.36	0.056	27.42	296.83	18.42	0.074	50.83	71.65	21.12	0.014	23.70
VPM (V+T)	✓	**63.46**	**21.75**	**0.015**	**45.51**	**84.13**	**19.68**	**0.044**	**33.18**	**255.19**	**20.57**	**0.064**	**64.75**	**57.49**	**21.58**	**0.008**	**31.62**
+ camera cond.	✓	60.14	22.22	0.014	52.42	79.07	19.91	0.043	37.42	244.84	21.03	0.062	66.75	48.76	22.29	0.007	39.38
Upper bound	–	–	–	0.003	65.98	–	–	0.026	55.40	–	–	0.010	80.40	–	–	0.003	53.03

In Table 17, we provide a table with 2-view I2V setup on VP-WorldBench.

Table 18: **Results on VP-WorldBench (V2V with all-view).** V and T denote video and point-trajectory supervision. Best results are in **bold**. The upper bound is obtained by applying the same evaluation pipeline to the ground-truth videos.

		Simulation								Real-world							
		Robot (500)				General (500)				Robot (100)				General (100)			
Method	MV	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑
Cosmos3-N	✗	111.92	20.06	0.044	40.91	140.65	11.05	0.067	23.27	296.25	16.34	0.099	43.53	77.55	21.03	0.018	21.72
+ camera cond.	✗	110.23	19.98	0.040	43.27	145.96	10.46	0.067	19.08	288.63	17.49	0.095	44.92	72.59	21.17	0.014	22.80
VPM (V)	✓	73.31	21.44	0.028	43.75	125.83	17.53	0.060	28.92	243.62	18.32	0.073	53.12	66.96	21.21	0.012	28.09
VPM (V+T)	✓	**63.77**	**22.18**	**0.021**	**50.17**	**81.49**	**19.36**	**0.055**	**38.31**	**239.61**	**20.56**	**0.064**	**68.73**	**48.25**	**22.11**	**0.010**	**34.94**
+ camera cond.	✓	59.56	22.56	0.018	52.57	75.58	19.81	0.053	42.81	198.44	21.16	0.060	73.60	47.69	22.29	0.007	40.25
Upper bound	–	–	–	0.004	73.27	–	–	0.029	61.87	–	–	0.010	80.40	–	–	0.003	57.98

In Table 18, we provide a table with all-view Video-to-Video (V2V) setup on VP-WorldBench. Compared to other baselines, Cosmos3-Nano (Agarwal et al., 2026) is the only model that supports V2V generation. With camera condition, VPM surpasses the baseline, *i.e.*, Cosmos3-Nano on every split.

Table 19: **Results on VP-WorldBench with VPM trajectory head.** V and T denote video and point-trajectory supervision. Best results are in **bold**.

		Simulation		Real-world	
		Robot (500)	General (500)	Robot (100)	General (100)
Method	Tracker	δ^{3D}_{avg} ↑	δ^{3D}_{avg} ↑	δ^{3D}_{avg} ↑	δ^{3D}_{avg} ↑
VPM (V)	MVTracker	43.62	28.71	50.83	26.54
VPM (V+T)	MVTracker	50.05	36.80	64.75	33.92
+ camera cond.	MVTracker	55.05	40.04	66.75	38.46
VPM (V)	Ours	45.06	29.92	51.78	28.39
VPM (V+T)	Ours	**52.15**	**38.46**	**67.22**	**36.16**
+ camera cond.	Ours	56.81	42.82	69.17	39.41

VPM has a trajectory head that predicts the positions of future 3D points. Unlike Table 2, where point tracks are extracted from MVTracker (Rajič et al., 2025), here we report the quality of point tracks directly predicted by our trajectory head. As shown in Table 19, direct prediction consistently improves 3D tracking accuracy, *e.g.*, δ^{3D}_{avg} of VPM (V+T) increases from 64.75 to 67.22 on Real-world Robot.

For all variations, VPM (V+T) achieves state-of-the-art performance on I2V and V2V, indicating that the supervision of point trajectories consistently enhances the multi-view video fidelity.

E.2 Detailed Experiments on 4D Generation

Table 20: **Results of 4D generation on ReCamMaster and CMU Panoptic.**

		ReCamMaster (100)		CMU Panoptic (100)			
Method	MV	FVD ↓	PSNR ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑
HV-1.5	✗	183.48	14.03	352.15	21.60	0.041	31.67
Wan2.2-5B	✗	457.16	12.10	258.06	18.65	0.061	27.88
Cosmos3-N	✗	125.70	13.99	162.36	20.72	0.060	30.12
VPM (V)	✓	103.12	15.09	150.63	20.85	0.037	34.72
VPM (V+T)	✓	**79.63**	**17.06**	**128.64**	**23.24**	**0.029**	**42.03**
+ camera cond.	✓	78.03	17.60	120.23	23.49	0.026	44.61
Upper bound	–	–	–	–	–	0.009	64.12

Table 21: **Results of 4D generation on RoboCasa Kitchen.**

		RGB			Geometry	
Method	MV	PSNR↑	SSIM↑	LPIPS↓	MEt3R↓	δ^{3D}_{avg} ↑
DreamZero	✗	20.47	0.7612	0.1651	0.061	30.83
Robot4DGen	✓	22.05	0.8035	0.1078	0.025	46.96
X-WAM	✓	22.41	0.8729	0.0571	0.042	42.10
VPM (V)	✓	22.61	0.8752	0.0558	0.033	46.94
VPM (V+T)	✓	**22.83**	**0.8801**	**0.0537**	**0.020**	**53.70**
+ camera cond.	✓	22.10	0.8835	0.0501	0.013	58.48
Upper bound	–	–	–	–	0.004	78.88

In Tables 20 and 21, we provide detailed results for Tables 3 and 4, including the upper bounds of cross-view MEt3R and δ^{3D}_{avg} on CMU Panoptic and RoboCasa Kitchen.

Table 22: **Per-frame geometry vs. 3D point trajectory supervision.** All variants share the backbone, data, feature layer, and training schedule of VPM (V+T). Per-frame variants are supervised with the same 3D point positions at every frame without correspondence across frames.

	Simulation				Real-world			
	Robot (500)		General (500)		Robot (100)		General (100)	
Supervision	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑
None	78.52	43.62	101.37	28.71	296.83	50.83	62.85	26.54
Per-frame geometry (no temporal identity)								
Depth	77.07	45.21	100.02	29.88	285.13	53.73	62.08	26.98
Pointmap	75.48	44.90	99.35	29.69	281.67	54.50	61.37	29.46
3D point trajectory (temporal identity)								
w/o $\mathcal{L}_{corr}$	69.88	48.54	85.73	31.54	271.01	59.94	59.33	30.75
Full	**67.35**	**50.05**	**81.92**	**36.80**	**255.19**	**64.75**	**56.48**	**33.92**

E.3 Comparison with Per-Frame Geometry Supervision

Setup. To compare per-frame geometry supervision with ours, we replace the trajectory head of VPM (V+T) with a per-frame geometry head. Both heads are supervised at the pixels where the annotated 3D points project in each frame, using their depth or 3D position. The per-frame head thus receives the same geometric supervision, but without linking points across frames: it predicts geometry at given pixel locations, whereas our head must predict where each queried point moves.

Results. As shown in Table 22, per-frame geometry supervision improves over VPM (V) on every split, recovering only 19% (depth) and 28% (pointmap) of the FVD improvement of VPM (V+T). Pointmaps improve over depth maps in FVD on every split, while δ^{3D}_{avg} is comparable, with depth slightly higher in simulation. Even without $\mathcal{L}_{corr}$, 3D trajectories outperform both on every split while supervising the same 3D positions at every frame. We therefore attribute this gain to temporal identity, which additionally constrains the motion of each surface point across frames. The gap is largest on the dynamic splits, *e.g.*, a δ^{3D}_{avg} of 54.50 vs. 59.94 on Real-world Robot and an FVD of 99.35 vs. 85.73 on Simulation General, where points move substantially and are often occluded.

Table 23: **Results on 3D point forecasting.** We evaluate the models on PointMotionBench (Zhang et al., 2026b). Best results are in **bold**, second best are underlined.

Paradigm	Model	Inputs		HOT3D			WorldTrack			DAVIS		
		Frames	Text	ADE↓	FDE↓	PWT↑	ADE↓	FDE↓	PWT↑	ADE↓	FDE↓	PWT↑
Non-parametric	Static	1	✗	0.180	0.316	0.293	0.167	0.317	0.390	2.281	4.360	0.085
	Extrapolate	3	✗	0.159	0.309	0.351	0.184	0.432	0.436	2.683	5.741	0.104
Pixel-space	Wan2.2-5B	1	✓	0.200	0.308	0.253	0.852	1.046	0.090	3.074	5.192	0.051
	Cosmos-Predict	5	✓	0.225	0.294	0.199	0.831	0.988	0.072	4.191	6.368	0.033
3D model	ObjectForesight	3	✗	0.129	0.192	0.353	–	–	–	–	–	–
	EgoScaler	1	✓	0.170	**0.179**	0.218	–	–	–	–	–	–
	Robot4DGen	3	✗	0.212	0.271	0.112	0.548	0.704	0.121	2.120	3.382	0.081
2D track	Track2Act	1	✗	0.294	0.413	0.202	1.230	1.567	0.053	4.853	8.110	0.018
3D track	MolmoMotion-FM	1	✓	0.183	0.311	0.286	0.165	0.305	0.401	1.380	2.205	0.165
	MolmoMotion-FM	3	✓	0.135	0.255	0.382	0.158	0.295	0.438	1.480	2.520	0.130
	MolmoMotion-AR	1	✓	0.157	0.290	0.303	0.148	0.269	0.424	1.146	1.843	**0.199**
	MolmoMotion-AR	3	✓	0.109	0.217	**0.444**	0.143	0.261	0.445	1.227	2.108	0.153
	VPM (ours)	3	✓	**0.098**	0.209	0.396	**0.129**	**0.248**	**0.470**	**1.124**	**1.804**	0.171

E.4 Performance of 3D Point Trajectory Head

To evaluate our 3D point trajectory head, we test various methods on PointMotionBench (Zhang et al., 2026b), a 3D point forecasting benchmark. As shown in Table 23, VPM achieves the lowest ADE on HOT3D (Banerjee et al., 2024) and the best results on all three metrics on WorldTrack (Feng et al., 2025). Compared with MolmoMotion-AR under the same three-frame input, VPM lowers both ADE and FDE on every dataset and improves PWT on WorldTrack and DAVIS (Pont-Tuset

et al., 2017). These results highlight that our trajectory head produces consistent 3D tracks across diverse domains, from egocentric manipulation to general scenes.

Table 24: **Results on single-view I2V world modeling.**

Model	PAIBench-G (↑)			RBench (↑)
	Overall	Domain	Quality	Score (%)
Wan2.2-A14B	81.3	85.3	77.3	50.7
HunyuanVideo-1.5	81.7	85.9	77.6	46.0
Cosmos-Predict2.5-2B	81.2	84.6	77.9	46.4
Cosmos-Predict2.5-14B	81.1	84.0	78.1	-
Wan2.1-14B	80.2	83.6	76.8	-
Wan2.2-5B	81.0	84.6	77.4	-
Veo-3.1	82.6	87.6	77.6	56.3
Seedance-1.5-Pro	80.8	84.7	76.9	58.4
Wan2.6	81.9	85.9	77.8	**60.7**
Cosmos3-Super	82.8	87.3	**78.2**	58.1
Cosmos3-Nano	82.7	87.2	78.1	58.4
VPM (ours)	**83.0**	**87.9**	**78.2**	60.1

E.5 Performance on Single-View World Modeling

I2V world modeling. We evaluate image-to-video (I2V) generation on PAIBench-G (Zhou et al., 2026a) and RBench (Deng et al., 2026), where each model predicts future frames from a single-view image. As shown in Table 24, VPM achieves the best Overall and Domain scores on PAIBench-G and ties for the best Quality score, outperforming both open-source models and proprietary models such as Veo-3.1. On RBench, VPM ranks second, trailing only Wan2.6 by 0.6 points.

Table 25: **Results on single-view V2V world modeling.**

Model	Mode	Score (↑)
VideoPoet	V2V	29.5
VideoGPT	V2V	35.0
Magi-1	V2V	56.0
Cosmos3-Super	V2V	**59.7**
Cosmos3-Nano	V2V	50.2
VPM (ours)	V2V	52.1

V2V world modeling. We further evaluate video-to-video (V2V) generation, where each model continues a single-view input video. As shown in Table 25, VPM reaches a score of 52.1, outperforming VideoPoet (Kondratyuk et al., 2023), VideoGPT (Yan et al., 2021), and Cosmos3-Nano while remaining behind Magi-1 (Teng et al., 2025) and Cosmos3-Super. These results demonstrate that VPM remains competitive even in the single-view setting.

Table 26: **Modality-wise ablation on ReCamMaster and CMU Panoptic.** V, T, and A denote video, point-trajectory, and action supervision. The upper bound applies the evaluation pipeline to the ground-truth videos.

Method	MV	ReCamMaster (100)		CMU Panoptic (100)			
		FVD ↓	PSNR ↑	FVD ↓	PSNR ↑	MEt3R ↓	δ^{3D}_{avg} ↑
Cosmos3-Nano	✗	125.70	13.99	162.36	20.72	0.060	30.12
VPM (V)	✓	103.12	15.09	150.63	20.85	0.037	34.72
VPM (V+A)	✓	101.35	15.63	149.06	21.14	0.035	36.51
VPM (V+T)	✓	79.63	17.06	128.64	23.24	0.029	42.03
VPM (V+T+A)	✓	76.12	17.48	118.05	24.08	0.029	44.33
+ camera cond.	✓	73.44	18.01	113.40	24.25	0.027	45.70
Upper bound	–	–	–	–	–	0.009	64.12

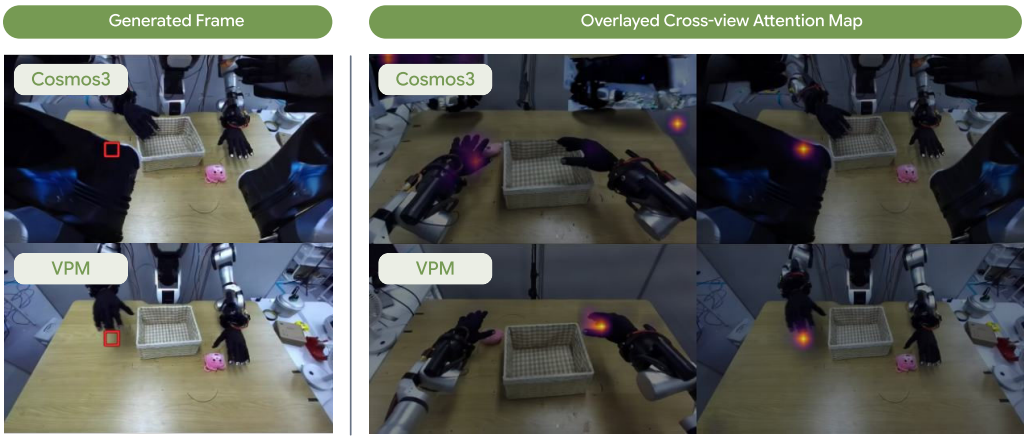


Figure 8: **Visualization of cross-view attention map.**

E.6 Additional Results on Modality-Wise Ablation Study

We repeat the modality-wise ablation of Table 6 on the zero-shot 4D generation benchmarks of Section 6.3, ReCamMaster and CMU Panoptic, which contain no robots and are not used for training. As shown in Table 26, the trends on VP-WorldBench carry over to these out-of-domain scenes. Adding point trajectories to video yields the largest gain, reducing FVD on ReCamMaster from 103.12 to 79.63 and raising δ^{3D}_{avg} on CMU Panoptic from 34.72 to 42.03, whereas adding actions yields only 101.35 and 36.51, respectively. Action supervision nonetheless improves or matches every metric, both on its own and on top of point trajectories (*e.g.*, $+2.30$ vs. $+7.31$ δ^{3D}_{avg} from trajectories), indicating that it complements point supervision even without robots in the scene. Adding camera conditioning to VPM (V+T+A) further improves all metrics, achieving the best results on both benchmarks.

E.7 Analysis on Multi-View Video Generation

To examine how VPM maintains consistency across views, we visualize its cross-view attention in Figure 8. We select a query grid in one generated view (red box) and overlay its attention map on the other views. Since Cosmos3-Nano does not natively take multi-view input, we concatenate the views into a single image. For Cosmos3-Nano, the attention scatters over the background and unrelated parts of the robot, and the arm in its generated frame is visibly distorted. In contrast, the attention of VPM lands on the same hand in each of the other views, matching the location of the query grid. This highlights that VPM learns the cross-view correspondence (An et al., 2025; Hong et al., 2024b;c; Hong & Kim, 2021; Hong et al., 2024a; Cho et al., 2021; 2022) and leverages it to generate consistent multi-view videos.

Table 27: **Clock-time analysis.** Wall-clock time (sec) to generate one clip of 64 frames at 480p per view on one GH200 GPU. Cosmos3-Nano generates the views independently, whereas VPM generates them jointly. Parentheses show the time relative to Cosmos3-Nano with the same number of views.

Method	Output	1 view	2 views	4 views
Cosmos3-Nano	Video	15.9	31.8	63.5
VPM (V)	Video	16.3 (1.03×)	38.7 (1.22×)	101.4 (1.60×)
VPM (V+T)	Video + 3D traj.	16.5 (1.04×)	39.1 (1.23×)	102.3 (1.61×)
VPM (V+T+A)	Video + 3D traj. + action	16.6 (1.04×)	39.2 (1.23×)	102.5 (1.61×)

E.8 Clock-Time Analysis

Setup. We measure the wall-clock time to generate one clip under the I2V protocol of Table 2, *i.e.*, 64 future frames at 480p per view from a single conditioning frame, on one NVIDIA GH200 GPU with BF16 precision and batch size 1. VPM (V+T+A) additionally denoises 16 action chunks

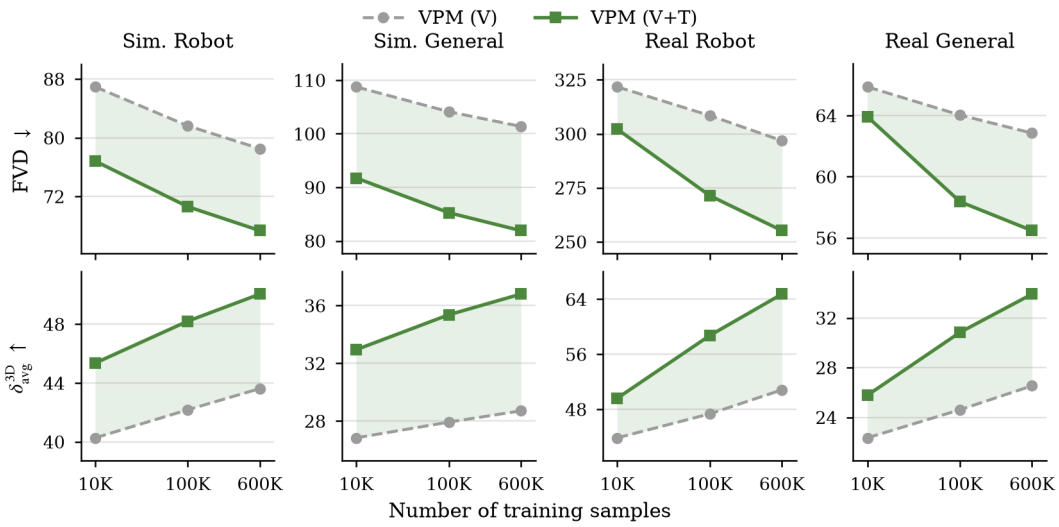


Figure 9: **Data scalability.** FVD (top) and δ^{3D}_{avg} (bottom) of VPM (V) and VPM (V+T) trained on 10K, 100K, and 600K samples. The shaded area marks the gain from point supervision.

for action generation. All models use 35 denoising steps with classifier-free guidance, following the default sampler of Cosmos3. The reported time includes VAE decoding. As in our evaluation, Cosmos3-Nano generates the views one at a time, whereas ours generates all views jointly in a single token sequence.

Results. For four views, decoding trajectories and actions adds only 0.9 s and 0.2 s, respectively, and with a single view, camera conditioning makes VPM only 3% slower than Cosmos3-Nano. The views can attend to one another, which contributes to the gains over Cosmos3-Nano.

E.9 Data Scalability

Setup. We train VPM (V) and VPM (V+T) on 10K, 100K, and 600K samples. The subsets are random subsamples of the full data that preserve their proportions in Table 8. At each scale, both models use the same training schedule.

Results. As shown in Figure 9, both models improve steadily with more training data, and point supervision improves over the video-only model at every scale. The gap widens as the data grows on every split. Specifically, the δ^{3D}_{avg} gap on Real-world Robot grows from 5.76 at 10K to 13.92 at 600K, as the robot datasets add trajectory supervision from real robot episodes. VPM (V+T) trained on 100K samples already outperforms VPM (V) trained on all 600K samples on every split, *e.g.*, 70.61 vs. 78.52 FVD on Simulation Robot, indicating that point supervision is a more data-efficient way to improve forecasting than scaling video data alone.

F Detailed Ablation Studies

In this section, we ablate the design choices of VPM. Unless otherwise stated, each variant starts from VPM (V+T) and changes a single component, keeping the training data, the three-stage schedule, and fixed iterations in Table 7. All variants are evaluated on VP-WorldBench with the I2V protocol of Table 2, and we report FVD for video fidelity and δ^{3D}_{avg} for 3D motion accuracy.

Table 28: **Ablation on camera conditioning.** Add. and Rot. denote the additive Plücker embedding and the camera-dependent rotation of queries and keys in Equation (2), and View denotes the view axis of the 4D RoPE in Equation (3).

Camera cond.			Simulation				Real-world			
			Robot (500)		General (500)		Robot (100)		General (100)	
Add.	Rot.	View	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑
✗	✗	✗	91.22	33.29	140.06	24.64	299.51	45.79	67.80	23.94
✗	✗	✓	75.17	41.70	128.41	29.08	281.67	49.25	61.43	26.99
✓	✗	✓	70.70	48.18	84.60	33.94	268.34	59.00	59.01	30.05
✓	✓	✗	70.01	48.59	85.15	33.40	265.37	61.88	58.90	31.55
✓	✓	✓	**67.35**	**50.05**	**81.92**	**36.80**	**255.19**	**64.75**	**56.48**	**33.92**

F.1 Ablation Study for Camera Embeddings

Setup. VPM conditions the generator on camera geometry through three components (Section 3.3): the additive Plücker embedding $\mathbf{We}_i$ (Add.) and the camera-dependent rotation of queries and keys $\mathbf{R}(\psi(\mathbf{e}_i))$ (Rot.) in Equation (2), and the view axis of the 4D RoPE in Equation (3) (View). Table 28 removes them individually and jointly.

Results. Removing all three components degrades every split of VP-WorldBench, most severely on Simulation General (FVD 140.06 vs. 81.92), whose cameras move during each clip. The view axis alone partially recovers performance, and adding the Plücker embedding further reduces FVD on Simulation General from 128.41 to 84.60. The camera-dependent rotation improves both metrics on every split, most notably δ^{3D}_{avg} on Real-world Robot (59.00 → 64.75). Removing only the view axis raises FVD by approximately 4% across all splits.

Table 29: **Ablation on training objectives.** $\mathcal{L}_{FM}$ is used in all rows, and $\mathcal{L}_{conf}$ is used together with $\mathcal{L}_{seq}$. The first row corresponds to VPM (V) in Table 2.

Objective			Simulation				Real-world			
			Robot (500)		General (500)		Robot (100)		General (100)	
$\mathcal{L}_{seq}$	$\mathcal{L}_{depth}$	$\mathcal{L}_{corr}$	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑
✗	✗	✗	78.52	43.62	101.37	28.71	296.83	50.83	62.85	26.54
✗	✗	✓	78.44	42.96	100.45	29.90	284.17	52.00	61.57	27.50
✓	✗	✗	74.82	43.90	91.92	30.00	275.24	54.69	59.71	27.99
✓	✓	✗	69.88	48.54	85.73	31.54	271.01	59.94	59.33	30.75
✓	✓	✓	**67.35**	**50.05**	**81.92**	**36.80**	**255.19**	**64.75**	**56.48**	**33.92**

F.2 Ablation Study for Objectives

Setup. We ablate the point-supervision objectives: the 2D position loss $\mathcal{L}_{seq}$ and the depth loss $\mathcal{L}_{depth}$ of the trajectory head in Equation (5), and the cross-view correspondence loss $\mathcal{L}_{corr}$ in Equation (6). $\mathcal{L}_{FM}$ is always used, and $\mathcal{L}_{conf}$ is kept whenever $\mathcal{L}_{seq}$ is used, since its target depends on the predicted 2D position. The first row of Table 29, trained with $\mathcal{L}_{FM}$ only, corresponds to VPM (V) in Table 2.

Results. Combining all objectives performs best on every split. $\mathcal{L}_{corr}$ alone has limited effect, except for FVD on Real-world Robot (296.83 → 284.17). $\mathcal{L}_{seq}$ mainly improves FVD (*e.g.*, 101.37 → 91.92 on Simulation General), and adding $\mathcal{L}_{depth}$ substantially raises δ^{3D}_{avg} (*e.g.*, 54.69 → 59.94 on Real-world Robot). Adding $\mathcal{L}_{corr}$ on top further improves every split, with the largest δ^{3D}_{avg} gain on Simulation General (31.54 → 36.80).

Table 30: **Ablation on training stages.** Iterations of a removed stage are reassigned to Stage 3, keeping the total number of iterations fixed.

Stage			Simulation				Real-world			
			Robot (500)		General (500)		Robot (100)		General (100)	
1	2	3	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑
✗	✗	✓	70.45	48.50	85.34	34.29	262.40	60.86	58.11	31.75
✗	✓	✓	69.83	49.15	83.27	34.51	259.42	62.00	58.02	32.80
✓	✗	✓	77.50	39.51	99.81	29.38	292.53	49.77	62.49	27.04
✓	✓	✓	**67.35**	**50.05**	**81.92**	**36.80**	**255.19**	**64.75**	**56.48**	**33.92**

F.3 Ablation Study for Training Stages

Setup. VPM is trained in three stages: Stage 1 adapts the generator to multi-view input and camera conditioning with $\mathcal{L}_{\text{FM}}$, Stage 2 warms up the trajectory head with $\mathcal{L}_{\text{track}}$ while the generator is frozen, and Stage 3 jointly optimizes both with $\mathcal{L}_{\text{forecast}}$. We remove Stage 1, Stage 2, or both, and reassign their iterations to Stage 3, so that every variant is trained for the same 12K iterations in total.

Results. As shown in Table 30, the full curriculum performs best on every split. Removing Stage 2 is the most harmful, performing worse than training with Stage 3 alone on every split, with the largest drop on Real-world Robot (δ^{3D}_{avg} 64.75 → 49.77). Removing Stage 1 leads to a smaller but consistent drop, most pronounced on Real-world Robot (δ^{3D}_{avg} 64.75 → 62.00).

Table 31: **Ablation on the generator layer.** The trajectory head reads query and key features from the l-th self-attention layer of the generator.

	Simulation				Real-world			
	Robot (500)		General (500)		Robot (100)		General (100)	
Layer l	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑	FVD↓	δ^{3D}_{avg}↑
6	72.81	45.92	96.28	31.10	277.11	57.04	60.26	28.75
12	69.45	48.30	85.76	34.90	266.16	61.21	58.30	31.88
18	**67.35**	**50.05**	**81.92**	**36.80**	**255.19**	**64.75**	**56.48**	**33.92**
24	68.76	49.16	84.96	35.28	259.84	63.80	58.65	31.60

F.4 Ablation Study for Layers

Setup. The trajectory head reads query and key features from a single self-attention layer l of the generator, following the DiT (Peebles & Xie, 2023) architecture. We vary $l \in \{6, 12, 18, 24\}$, where $l = 18$ is the default.

Results. As shown in Table 31, attaching the head to the 18th layer performs best on every split. Shallow layers perform clearly worse, *e.g.*, FVD on Simulation General rises from 81.92 to 96.28 with $l = 6$. Moving the head deeper to $l = 24$ causes a slight drop, increasing FVD by about 2–4% across splits.

F.5 Ablation Study for Refinement

Setup. The trajectory head refines its estimates over M iterations with a transformer whose attention is factorized over space, time, and views. We vary $M \in \{1, 2, 4, 8\}$ and remove each attention type from the refiner.

Results. As shown in Table 32, performance consistently improves up to $M = 4$, *e.g.*, FVD on Simulation General decreases from 87.23 at $M = 1$ to 81.92 at $M = 4$. Further increasing to $M = 8$ changes every metric by less than 1 while doubling the cost of the head. Among the attention types, removing cross-view attention causes the largest degradation on every split, *e.g.*,

Table 32: **Ablation on the trajectory refiner.** (a) Number of refinement iterations M. (b) Attention types of the refiner, with $M = 4$.

	Simulation				Real-world			
	Robot (500)		General (500)		Robot (100)		General (100)	
Setting	FVD↓	δ^{3D}_{avg} ↑	FVD↓	δ^{3D}_{avg} ↑	FVD↓	δ^{3D}_{avg} ↑	FVD↓	δ^{3D}_{avg} ↑
(a) Refinement iterations								
$M = 1$	69.46	47.11	87.23	33.44	267.42	61.05	59.83	30.19
$M = 2$	68.77	49.00	83.25	35.75	260.49	62.18	57.07	33.20
$M = 4$	67.35	**50.05**	**81.92**	**36.80**	255.19	64.75	**56.48**	**33.92**
$M = 8$	**67.14**	49.96	82.63	36.75	**255.01**	**64.92**	56.66	33.58
(b) Refiner attention								
w/o spatial	70.10	47.54	86.79	34.25	267.59	61.16	57.31	32.50
w/o temporal	70.87	48.21	88.54	34.28	268.79	59.64	59.18	30.77
w/o cross-view	74.95	46.49	91.14	29.89	272.50	57.09	61.07	29.95
Full	**67.35**	**50.05**	**81.92**	**36.80**	**255.19**	**64.75**	**56.48**	**33.92**

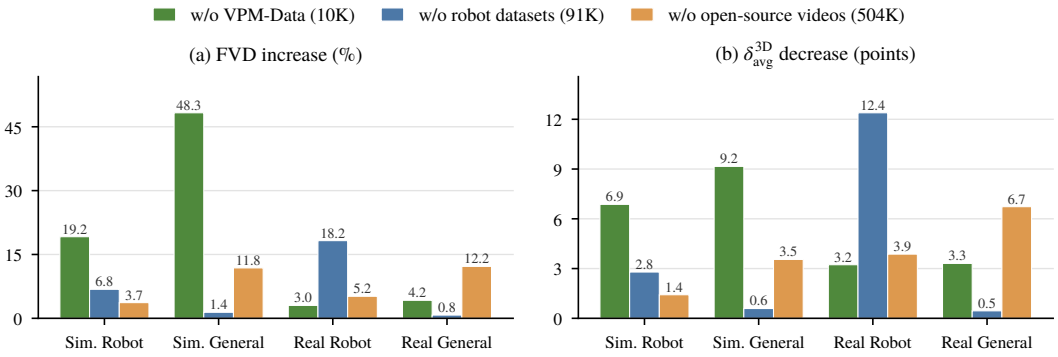


Figure 10: **Ablation on training data sources.** Relative FVD increase (a) and δ^{3D}_{avg} decrease (b) of VPM (V+T) when one source is removed from training, measured against the model trained on all sources. Higher is worse.

increasing FVD by 11.3% and lowering δ^{3D}_{avg} from 36.80 to 29.89 on Simulation General. Removing temporal attention increases FVD more than removing spatial attention on every split, while the two are comparable in δ^{3D}_{avg} on the simulation splits.

F.6 Ablation Study for Training Data

Setup. Our training data combines three sources (Table 8): VPM-Data (10K samples), robot manipulation datasets with kinematics-grounded point tracks (91K), and open-source videos (504K). We train VPM (V+T) while leaving out one source at a time and report the change in FVD and δ^{3D}_{avg} with respect to the model trained on all sources in Figure 10.

Results. Each source mainly benefits the evaluation domain it covers. Removing VPM-Data causes the largest degradation on the simulation splits, especially on Simulation General (+48.3% FVD and −9.16 δ^{3D}_{avg}), since it is the only source that pairs synchronized multi-view urban videos with persistent 3D trajectories. Removing VPM-Data also lowers δ^{3D}_{avg} on Real-world General by 3.31, suggesting that its accurate simulated trajectories transfer to real human motion. Removing the robot datasets mainly affects Real-world Robot (+18.2% FVD and −12.39 δ^{3D}_{avg}), and removing the open-source videos mainly affects Real-world General (+12.2% FVD and −6.73 δ^{3D}_{avg}). Removing any source degrades every split, indicating that the three sources are complementary.

G Qualitative Results

G.1 Multi-View Video and Point Trajectory Prediction

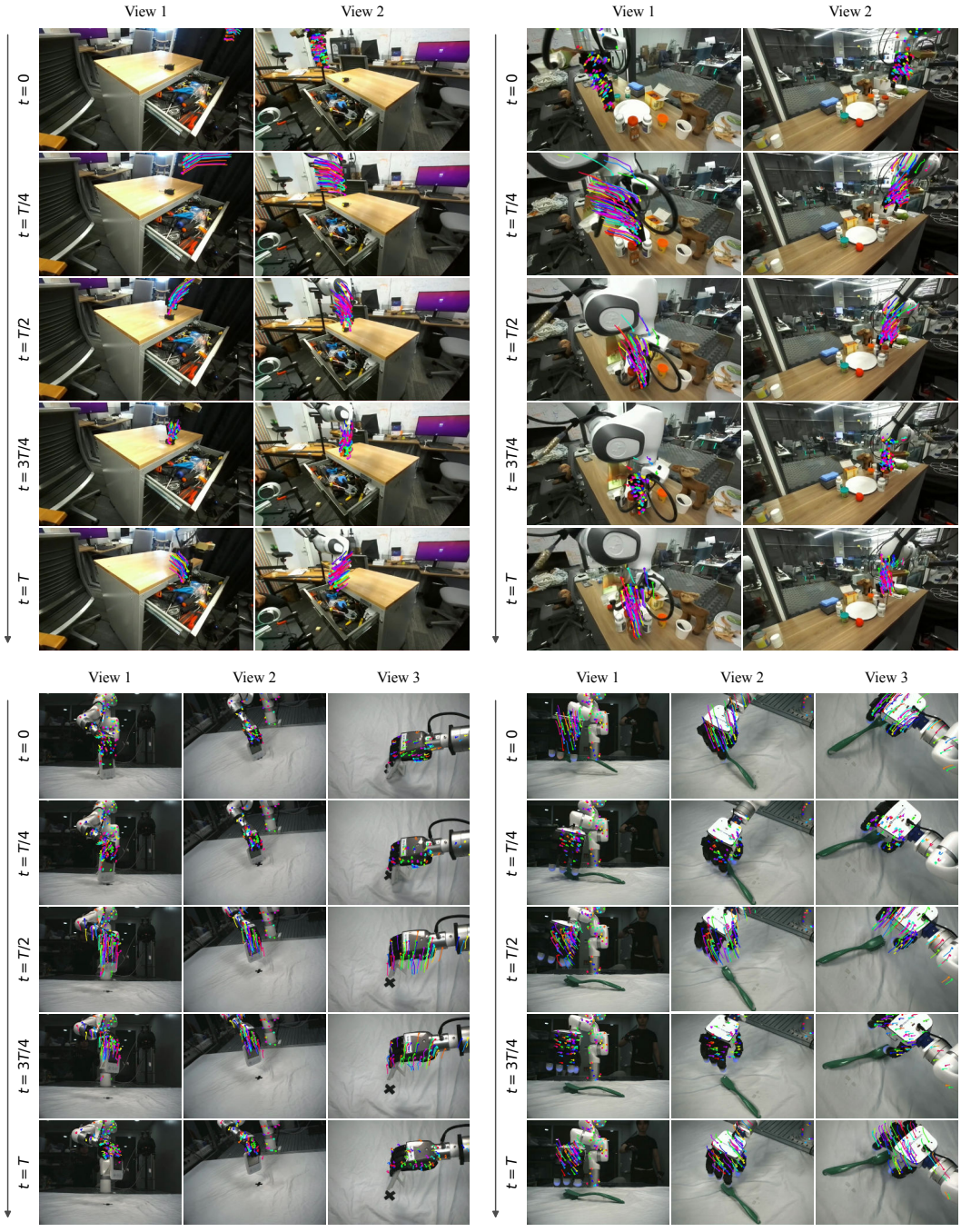


Figure 11: **Qualitative results of multi-view prediction.** Predicted video with overlaid predicted point track for two-view (top) and three-view (bottom) scenes. Conditioning frames are omitted.

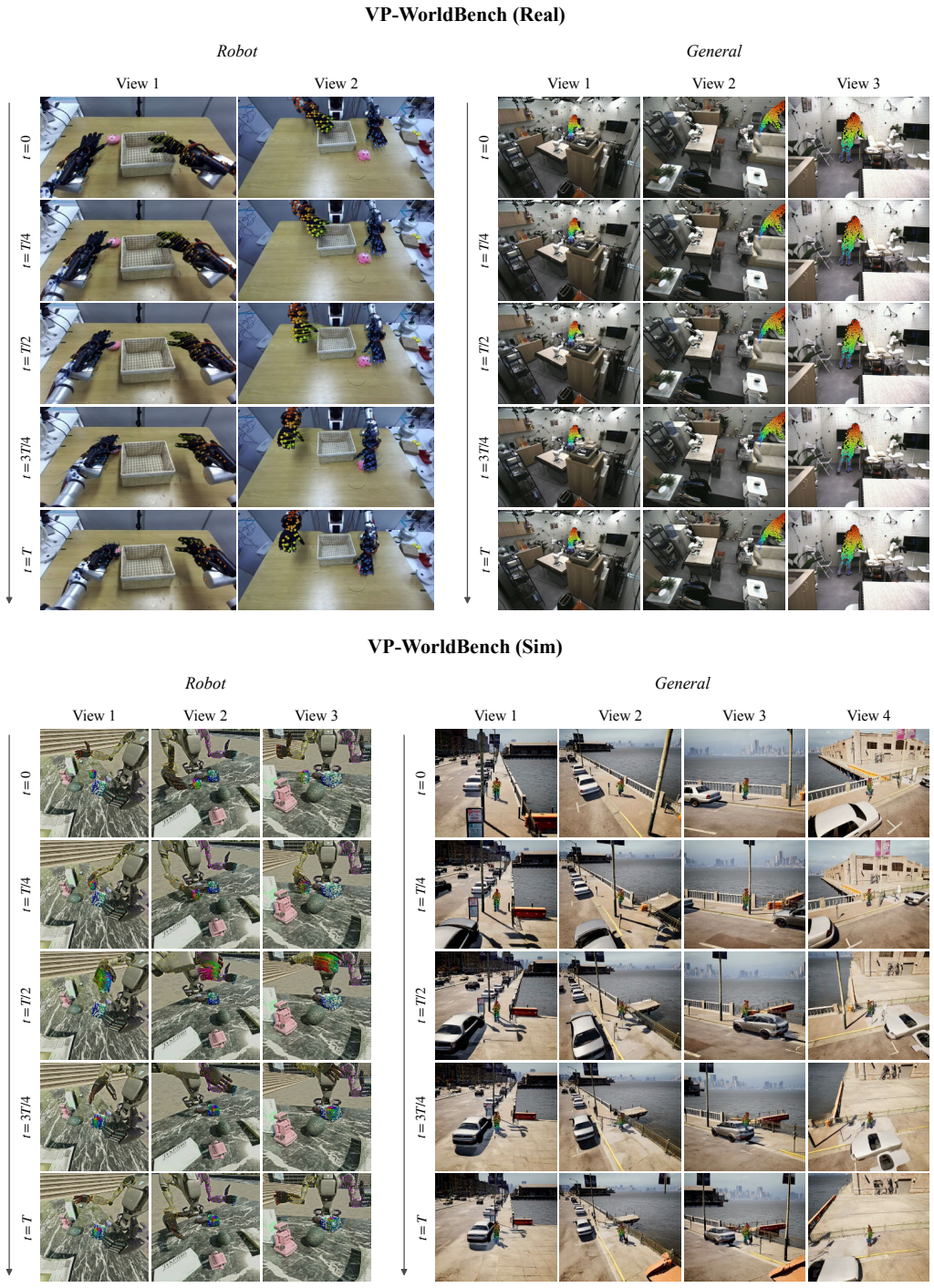


Figure 12: **Additional qualitative results of multi-view prediction.** Predicted video with overlaid predicted point track on robot and general scenes from VP-WorldBench (Real, top) and VP-WorldBench (Sim, bottom).

G.2 Predicted vs. Ground-Truth Videos

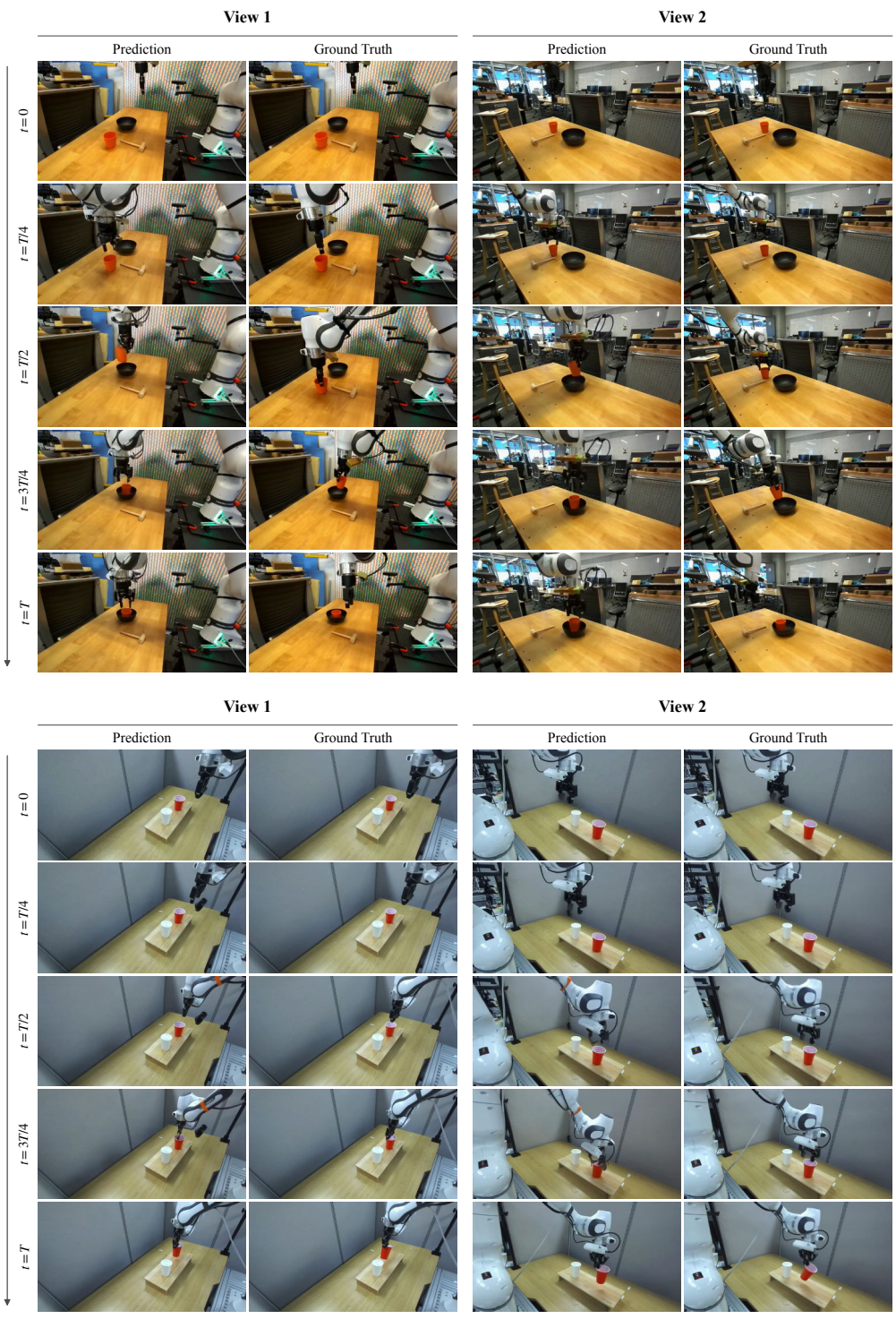


Figure 13: **Predicted vs. ground-truth videos.** Each view shows the prediction next to the ground truth. Conditioning frames are omitted.

G.3 Example Rollouts of Robot Manipulation Tasks

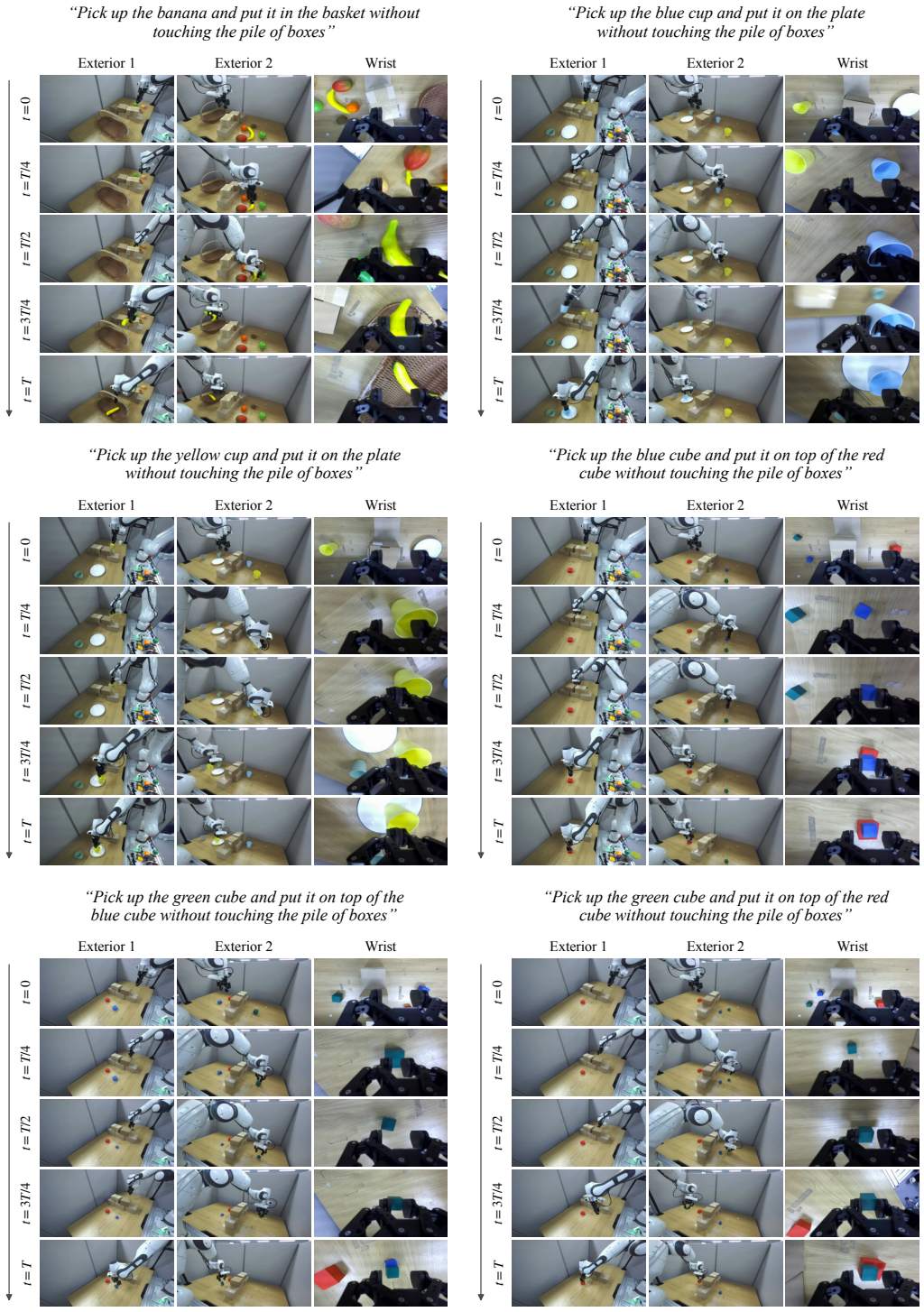


Figure 14: **Example rollouts of robot manipulation tasks.** Successful rollouts of our policy on six tasks, shown from two exterior cameras and a wrist camera. Rows are time steps sampled uniformly over each episode of length T.