POINTACTION: 3D POINTS AS UNIVERSAL ACTION REPRESENTATIONS FOR ROBOT CONTROL

Anonymous authors
Paper under double-blind review

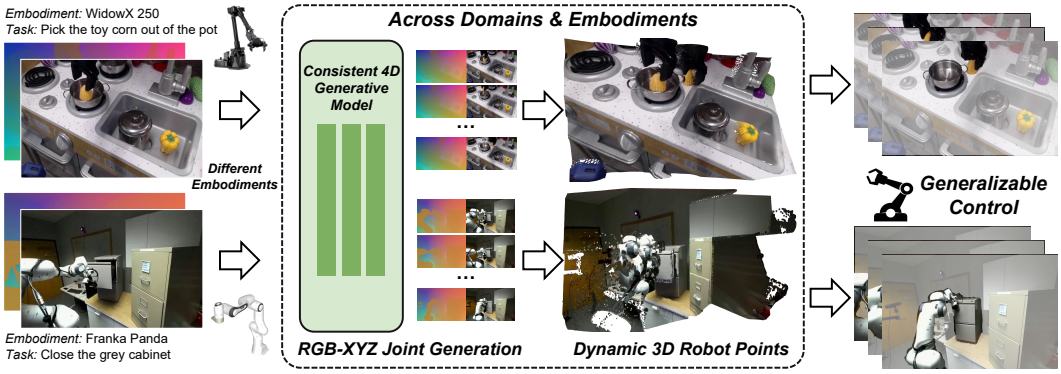


Figure 1: **A geometric interface between video prediction and robot control.** PointAction jointly predicts RGB and XYZ rollouts, extracts the robot's predicted 3D points, and decodes them into embodiment-specific controls. The video model is post-trained for the deployment setting and the action decoder is trained with paired demonstrations.

ABSTRACT

World Action Models (WAMs) leverage the broad visual dynamics captured by pretrained video diffusion models, offering a promising path toward generalizable robot manipulation. However, RGB-only rollouts leave 3D motion and fine-grained spatial constraints implicit, making action grounding ambiguous. Meanwhile, scaling action supervision across tasks and embodiments remains costly. We present **PointAction**, a framework that uses dynamic 3D points as an action interface between video prediction and robot control. PointAction adapts a foundation video model to jointly predict future RGB frames and dynamic 3D pointmaps. This interface expresses proposed robot motion in a common geometric form, which an embodiment-specific diffusion decoder translates into controls. Geometric supervision from videos enables the rollout model to learn beyond paired action data, while target-robot video adaptation and decoder training support transfer with limited demonstrations. Experiments demonstrate strong 4D generation quality on robot scenes, competitive manipulation performance in simulation, and adaptation to two real robot arms unseen during pretraining. These results support 3D point dynamics as a common representation for connecting video prediction to robot control. Robot videos and interactive 4D visualizations are available on our project page.

1 INTRODUCTION

Building generalist robot policies capable of solving diverse manipulation tasks across environments and embodiments remains a central goal in robot learning. Recent Vision-Language-Action (VLA) models (Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024a; 2025; Bjorck et al., 2025) combine semantic priors from pretrained vision-language models with large-scale robot datasets (O'Neill et al., 2024; Khazatsky et al., 2024; Walke et al., 2023), enabling language-conditioned manipulation. However, semantic understanding alone does not explicitly describe how scenes evolve through

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

motion and physical interaction. Predicting such changes offers a complementary source of information for robot control.

To incorporate these dynamics, World Action Models (WAMs) use pretrained video generators to predict future scene rollouts for action generation (Du et al., 2023; Black et al., 2024b; Hu et al., 2025; Kim et al., 2026). Existing methods either jointly model observations and actions or decode actions from predicted rollouts. Despite their promise, two challenges remain. First, RGB-dominant representations leave 3D motion and fine-grained spatial constraints implicit, requiring the action module to infer motion from appearance. Second, learning this mapping relies on paired observation–action supervision, which is costly to collect and tied to robot-specific control spaces. A representation that exposes geometric motion while supporting supervision from videos could help address both challenges.

We argue that dynamic 3D points provide a natural action interface between world prediction and robot control. They represent proposed robot motion in a common geometric form, making it available for decoding into each robot's control space. As a training target, pointmaps can be obtained from videos through stereo reconstruction or monocular depth estimation, extending the available supervision beyond robot action labels. Building on this insight, we propose PointAction, a framework that learns this action interface through joint RGB-XYZ prediction and point-to-action decoding.

Concretely, PointAction factorizes video-to-action learning into a shared video-to-point model and an embodiment-specific point-to-action decoder. We adapt a foundation robot video model (Chen et al., 2025a) to jointly generate RGB frames and pixel-aligned XYZ pointmaps, then extract the predicted robot points for a lightweight diffusion decoder. The point-based interface separates predicted motion from each robot's control parameterization. For a new embodiment, target-robot videos adapt the generator and paired demonstrations train the decoder, reusing the pretrained model and common geometric representation.

We evaluate PointAction through 4D generation, simulated manipulation, and adaptation to real robot arms unseen during pretraining. Our main contributions are threefold:

- **3D points as an action interface.** We represent predicted robot motion with dynamic 3D points, separating its geometric description from embodiment-specific control commands.
- **Scalable supervision with embodiment-specific decoding.** Joint RGB-XYZ prediction learns from videos with geometric supervision, while a lightweight decoder learns the robot-specific mapping to actions.
- **Strong generation and manipulation results.** Experiments show strong 4D generation quality, competitive simulation performance, and transfer to two additional arms through adaptation. Ablations support robot-centric geometry as the decoder input.

2 Related Work

Vision-language and world action models. Vision-language-action models connect pretrained semantic representations to robot controls (Zitkovich et al., 2023; Kim et al., 2024; Octo Model Team et al., 2024; Black et al., 2024a; 2025; Bjorck et al., 2025). Visual prediction extends this supervision through subgoal images in CoT-VLA (Zhao et al., 2025), video pretraining in GR-1 and GR-2 (Wu et al., 2024; Cheang et al., 2024), and joint visual-action prediction in WorldVLA (Cen et al., 2025). UniPi (Du et al., 2023) and SuSIE (Black et al., 2024b) separate visual planning from action prediction; VPP (Hu et al., 2025) decodes actions from video-model features. UVA (Li et al., 2025a) learns a joint video-action latent representation with separate diffusion heads, allowing action inference without video decoding. UWM (Zhu et al., 2025) couples video and action diffusion with modality-specific noise levels, supporting policy learning and dynamics modeling. Cosmos Policy (Kim et al., 2026) incorporates states, actions, and values into a video model. PointAction instead passes generated robot geometry to a separate decoder, making the motion-to-control interface explicit.

Motion interfaces for robot control. In AVDC, Ko et al. (2024) combine optical flow from generated RGB videos with initial depth to recover object transformations and robot commands.

Gen2Act (Bharadhwaj et al., 2025) conditions a learned policy on generated human videos, with auxiliary point-track prediction. Dreamitate (Liang et al., 2025) tracks tools in generated stereo videos to recover 3D trajectories for execution. PointAction jointly predicts RGB and dense XYZ pointmaps, then learns action decoding from the robot's predicted surface geometry, avoiding a separate reconstruction or tool-pose tracking stage.

ATM (Wen et al., 2023) uses predicted 2D point trajectories for policy learning; Track2Act (Bharadhwaj et al., 2024) recovers object motion from tracks and refines execution with a residual policy. Im2Flow2Act (Xu et al., 2024) uses object flow to connect human-video generation and simulated robot learning. General Flow (Yuan et al., 2025) predicts 3D object-point trajectories from human RGB-D videos for a geometric controller. 3DFlowAction (Zhi et al., 2025) learns 3D object flow and translates it into actions through optimization. Motion Tracks (MT-π) (Ren et al., 2025) predicts 2D hand/end-effector tracks across two views to recover 6DoF trajectories. PointAction shares the motion of a common motion interface, but generates dense robot-surface point sets, without persistent point identities, and learns their translation to commands.

3D manipulation policies. Act3D (Gervet et al., 2023) predicts end-effector poses using 3D feature fields; 3D Diffuser Actor (Ke et al., 2025) conditions action diffusion on 3D scene features, while DP3 (Ze et al., 2024) uses compact point-cloud representations. Our decoder consumes a generated sequence of future robot geometry, using points to represent proposed motion as well as spatial structure.

Geometric video generation. 4DNeX (Chen et al., 2025b), World-consistent Video Diffusion (Zhang et al., 2025), and Geo4D (Jiang et al., 2025) connect generation to explicit geometry. For manipulation, Liu et al. (2026) combine generated geometry with pose estimation, while TesserAct (Zhen et al., 2025) models RGB, depth, and normals. PointAction couples joint RGB-XYZ prediction with robot-point action decoding, so the generated geometry directly drives control.

3 POINTACTION

3.1 A GEOMETRIC INTERMEDIATE REPRESENTATION

Given an RGB observation o_t, robot state s_t, and instruction l, a world action model predicts a visual rollout $\tilde{o}$ and an action sequence $\tilde{a}$:

$$(\tilde{o}, \tilde{a}) \sim \pi^{\mathrm{WAM}}(\cdot \mid o_t, s_t, l). \tag{1}$$

We study how to represent the intermediate rollout for action decoding. Rather than requiring the decoder to infer robot motion from appearance alone, PointAction predicts pixel-aligned XYZ pointmaps alongside RGB. For frame k, $u_k(p) \in \mathbb{R}^3$ stores the camera-frame coordinate of visible pixel p. These are dense per-frame surface samples rather than persistent point tracks.

Let S segment robot masks from the generated RGB rollout. We express the two-stage policy as

$$\begin{aligned} (\tilde{o}, \tilde{u}) &\sim \pi_\theta^{\mathrm{4DVM}}(\cdot \mid o_t, l), \\ \tilde{a} &\sim \pi_\psi^{\mathrm{DEC}}(\cdot \mid R(\tilde{u}, S(\tilde{o})), s_t), \end{aligned} \tag{2}$$

where $\pi_\theta^{\mathrm{4DVM}}$ is the joint RGB-XYZ video model, π_ψ^{DEC} is the action decoder, and R retains robot-surface points. The generator models the full scene, whereas the decoder consumes only the predicted robot geometry, so object context influences control through the generated robot motion. Table 1 contrasts this factorization with direct VLAs and RGB-based WAMs.

3.2 JOINT RGB-XYZ VIDEO PREDICTION

We initialize the generator from Large Video Planner (LVP) (Chen et al., 2025a). Following 4DNeX (Chen et al., 2025b), a shared, frozen video VAE independently encodes RGB and XYZ sequences into latents z^o and z^u. After normalizing XYZ latents using dataset statistics, we concatenate the streams along their spatial width:

$$z^{\mathrm{joint}} = \mathrm{WidthConcat}(z^o, z^u) \in \mathbb{R}^{T' \times C \times h \times 2w}. \tag{3}$$

Table 1: **Policy factorizations.** PointAction exposes generated robot geometry to an embodiment-specific action decoder. Individual VLA and WAM architectures may add further prediction targets.

Design	Intermediate prediction	Mapping to actions
Direct VLA	Visual-language features	Action head conditioned on observations and robot state
RGB WAM	Future RGB rollout or video latents	Inverse-dynamics or jointly trained action model
PointAction	Joint RGB-XYZ rollout; selected robot point sets	Embodiment-specific diffusion decoder conditioned on point features and initial state

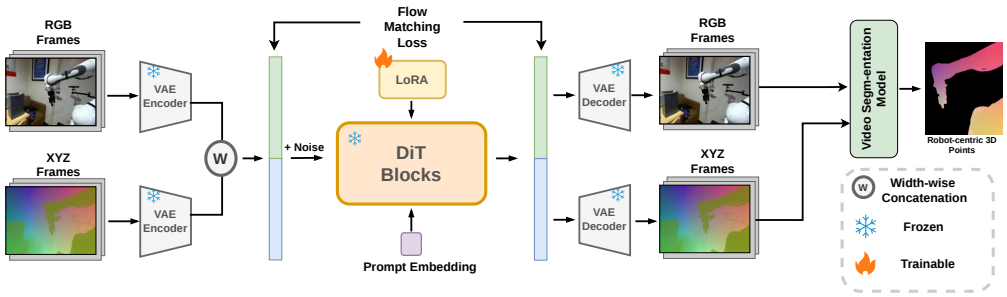


Figure 2: **Joint RGB-XYZ prediction.** A pretrained video backbone is adapted to jointly model RGB and camera-frame XYZ sequences. Width-wise latent concatenation retains the pretrained input-channel structure.

The pretrained channel structure is unchanged. Modality embeddings distinguish the streams, and repeated spatial rotary position embeddings align their coordinate indexing. Self-attention associates corresponding appearance and geometric features across the two streams.

Learning joint appearance and geometry. Training uses flow matching (Lipman et al., 2023) with independently noised history and future segments, following Diffusion Forcing (Chen et al., 2024). Let $\hat{z}$ and $\tilde{z}$ denote the history and future portions of a joint RGB-XYZ latent sequence. At noise level $\tau \in [0, 1]$, we form $\tilde{z}_\tau = (1 - \tau)\tilde{z} + \tau\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$, and predict the velocity toward noise:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}\left[\left\|v_\theta(\tilde{z}_\tau, \hat{z}_{\tau'}, l, \tau) - (\epsilon - \tilde{z})\right\|_2^2\right]. \tag{4}$$

The expectation is over training sequences, random history lengths, noise, and noise levels. History noise τ' is sampled independently and set to zero with probability 0.5. Both modalities participate in the same prediction objective, so the transformer must model their joint evolution rather than attach an independent geometry estimator after RGB generation. LoRA (Hu et al., 2022) updates the transformer while the text encoder and VAE remain frozen; further architecture and training details appear in Appendix A.

Pretraining and downstream adaptation. The joint generator is pretrained on approximately 75K manipulation trajectories: 50K DROID (Khazatsky et al., 2024) and 25K BridgeData V2 (Walke et al., 2023) examples, covering Franka Panda and WidowX arms. DROID depth is recomputed from stereo pairs with FoundationStereo (Wen et al., 2025); monocular BridgeData V2 uses Depth Anything 3 (DA3) (Lin et al., 2025) estimates for depth and camera intrinsics. Corrupted camera records and failed demonstrations are filtered out. These data provide geometric supervision without requiring low-level action labels for the generator.

Downstream transfer adapts *both* stages: the video model receives 1,000 post-training steps for the RoboCasa setting or 500 steps for each real arm, and a robot-specific decoder is trained from scratch. One decoder covers all ten seen RoboCasa tasks. The pretrained generator and the point representation are shared across all settings.

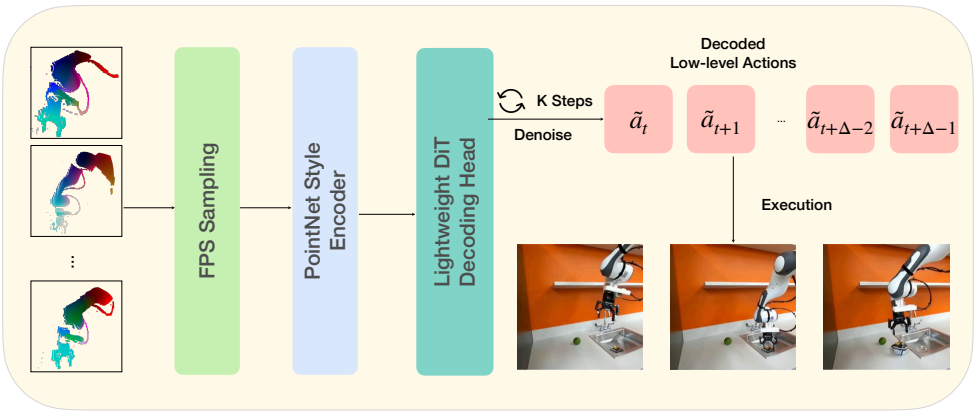


Figure 3: **Point-to-action decoding.** Robot-surface points are sampled and encoded per frame. A conditional diffusion transformer predicts the action sequence using the point features and the initial robot state.

Training and sampling configuration. Generator training uniformly resamples each trajectory to 49 frames at 832×480 resolution. LoRA adaptation (rank 128, $\alpha = 64$) updates approximately 209M parameters, optimized with AdamW at a learning rate of 10^{-4} and a global batch size of 16. At inference, a UniPC sampler uses 40 denoising steps with text classifier-free guidance scale 2.5. The action decoder uses six transformer layers with hidden dimension 256 and four attention heads. This pairs a large pretrained video model for prediction with a lightweight model for action translation. Full hyperparameters are listed in Appendix A.

3.3 Robot-centric point-to-action decoding

SAM 3 (Carion et al., 2025) segments each generated RGB sequence using the text prompt "robot". We retain XYZ values at the masked pixels, filter outliers using the 1st–99th percentiles, and sample $N = 512$ points per frame with farthest-point sampling. This produces a sequence of point sets, each encoded by a shared three-layer PointNet-style MLP (Qi et al., 2017). A conditional diffusion transformer (Peebles & Xie, 2023), inspired by point-based policies (Ze et al., 2024), concatenates per-frame point features with noisy action tokens. The initial robot state and diffusion timestep condition its AdaLN layers. Training uses noise prediction over a 100-step schedule; inference uses ten DDIM steps (Song et al., 2021). Random point dropout of up to 20% and Gaussian jitter ($\sigma = 0.05$) augment decoder training, which runs for 3,000 epochs with AdamW (learning rate 5×10^{-4}, batch size 32) and a cosine schedule.

Writing $g = \Phi(\mathcal{P}_{0:48})$ for the sequence of encoded point sets and $a^{(j)}$ for a noised action sequence at diffusion step j, the decoder minimizes

$$\mathcal{L}_{\text{dec}} = \mathbb{E}_{a,\epsilon',j}\left[\left\|\epsilon' - \epsilon_\psi(a^{(j)}, g, s_t, j)\right\|_2^2\right], \tag{5}$$

where a is the demonstrated action sequence and $\epsilon' \sim \mathcal{N}(0, I)$. Point features condition the temporally corresponding action tokens, while s_t provides the initial configuration needed to interpret the proposed motion for a particular arm. The decoder generates the entire sequence jointly, and because sampling and encoding operate on per-frame sets, point identities need not persist across the rollout.

Execution protocol. Each 49-frame rollout is decoded into one 49-step action sequence and executed open-loop; training demonstrations are uniformly resampled to 49 steps. Generation, segmentation, and decoding take approximately 372 s, 9 s, and 10 ms per sequence, respectively, on one NVIDIA B200.

Table 2: **RoboCasa365 success rates (%).** ID and OOD-Env average ten seen tasks; OOD-Task averages five unseen tasks. The five VLA/WAM/PointAction methods use 100 rollouts per task and regime; their per-task results appear in Table 9. ATM is included as an additional point-trajectory baseline. Bold marks the best result.

Method	ID	OOD-Env	OOD-Task
ATM (Wen et al., 2023)	24.3	5.6	0.0
GR00T N1.7 (Bjorck et al., 2025)	44.5	37.6	8.6
$\pi_{0.5}$ (Black et al., 2025)	39.8	35.2	6.8
VPP (Hu et al., 2025)	34.5	32.2	7.4
Cosmos Policy (Kim et al., 2026)	45.2	42.9	14.0
PointAction	**47.7**	**44.1**	**17.0**

4 Experimental Setup

Our evaluation asks three complementary questions: whether the point interface supports competitive task success, whether it can be adapted to arms absent from joint RGB-XYZ pretraining, and which information the decoder needs from the predicted rollout. Manipulation outcomes test the complete pipeline; decoder-input ablations probe the intermediate representation; generation metrics examine the quality of the predicted signal. These measurements are complementary because perceptually accurate video alone need not yield accurate robot commands.

Manipulation. In the VLA/WAM comparison on RoboCasa365 (Nasiriany et al., 2026), methods receive the same single third-person observation view and training split: 100 demonstrations for each of ten seen tasks. We evaluate seen tasks in familiar environments (ID), seen tasks under environment and initial-state changes (OOD-Env), and five unseen tasks in familiar environments (OOD-Task), Baselines include GR00T N1.7 (Bjorck et al., 2025), $\pi_{0.5}$ (Black et al., 2025), VPP (Hu et al., 2025), Cosmos Policy (Kim et al., 2026), and ATM (Wen et al., 2023). All methods except ATM are evaluated with 100 rollouts per task and regime; ATM is reported from aggregate results.

The xArm7 and YAM arms are absent from pretraining. We use 50 demonstrations and 100 evaluation rollouts per task on xArm7, and 20 demonstrations and 20 rollouts per task on YAM, with the camera fixed relative to the robot. Per-arm tasks and baselines are detailed in Appendix C.

Generation. Held-out DROID and BridgeData V2 trajectories assess RGB quality (PSNR, SSIM, FVD (Unterthiner et al., 2018)) and geometry (depth AbsRel, δ_1, and point-cloud Chamfer L_1) against estimated reference depth. Baselines are TesserAct (Zhen et al., 2025), 4DNeX (Chen et al., 2025b), LVP, Wan 2.1 (Wan et al., 2025), and a cascade applying StreamVGGT (Zhuo et al., 2025) to our generated RGB.

5 Results

5.1 Simulation: Task Performance and Generalization

Comparison with learned policies. On the ten seen RoboCasa365 tasks, PointAction achieves the highest success rates, 47.7% ID and 44.1% OOD-Env (Table 2). It leads Cosmos Policy, the strongest baseline, by 2.5 and 1.2 percentage points (pp), and GR00T N1.7 by 3.2 and 6.5 pp.

Unseen tasks and point-trajectory comparison. On five unseen tasks, PointAction reaches 17.0%, compared with 14.0% for Cosmos Policy, 8.6% for GR00T N1.7, and 6.8% for $\pi_{0.5}$. ATM (Wen et al., 2023), which conditions its policy on predicted 2D point trajectories, obtains $24.3/5.6/0.0\%$ across the three regimes and falls furthest behind under environment and task shifts. Appendix B gives the full per-task breakdown.

From predicted points to execution. Figure 4 traces one rollout through the pipeline. The generated RGB frames and the predicted point cloud describe the same arm motion, and the simulated execution of the decoded actions follows that motion over the chunk. The geometry specifies where the arm should move, and the decoder converts that motion into commands.

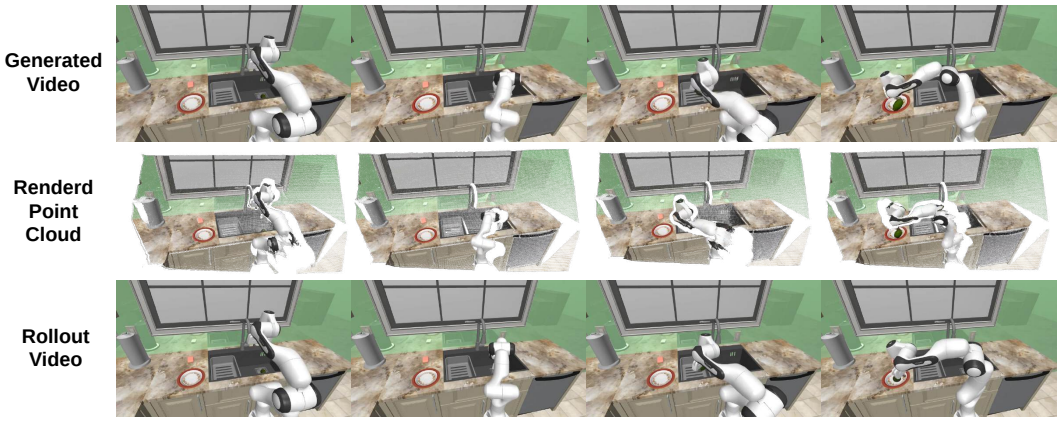


Figure 4: **Predicted geometry and execution on RoboCasa365.** Generated RGB (top), the predicted XYZ rendered as a point cloud (middle), and the simulated execution of the decoded actions (bottom), at four matched timesteps.

Table 3: **Real-world success rates (%).** Both arms are unseen during pretraining; PointAction adapts the generator and trains an action decoder for each arm. All methods use 50 demonstrations/task on xArm7 (100 evaluation trials/task), or 20 demonstrations/task on YAM (20 trials/task). The baseline versions differ between setups.

xArm7 Method	Pick & place	Stack cubes	Stack cups	Mean
GR00T N1.7 (Bjorck et al., 2025)	30.0	7.0	7.0	14.7
$\pi_{0.5}$ (Black et al., 2025)	42.0	12.0	14.0	22.7
PointAction	**67.0**	**28.0**	**34.0**	**43.0**
YAM Method	Stack cubes	Pick pens	Insert cups	Mean
GR00T N1.5 (Bjorck et al., 2025)	0.0	20.0	15.0	11.7
π_0 (Black et al., 2024a)	0.0	10.0	15.0	8.3
PointAction	**20.0**	**60.0**	**50.0**	**43.3**

Per-task behavior. Gains vary across tasks. PointAction is strongest on OpenDrawer, with $70/66\%$ ID/OOD-Env success versus $57/59\%$ for Cosmos Policy, but trails GR00T N1.7 on PickPlaceSinkToCounter ($39/36\%$ versus $59/59\%$). CloseBlenderLid remains nearly unsolved by all methods (at most 5% in either regime), possibly because of its long demonstrations and occlusion.

5.2 Adaptation to two real robot arms

The two arms differ in hardware, visual domain, and task suite: xArm7 is observed by a ZED 2i stereo camera and evaluated on pick-and-place and two stacking tasks, whereas YAM uses a RealSense D455 and is evaluated on cube stacking, pen picking, and cup insertion. Both follow the adaptation protocol and budgets in Sections 3.2 and 4.

On xArm7, PointAction obtains 43.0% average success, versus 22.7% for $\pi_{0.5}$ and 14.7% for GR00T N1.7, and leads on all three tasks (Table 3). On YAM, it averages 43.3% ($20/60/50\%$ for stacking cubes, picking pens, and inserting cups), compared with 11.7% for GR00T N1.5 and 8.3% for π_0. These results support the point interface on two new embodiments. Hardware and task details are in Appendix C.

Per task, the largest xArm7 gain is in pick-and-place (67% versus 42% for $\pi_{0.5}$). The two stacking tasks, which require precise relative placement, remain harder for every method, yet PointAction still more than doubles the best baseline on each. On YAM, PointAction succeeds in 12 of 20 pen-picking trials and 10 of 20 cup insertions, and it alone completes any cube-stacking trial.

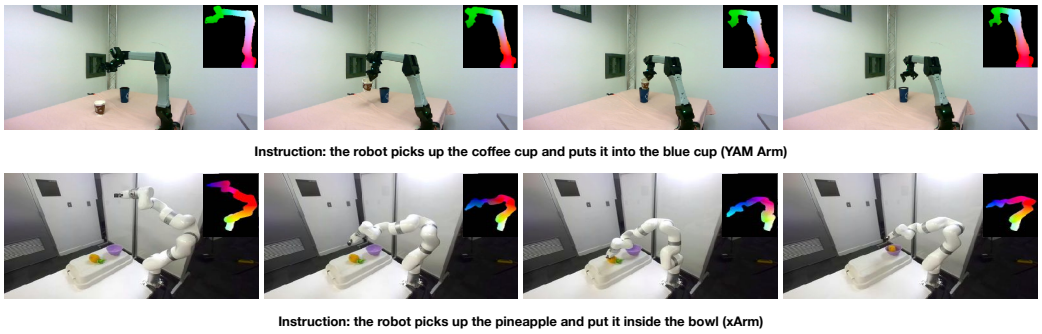


Figure 5: **Qualitative rollouts on adapted embodiments.** YAM Insert Cups (top) and xArm7 Pick-and-Place (bottom). Both arms are unseen during pretraining; each receives generator post-training and decoder training. Insets show segmented robot points predicted by the 4D model.

Table 4: **Decoder-input ablation.** Average success (%) on the ten seen simulation tasks. Robot+scene uses separate encoders; full scene uses one encoder. DA3 estimates geometry from our generated RGB.

Decoder input	ID	OOD-Env
RGB only	25.1	20.3
RGB + XYZ (robot only)	37.2	30.9
XYZ only: full scene	27.1	19.4
XYZ only: robot + scene	40.3	33.7
XYZ only: robot only, DA3 source	28.4	21.7
XYZ only: robot only, joint prediction	**47.7**	**44.1**

5.3 What information should the action decoder consume?

Table 4 compares decoder inputs under the same simulation setup. Robot-only XYZ yields 47.7% ID success, compared with 25.1% for RGB alone and 37.2% for RGB plus robot XYZ. Among XYZ inputs, full-scene points achieve 27.1% and separately encoded robot-plus-scene points achieve 40.3%. Robot-only input therefore outperforms adding scene points, supporting a compact geometric interface. Replacing joint XYZ predictions with DA3 estimates from the generated RGB reduces ID success to 28.4%, showing the value of jointly predicted geometry for downstream control.

Interpreting the robot-only bottleneck. The generator observes the scene and instruction before proposing robot motion, so masking the decoder input does not remove scene context from the policy; it focuses the final translation stage on the moving embodiment. Organizing the same information into separate robot and scene streams raises ID success from 27.1% to 40.3%, yet robot-only input remains best at 47.7%. Making robot motion directly accessible to the decoder thus matters as much as providing geometry.

5.4 Geometry quality

On held-out DROID and BridgeData V2 clips, PointAction achieves the best RGB and geometry metrics among the evaluated models (Table 5). Its visual quality is on par with LVP, and its geometry is markedly more accurate than that of the 4D generators or of StreamVGGT applied to the same generated RGB.

Geometry against simulator ground truth. Because real-world reference depth is estimated, we also evaluate against exact simulator depth on the RoboCasa365 test split (Table 6). We compare the jointly generated XYZ with two post-hoc routes applied to the same generated RGB: DA3, a monocular depth estimator, and MegaSaM, a structure-and-motion method for monocular dynamic video that LVP uses as its depth source. Framewise, joint prediction attains an AbsRel of 0.118 and a Chamfer L_1 of 0.151, compared with 0.187 and 0.327 for MegaSaM and 0.198 and 0.361 for DA3. Over full clips, a single frame-0 anchor yields an AbsRel of 0.108, whereas DA3 and MegaSaM

Table 5: **4D generation on held-out DROID and BridgeData V2 clips.** Geometry is evaluated against estimated reference geometry. "RGB + StreamVGGT" reconstructs our generated RGB. Bold marks the best result.

	RGB			Depth		Points
Method	PSNR↑	SSIM↑	FVD↓	AbsRel↓	δ_1↑	CD-L_1↓
TesserAct (Zhen et al., 2025)	12.225	0.487	746	0.403	0.641	0.389
4DNeX (Chen et al., 2025b)	13.858	0.542	818	0.348	0.681	0.370
LVP (Chen et al., 2025a)	19.613	0.816	330	–	–	–
Wan 2.1 14B (Wan et al., 2025)	14.532	0.674	671	–	–	–
RGB + StreamVGGT (Zhuo et al., 2025)	–	–	–	0.382	0.675	0.341
PointAction	**19.631**	**0.821**	**320**	**0.176**	**0.890**	**0.122**

Table 6: **Geometry on the RoboCasa365 test split against simulator ground truth.** Cascaded estimators use our generated RGB. The clip column covers 100 trajectories: baselines receive per-frame oracle re-anchoring, whereas PointAction uses a single frame-0 anchor.

	Framewise				Clip
Geometry source	AbsRel↓	δ_1↑	δ_2↑	CD-L_1↓	AbsRel↓
MegaSaM (Li et al., 2025b) on our RGB	0.187	0.775	0.877	0.327	0.162
DA3 (Lin et al., 2025) on our RGB	0.198	0.755	0.867	0.361	0.164
Joint RGB-XYZ (PointAction)	**0.118**	**0.872**	**0.904**	**0.151**	**0.108**

reach 0.164 and 0.162 even with per-frame oracle re-anchoring, indicating consistent scale across each clip. Together with the DA3 decoder ablation, these results show that jointly learned XYZ is both more accurate and more useful for control than geometry reconstructed post hoc from the same RGB. SAM 3 masks also remain reliable on generated frames, with 0.942 mIoU compared with 0.959 on ground-truth rendered frames.

6 Discussion and Limitations

PointAction connects video generation to robot control through predicted robot geometry. Factorizing the policy into RGB-XYZ prediction and point-to-action decoding separates two learning problems with different supervision requirements: the generator learns from videos with estimated geometry and no action labels, while a lightweight decoder learns each robot's mapping from proposed motion to commands. The same pretrained generator initializes adaptation to new arms, and changing only the decoder input substantially changes task success, underscoring the role of the interface itself.

An inspectable interface. The intermediate rollout exposes the proposed robot motion before execution, making it possible to attribute a failure to generation, point selection, or action translation. Evaluating these stages separately under controlled geometric and mask perturbations would help identify where additional supervision or feedback is most valuable.

Scaling the interface. The factorization suggests two routes to broader transfer: enlarging the action-free video corpus used to pretrain the generator and sharing a decoder across arms with similar kinematics so that a new arm needs fewer paired demonstrations.

Limitations. In simulation, the margins over Cosmos Policy are small, and the YAM evaluation uses 20 trials per task. All methods receive a single third-person view, so the comparison does not reflect the multi-camera setups VLA policies often use, and our camera-frame geometry assumes a fixed camera–robot configuration. Real-world geometry targets come from estimated depth, and the robot-only decoder input carries no explicit contact information. Adapting to a new arm requires post-training the generator and training a decoder. Finally, each 49-step chunk is executed open-loop after roughly six minutes of planning, which precludes reactive control. Faster generation, closed-loop replanning, and controlled viewpoint studies are natural next steps.

AI USE STATEMENT

In this work, we used generative AI tools (ChatGPT and Claude) to draft and edit parts of the paper, including text that describes and interprets our experimental results, to restructure the manuscript, and to check grammar. We have not used generative AI tools to design the method or experiments, implement methods, generate data, produce experimental results, or create figures; all reported results come from our own experiments. We have reviewed all AI-assisted content and checked reported numbers and citations against our records and the original sources, and we take responsibility for the final content of this work. The proposed method itself builds on pretrained models: the generator is initialized from Large Video Planner (LVP), SAM 3 segments the robot in generated videos, and FoundationStereo and Depth Anything 3 provide depth supervision for pretraining. Their roles are described explicitly in the method and implementation details.

REPRODUCIBILITY STATEMENT

The model architecture, training objectives, and hyperparameters are described in Appendix A. The evaluation regimes and per-task simulation results appear in Appendix B; real-robot hardware, demonstration budgets, and evaluation protocols appear in Appendix C. Additional geometry analyses and deployment failure modes are reported in Appendices D and E. The supplementary videos show task executions on both xArm7 and YAM, together with generated RGB-XYZ rollouts.

ETHICS STATEMENT

This work studies learning representations for robot manipulation. The current open-loop execution and long inference latency limit recovery from unexpected events, so the reported results do not establish safe autonomous deployment around people. Reuse or release of demonstrations and derived geometry should respect the source datasets' terms and the privacy of anyone visible in recordings.

Upon acceptance, we will release all code, model weights, the full implementation, and the datasets collected for this work.

REFERENCES

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2Act: Predicting point tracks from internet videos enables generalizable robot manipulation. *arXiv preprint arXiv:2405.01527*, 2024. URL https://arxiv.org/abs/2405.01527.

Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2Act: Human video generation in novel scenarios enables generalizable robot manipulation. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 3936–3951. PMLR, 2025. URL https://proceedings.mlr.press/v305/bharadhwaj25a.html.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024a.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Rich Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=c0chJTSbci.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=vlhoswksBO.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Boyuan Chen, Tianyuan Zhang, Haoran Geng, Kiwhan Song, Caiyi Zhang, Peihao Li, William T Freeman, Jitendra Malik, Pieter Abbeel, Russ Tedrake, et al. Large video planner enables generalizable robot control. *arXiv preprint arXiv:2512.15840*, 2025a.

Zhaoxi Chen, Tianqi Liu, Long Zhuo, Jiawei Ren, Zeng Tao, He Zhu, Fangzhou Hong, Liang Pan, and Ziwei Liu. 4dnex: Feed-forward 4d generative modeling made easy. *arXiv preprint arXiv:2508.13154*, 2025b.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3D: 3D feature field transformers for multi-task robotic manipulation. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pp. 3949–3965. PMLR, 2023. URL https://proceedings.mlr.press/v229/gervet23a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/hu25g.html.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=mSiN7i0BYH.

Zeren Jiang, Chuanxia Zheng, Iro Laina, Diane Larlus, and Andrea Vedaldi. Geo4d: Leveraging video generators for geometric 4d scene reconstruction, 2025. URL https://arxiv.org/abs/2504.07961.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3D diffuser actor: Policy diffusion with 3D scene representations. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 1949–1974. PMLR, 2025. URL https://proceedings.mlr.press/v270/ke25a.html.

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.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B. Tenenbaum. Learning to act from actionless videos through dense correspondences. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b2d4051f03a7038a2771dfbbe5c7b54e-Abstract-Conference.html.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025a. URL https://unified-video-action-model.github.io/.

Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. MegaSaM: Accurate, fast, and robust structure and motion from casual dynamic videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10486–10496, 2025b.

Junbang Liang, Ruoshi Liu, Ege Ozguroglu, Sruthi Sudhakar, Achal Dave, Pavel Tokmakov, Shuran Song, and Carl Vondrick. Dreamitate: Real-world visuomotor policy learning via video generation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 3943–3960. PMLR, 2025. URL https://proceedings.mlr.press/v270/liang25b.html.

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 T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Zeyi Liu, Shuang Li, Eric Cousineau, Siyuan Feng, Benjamin Burchfiel, and Shuran Song. Geometry-aware 4d video generation for robot manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=18gC6pZVVc.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tQJYKwc3n4.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017. URL https://api.semanticscholar.org/CorpusID:5115938.

Juntao Ren, Priya Sundaresan, Dorsa Sadigh, Sanjiban Choudhury, and Jeannette Bohg. Motion tracks: A unified representation for human-robot transfer in few-shot imitation learning. *arXiv preprint arXiv:2501.06994*, 2025. URL https://arxiv.org/abs/2501.06994.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *ArXiv*, abs/1812.01717, 2018. URL https://api.semanticscholar.org/CorpusID:54458806.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

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

Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching. *CVPR*, 2025.

Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2023. URL https://arxiv.org/abs/2401.00025.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. In *8th Annual Conference on Robot Learning*, 2024. URL https://arxiv.org/abs/2407.15208.

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

Chengbo Yuan, Chuan Wen, Tong Zhang, and Yang Gao. General flow as foundation affordance for scalable robot learning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 1541–1566. PMLR, 2025. URL https://proceedings.mlr.press/v270/yuan25a.html.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Qihang Zhang, Shuangfei Zhai, Miguel Angel Bautista Martin, Kevin Miao, Alexander Toshev, Joshua Susskind, and Jiatao Gu. World-consistent video diffusion with explicit 3d modeling. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 21685–21695, 2025.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 1702–1713, 2025.

Haoyu Zhen, Qiao Sun, Hongxin Zhang, Junyan Li, Siyuan Zhou, Yilun Du, and Chuang Gan. Tesseract: Learning 4d embodied world models. *CoRR*, abs/2504.20995, 2025. doi: 10.48550/ARXIV.2504.20995. URL https://doi.org/10.48550/arXiv.2504.20995.

Hongyan Zhi, Peihao Chen, Siyuan Zhou, Yubo Dong, Quanxi Wu, Lei Han, and Mingkui Tan. 3DFlowAction: Learning cross-embodiment manipulation from 3D flow world model. *arXiv preprint arXiv:2506.06199*, 2025. URL https://arxiv.org/abs/2506.06199.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. *arXiv preprint arXiv:2602.02214*, 2026.

Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4d visual geometry transformer. *arXiv preprint arXiv:2507.11539*, 2025.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

APPENDIX OVERVIEW

This appendix complements the main paper as follows. Appendix A provides extended implementation and training details for the joint RGB-XYZ video model and the action decoder, including hyperparameters and per-task baseline configurations. Appendix B expands the simulation evaluation (Table 2) with an illustration of the three ID/OOD regimes, per-task success rates, and per-task rollout visualizations across all 15 RoboCasa365 tasks. Appendix C details the cross-embodiment real-world setup of Section 5.2: hardware, task suites, and demonstration budgets for both the xArm7 and the YAM arm, together with task illustrations. Appendix D provides additional analyses of the 4D generation backbone: qualitative rollouts illustrating temporal and geometric consistency and head-to-head comparisons with 4D-generation baselines. Appendix E discusses the dominant failure modes we observe in deployment. The supplementary ZIP contains twelve real-robot execution videos, two qualitative rollouts per task: xArm7 pick-and-place, stack cubes, and stack cups, and YAM Stack Cubes, Pick Pens, and Insert Cups. It also includes generated RGB videos and the corresponding pixel-aligned XYZ pointmap videos for four DROID and BridgeData V2 scenes.

A EXTENDED IMPLEMENTATION DETAILS

A.1 MODEL ARCHITECTURE

Our joint RGB-XYZ model retains the original LVP (Chen et al., 2025a) image-to-video (I2V) backbone—comprising the text encoder, video VAE, and transformer—while extending it to concurrently model RGB frames and XYZ pointmaps. Both RGB and XYZ videos are encoded with the shared VAE. The resulting XYZ latents are normalized to a standard normal distribution using precomputed dataset statistics. The two latent tensors are then concatenated along the width dimension, effectively doubling the spatial token width and the maximum token budget. To enable the shared transformer to disambiguate between modalities, we introduce a lightweight modality embedding using separate learnable vectors for RGB and XYZ. We additionally repeat the Rotary Position Embedding (RoPE) along the width dimension to accommodate the doubled layout. The output latent is split along the width dimension into RGB and XYZ streams, which are decoded separately. The model is trained using the standard latent flow-matching objective. During LoRA (Hu et al., 2022) fine-tuning, we update approximately 209M parameters while keeping the text encoder and VAE frozen.

For the DiT-based action decoder, each frame's robot-centric point cloud is encoded into a per-frame feature vector using a shared, 3-layer PointNet-style (Qi et al., 2017) MLP. The diffusion model operates directly on the full 49-step action sequence. We concatenate the noisy action tokens with the per-frame point features, project them to a DiT hidden dimension of 256, add learnable positional embeddings, and process them through a stack of AdaLN-modulated Transformer blocks. Global conditioning—comprising the diffusion timestep and the first-frame robot state—is injected exclusively via AdaLN layers to cleanly separate modality roles. The prediction head outputs the added noise (ϵ) following a standard 100-timestep diffusion schedule. During inference, we employ a DDIM-style (Song et al., 2021) sampler for 10 steps, allowing the decoder to predict the full action sequence in a single shot.

A.2 4D GENERATION TRAINING DETAILS

The joint RGB-XYZ generation model is trained on a combined dataset of 75K trajectories: 50K from DROID (Khazatsky et al., 2024) and 25K from BridgeData V2 (Walke et al., 2023). All training videos are uniformly downsampled to 49 frames and spatially resized to 832×480. Detailed training hyperparameters are summarized in Table 7. During inference, we utilize a UniPC sampler with 40 denoising steps and a default text classifier-free guidance (CFG) scale of 2.5. Generation of one 49-frame chunk takes 372 seconds on a single NVIDIA B200 GPU. SAM 3 segmentation adds 9 seconds and action decoding approximately 10 milliseconds, for a serial planning latency of roughly 381 seconds excluding robot execution.

Flow-matching objective. Each sequence of joint latents z^{joint} is randomly split into a history context $\hat{z}$ (of length m) and a future trajectory $\tilde{z}$. Independent noise levels are applied to each. For

Table 7: **RGB-XYZ model training hyperparameters.**

Hyperparameter	Value
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	5×10^{-2}
β_1, β_2	$0.9, 0.95$
Global batch size	16
LoRA rank / α / dropout	128 / 64 / 0.0
XYZ latent statistics (μ, σ)	$(-0.227444, 1.437663)$

the future frames, given a noise level $\tau \in [0, 1]$ and standard Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, the noisy target is constructed via linear interpolation:

$$\tilde{z}_\tau = (1 - \tau)\, \tilde{z} + \tau\, \epsilon. \tag{6}$$

The history $\hat{z}$ is diffused with an independent noise level τ', which we set to $\tau' = 0$ (clean context) with 50% probability to encourage robust conditioning. The joint transformer backbone v_θ is then optimized to predict the flow field $v = (\epsilon - \tilde{z})$, conditioned on l:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{z^{joint}, \epsilon, \tau, \tau', m} \left[\left\| v_\theta(\tilde{z}_\tau, \hat{z}_{\tau'}, l, \tau) - v \right\|_2^2 \right]. \tag{7}$$

A.3 Action Decoder Training (Simulation)

For evaluation on RoboCasa365, we post-train the 4D model for 1,000 iterations on the training split and train one DiT-based action decoder from scratch across all ten seen tasks. Per-task data, baseline implementations, and training configurations are summarized below.

Decoder objective. Per-frame robot-centric points $\tilde{u}_{\text{robo}}$ are downsampled by FPS to $\tilde{\mathcal{P}} = \text{FPS}(\tilde{u}_{\text{robo}}, N)$ with N=512 and encoded by a PointNet-style MLP Φ. Let a denote the ground-truth action sequence. At diffusion step j we inject Gaussian noise $\epsilon' \sim \mathcal{N}(0, I)$ to obtain a noisy action $a^{(j)}$, and train the DiT-based decoder ϵ_ψ via the standard ϵ-prediction loss conditioned on the extracted point features and the initial robot state s_t:

$$\mathcal{L}_{\text{dec}} = \mathbb{E}_{a, \epsilon', j} \left[\left\| \epsilon' - \epsilon_\psi\big(a^{(j)}, \Phi(\tilde{\mathcal{P}}), s_t, j\big) \right\|_2^2 \right]. \tag{8}$$

Datasets. Each simulation task comprises 100 expert episodes collected via human teleoperation. For PointAction, trajectories are uniformly resampled to 49 frames prior to training, including demonstrations longer than one nominal generation window. At inference, the 49 predicted frames condition a single 49-action chunk executed open-loop.

Baseline implementations and observation parity. The simulation VLA results use $\pi_{0.5}$ (Black et al., 2025) and GR00T N1.7 (Bjorck et al., 2025). We use the official codebases for Cosmos Policy (Kim et al., 2026) and VPP (Hu et al., 2025); Cosmos Policy is initialized from `nvidia/Cosmos-Policy-RoboCasa-Predict2-2B` and fine-tuned on the same training split. VPP fine-tunes its pretrained Stable Video Diffusion backbone and trains its action head from scratch. The compared policies use the same task-training split and single third-person camera view.

Segmentation and point processing. SAM 3 masks select robot points from generated XYZ frames. Before sampling, masked points are clipped using the 1st–99th percentile to suppress outliers; training also uses the dropout and jitter in Table 8. SAM 3 attains 0.959 mIoU on ground-truth rendered frames and 0.942 on generated frames.

Training configurations. Detailed action decoder hyperparameters are listed in Table 8.

Table 8: **Simulation post-training and action-decoder configuration.** A single decoder is trained across the ten seen RoboCasa365 tasks.

Hyperparameter	Value
Video model post-training	1,000 iterations
Decoder architecture	DiT: 6 layers, 256 hidden units, 4 heads
Decoder training	3,000 epochs; batch size 32
Optimizer	AdamW; learning rate 5×10^{-4}
Weight decay	10^{-4}
Learning-rate schedule	Cosine decay; 1,000 warmup steps
Point augmentation	Dropout up to 0.2; jitter $\sigma = 0.05$
Points per frame	512 (farthest-point sampling)
Action sampling	10 DDIM steps; 49-action output

A.4 Baseline Details

This section expands the compact baseline list in the main paper (Section 4).

Robot Manipulation (Simulation).

- **GR00T N1.7** (Bjorck et al., 2025): a VLA combining a flow-matching diffusion transformer with synthetic-data integration for cross-embodiment robotic control.
- $\boldsymbol{\pi_{0.5}}$ (Black et al., 2025): a general-purpose VLA combining a pretrained VLM backbone (Beyer et al., 2024) with a flow-matching action expert for continuous control across embodiments.
- **ATM** (Wen et al., 2023): uses predicted point trajectories to condition policy learning; its aggregate simulation results are included as a point-trajectory comparison.
- **VPP** (Hu et al., 2025): a decoupled World Action Model built on Stable Video Diffusion (SVD) (Blattmann et al., 2023); the SVD backbone is pretrained on robot and human video, and an inverse dynamics model decodes actions from the SVD latent.
- **Cosmos Policy** (Kim et al., 2026): built on Cosmos-Predict2 (Ali et al., 2025), jointly generating video and action encoded as latent frames within the video diffusion process.

Cross-Embodiment Real-World. On the xArm7 setup the VLA baselines are $\boldsymbol{\pi_{0.5}}$ (Black et al., 2025) and **GR00T N1.7** (Bjorck et al., 2025); on the YAM setup we use $\boldsymbol{\pi_0}$ and **GR00T N1.5**. All baselines are fine-tuned on the same per-arm teleoperation set used by PointAction and evaluated under the same rollout budget (see Appendix C).

4D Generation.

- **TesserAct** (Zhen et al., 2025): predicts RGB-DN (RGB, depth, and normal) sequences for robotic scenes, refined by normal integration into a 4D representation.
- **4DNeX** (Chen et al., 2025b): a feed-forward 4D generation model built on the Wan 2.1 architecture, trained on large-scale 4D datasets to jointly synthesize RGB and XYZ.
- **Large Video Planner (LVP)** (Chen et al., 2025a): a foundation video model tailored for robotic tasks and trained across human and robot interaction video.
- **Wan 2.1 (14B)** (Wan et al., 2025): a large-scale pretrained video diffusion model for zero-shot generation across visual domains.
- **PointAction (RGB) + StreamVGGT** (Zhuo et al., 2025): a cascaded variant in which our model's generated 2D RGB videos are passed through StreamVGGT, a streaming 4D reconstruction model, and the extracted geometry is compared against our direct joint XYZ predictions.

B Detailed Simulation Results

This appendix provides the full per-task numerical breakdown that supports the averages in Table 2 of the main paper, together with visualizations of the evaluation regimes and per-task rollouts.

B.1 Evaluation Settings

Fig. 6 illustrates the three evaluation regimes used in Tables 2 and 9: **ID** (seen scene and seen task), **OOD-Env** (unseen scene, seen task), and **OOD-Task** (seen scene, unseen task).

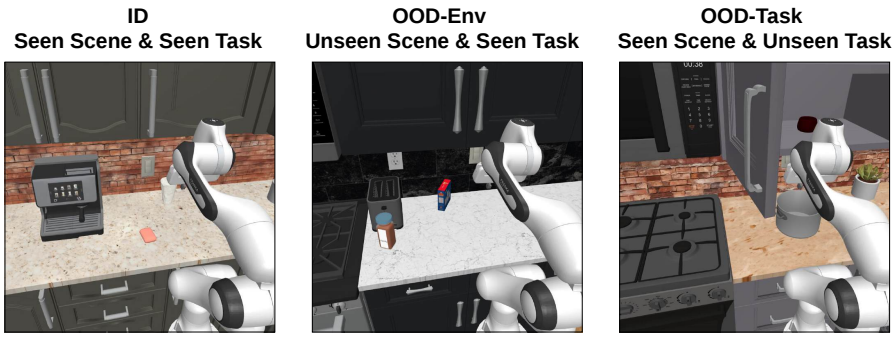


Figure 6: **Visualization of simulation evaluation settings.**

B.2 Per-Task Success Rates

Table 9 expands Table 2 into per-task success rates across all 15 RoboCasa365 tasks (10 seen, 5 unseen). The top block reports both ID and OOD-Env success per seen task, while the bottom block reports OOD-Task success on five completely unseen tasks.

Table 9: **Per-task RoboCasa365 success rates (%).** Seen-task cells report ID | OOD-Env; unseen-task cells report OOD-Task. Each method–task–regime cell uses 100 rollouts with randomized initial states. Baseline versions and values match Table 2. ATM per-task results are not available. Bold marks the best result in each regime.

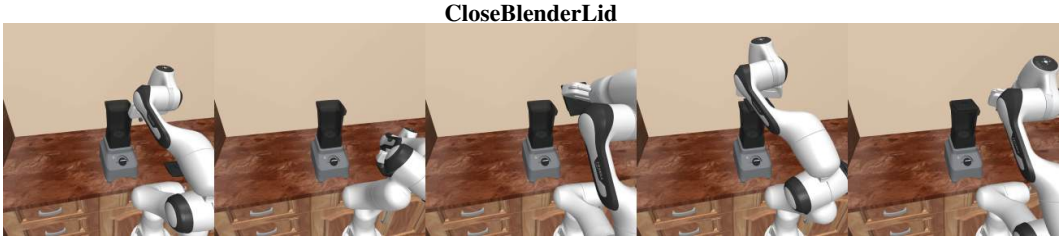
Task	GR00T N1.7	$\pi_{0.5}$	VPP	Cosmos Policy	PointAction
Seen tasks: ID | OOD-Env					
CloseBlenderLid	**5** | 0	4 | 3	3 | 2	**5** | **5**	1 | 0
CloseToasterOvenDoor	54 | 45	**74** | 58	54 | 48	59 | 57	67 | **63**
OpenCabinet	35 | 31	39 | 28	29 | 28	35 | 34	**42** | **39**
OpenDrawer	48 | 39	34 | 38	27 | 29	57 | 59	**70** | **66**
PickPlaceCounterToCabinet	38 | 31	24 | 28	29 | 26	40 | 36	**46** | **42**
PickPlaceCounterToStove	32 | 30	13 | 15	19 | 17	**37** | **35**	31 | 25
PickPlaceSinkToCounter	**59** | **59**	30 | 27	30 | 27	48 | 44	39 | 36
SlideDishwasherRack	69 | 64	**79** | **70**	62 | 61	63 | 59	66 | 63
TurnOnElectricKettle	60 | 50	56 | 45	53 | 50	62 | 57	**68** | **63**
TurnOnSinkFaucet	45 | 27	45 | 40	39 | 34	46 | 43	**47** | **44**
Seen-task mean	44.5 | 37.6	39.8 | 35.2	34.5 | 32.2	45.2 | 42.9	**47.7** | **44.1**
Unseen tasks: OOD-Task					
CloseCabinet	8	4	5	13	**17**
CloseDishwasher	9	6	8	11	**14**
CoffeeServeMug	1	1	6	**9**	2
OpenDishwasher	12	7	4	18	**25**
SlideOvenRack	13	16	14	19	**27**
Unseen-task mean	8.6	6.8	7.4	14.0	**17.0**

Task heterogeneity. Outcomes vary across tasks: PointAction is strong on OpenDrawer but below GR00T N1.7 on PickPlaceSinkToCounter, and CloseBlenderLid remains near-unsolved for every method (at most 5% ID success), possibly because of its long demonstration window.

B.3 Per-Task Rollouts

Each labeled row shows sampled keyframes from one example simulated execution, ordered from left to right. The first ten tasks are seen in training; the final five are unseen tasks.

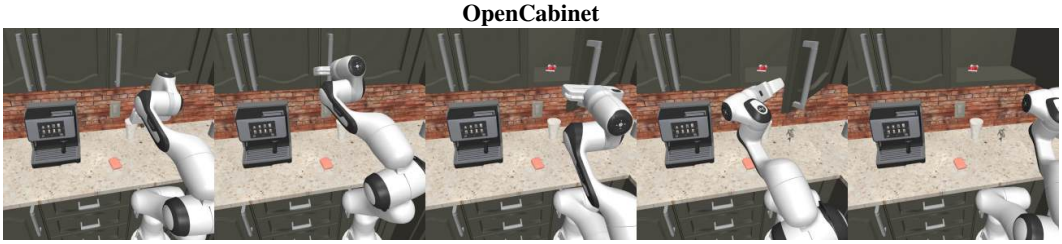
CloseBlenderLid



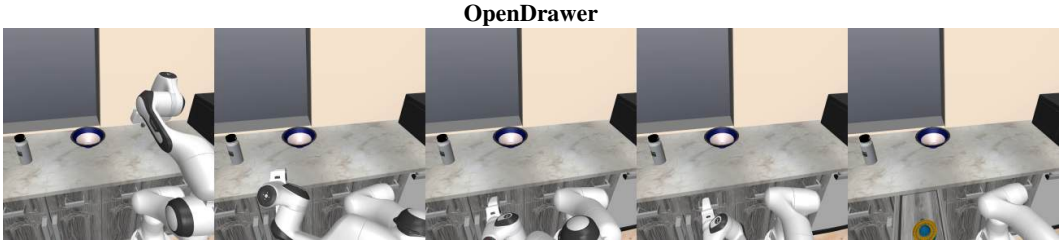
CloseToasterOvenDoor



OpenCabinet



OpenDrawer



PickPlaceCounterToCabinet



Figure 7: **Example RoboCasa365 executions, group 1 of 3.** Seen tasks; keyframes run from left to right.

PickPlaceCounterToStove



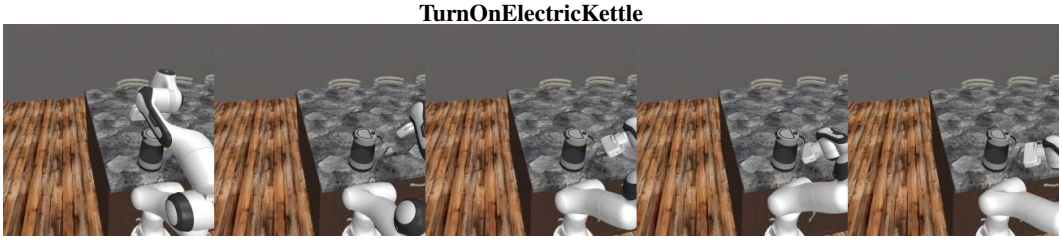
PickPlaceSinkToCounter



SlideDishwasherRack



TurnOnElectricKettle



TurnOnSinkFaucet



Figure 8: **Example RoboCasa365 executions, group 2 of 3.** Seen tasks; keyframes run from left to right.

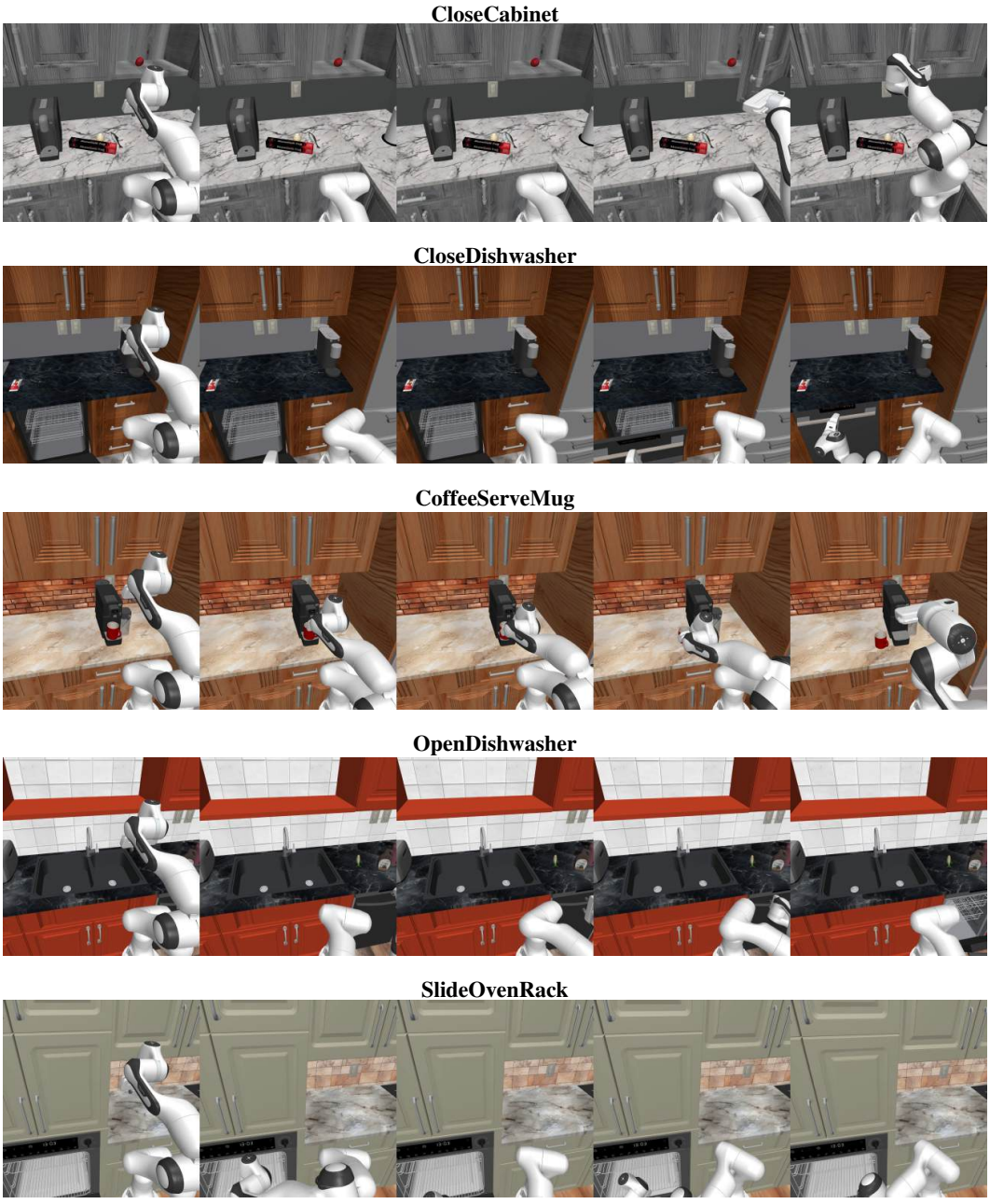


Figure 9: **Example RoboCasa365 executions, group 3 of 3.** Unseen tasks; keyframes run from left to right.

C CROSS-EMBODIMENT REAL-WORLD DETAILS

Section 5.2 reports cross-embodiment results on two real robotic arms—xArm7 and YAM. Both arms were **unseen during pretraining**; they differ in hardware, visual domain, task suite, and the VLA baselines they are compared against. We detail the per-arm setup, tasks, and training protocol below. The corresponding combined success-rate table is Table 3 and qualitative rollouts on both arms are visualized in Fig. 5. The supplementary videos cover all three tasks on each arm.

C.1 XARM7 SETUP

The xArm7 evaluation uses a single ZED 2i stereo camera mounted to provide a frontal view of the workspace. We fine-tune all methods on 50 expert trajectories per task collected via human teleoperation and evaluate over 100 independent rollouts per task with *randomized initial object poses*. Baselines are $\pi_{0.5}$ (Black et al., 2025) and GR00T N1.7 (Bjorck et al., 2025). For PointAction, the RGB-XYZ generator is post-trained for 500 steps and an action decoder is trained for the arm. The three evaluation tasks are visualized in Fig. 10: pick-and-place, stack cubes, and stack cups.

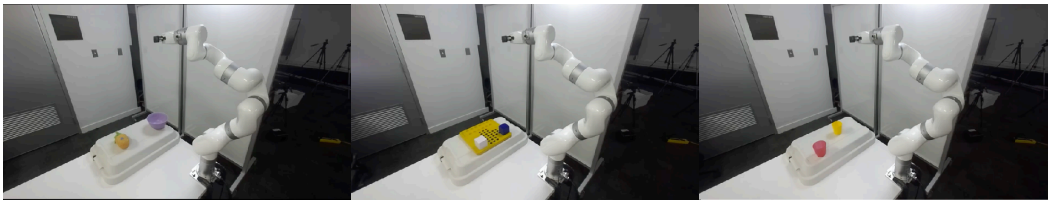


Figure 10: **xArm7 cross-embodiment setup.** The three evaluation tasks: pick-and-place, stack cubes, and stack cups. Test rollouts randomize initial object poses to assess generalization.

C.2 YAM ARM SETUP

The YAM evaluation uses an Intel RealSense D455 stereo camera, with depth computed by FoundationStereo (Wen et al., 2025) as for DROID. Three manipulation tasks are evaluated:

- **Stack Cubes:** the robot stacks a toy cube on top of two already-stacked cubes.
- **Pick Pens:** the robot picks a marker pen out of a container bowl.
- **Insert Cups:** the robot precisely inserts a paper cup into a round plastic tumbler.

We collect 20 expert trajectories per task via teleoperation. The joint RGB-XYZ model is post-trained for 500 additional steps on this dataset, and the point-action decoder is trained from scratch ($\sim$1 hour for 3,000 epochs on the combined three-task dataset). Baselines are π_0 (Black et al., 2024a) and GR00T N1.5 (Bjorck et al., 2025). Each method is evaluated 20 times per task. The three tasks are illustrated in Fig. 11.



Figure 11: **YAM real-world tasks and setup.**

D Additional 4D Generation Analyses

This appendix complements Tables 5 and 6 in the main paper with (i) qualitative rollouts illustrating the temporal and geometric consistency of PointAction's joint RGB-XYZ prediction and (ii) qualitative comparisons against 4D generation baselines.

D.1 Temporal and Geometric Consistency

Fig. 12 visualizes PointAction's predicted rollouts on two real-world scenes. Within each block, the four columns are four uniformly sampled timesteps of the predicted rollout (left to right), and the three rows show the jointly generated (a) RGB frames, (b) XYZ pointmaps, and (c) the XYZ rendered as a 3D point cloud. The figure highlights that the joint RGB-XYZ prediction remains coherent across time and across modalities—a prerequisite for the action decoder to extract reliable point dynamics.

D.2 Comparison with 4D Generation Baselines

We additionally compare against 4D generation baselines on the same two real-world domains (Fig. 13). In these examples, PointAction produces sharper RGB frames and cleaner depth maps. In contrast, 4DNeX (Chen et al., 2025b) tends to predict overly static scenes with limited object motion, while TesserAct's (Zhen et al., 2025) depth maps exhibit noticeable noise and artifacts on real-world scenes.

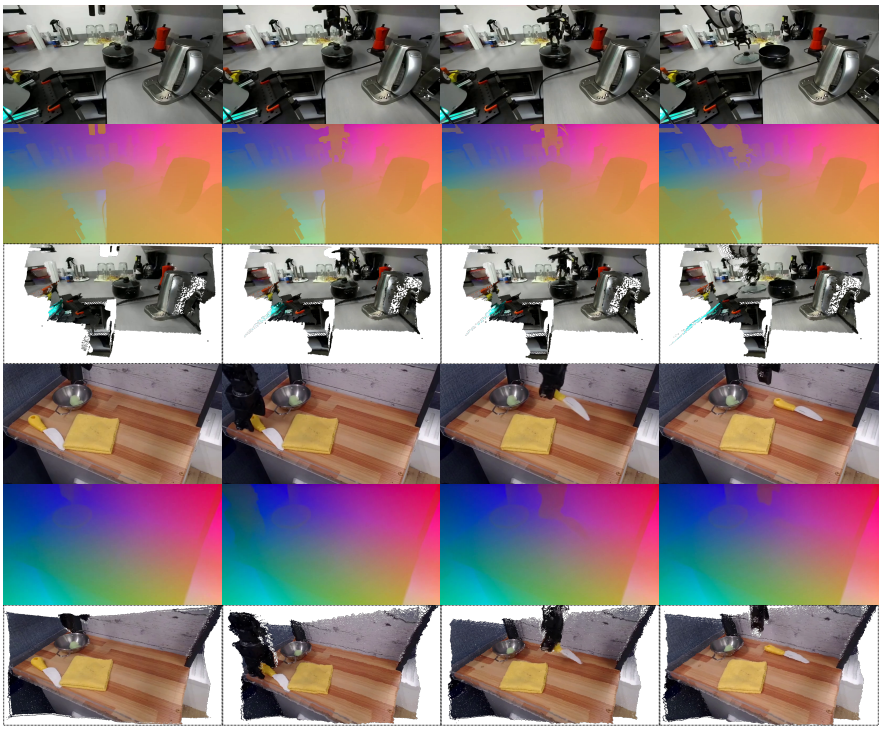


Figure 12: **Qualitative 4D generation by PointAction on real-world rollouts.** Top: a held-out trajectory from DROID (Khazatsky et al., 2024); bottom: a held-out trajectory from BridgeData V2 (Walke et al., 2023). Columns: four uniformly sampled timesteps. Rows: (a) generated RGB frames, (b) generated XYZ pointmaps, (c) rendered XYZ as a 3D point cloud.

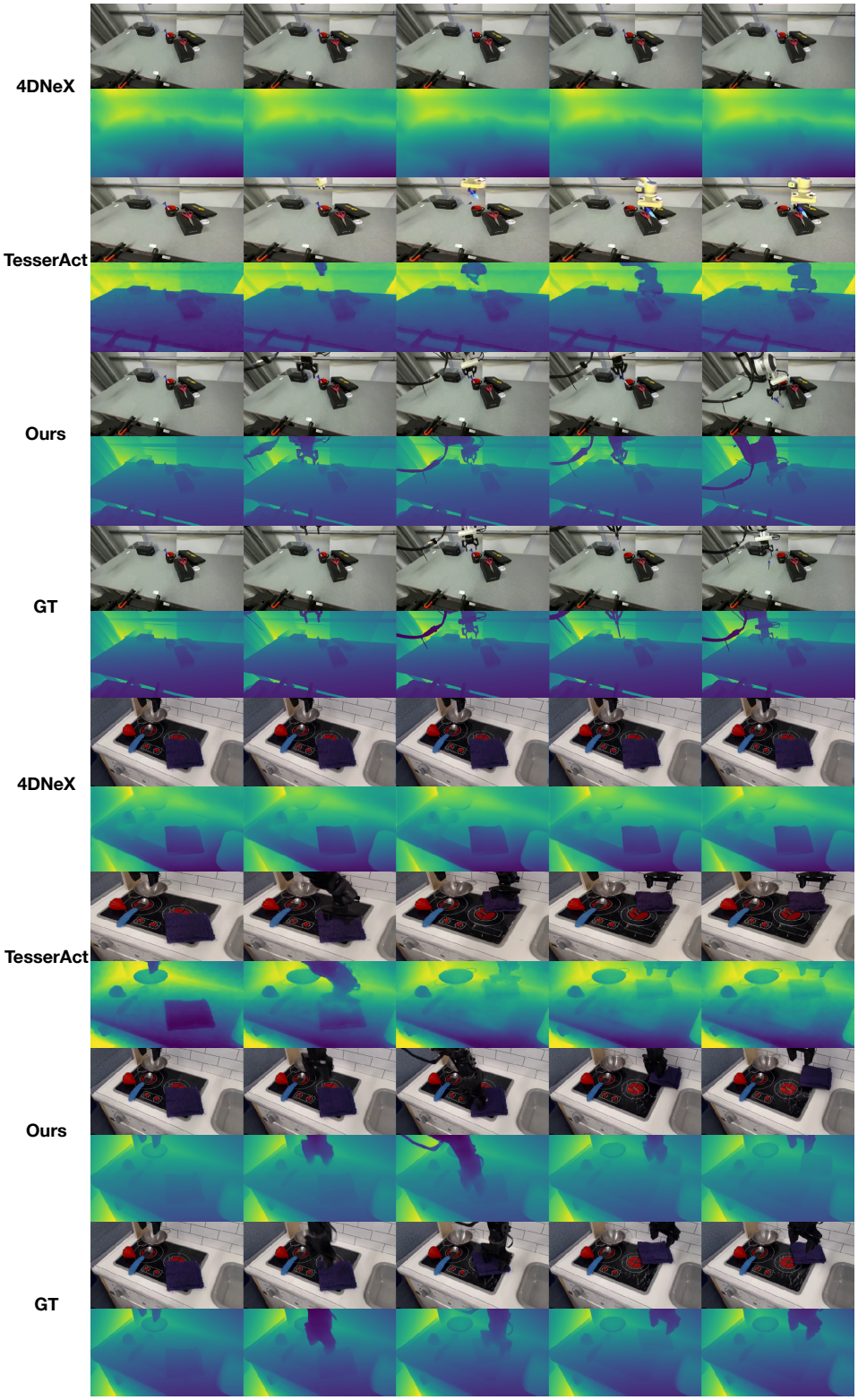


Figure 13: **Qualitative comparisons on RGB and depth generation.** Top: DROID (Khazatsky et al., 2024). Bottom: BridgeData V2 (Walke et al., 2023). Within each dataset block, rows compare 4DNeX, TesserAct, PointAction, and the reference; each method shows RGB and depth over sampled timesteps. Reference depth is estimated.

E FAILURE MODE ANALYSIS

The following recurring failure modes limit deployment:

(1) Self-occlusion. When the robot occludes parts of itself in the generated video, the robot-centric XYZ pointmap is incomplete, and the action decoder receives degraded geometric input. This is most pronounced when the gripper is partly hidden behind the arm from the camera's viewpoint.

(2) Open-loop execution. Because the policy executes a 49-action chunk decoded from one 49-frame generation without re-planning, it cannot recover from unexpected events such as objects slipping during grasping or perturbations from environmental contact. Closing this loop—e.g., by distilling the video backbone into an autoregressive, causally cached architecture (Huang et al., 2025; Yin et al., 2025; Zhu et al., 2026)—is a clear direction for future work.

(3) Camera dependence and uncertain geometry. The points are in the camera coordinate frame, and no viewpoint-invariance experiment is reported. Real-world reference geometry is estimated from stereo or monocular depth, so the learning signal can contain depth artifacts. Robot-only masking also omits explicit object and contact state from the action decoder.

(4) Slow planning and temporal compression. The approximately 381-second serial generation–segmentation–decoding cost per chunk is much longer than the nominal generated clip. Uniformly resampling longer demonstrations into 49 steps can discard fine temporal detail.