# XGENACT: GEOMETRY-ENHANCED WORLD ACTION MODELS THROUGH CROSS-TASK GENERATION

**Anonymous authors**
Paper under double-blind review

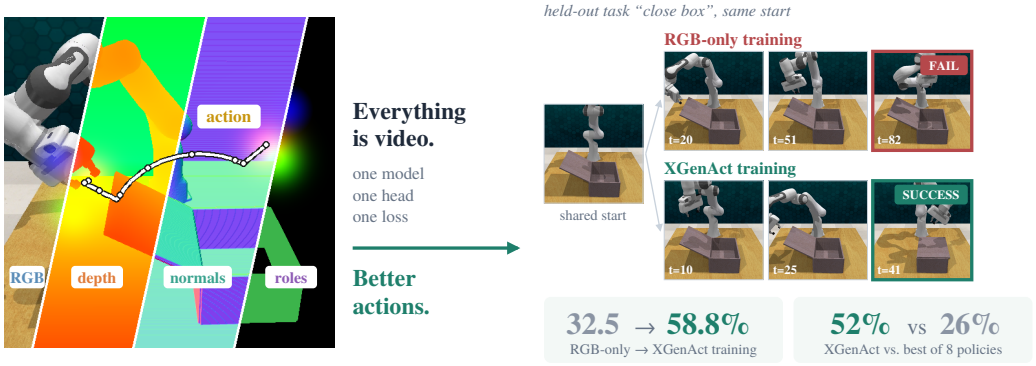



Figure 1: **One checkpoint, multiple future spaces.** Given initial observation and action anchors and the instruction "slide the bottom drawer open," XGENACT generates future RGB, action images, metric depth, functional-role segmentation, and surface normals at matched temporal indices. The $t$=0 column is given; the rest are generated. The visual rows are obtained through separate prompts to the same model rather than one five-stream decode; one camera view is shown.

## ABSTRACT

World action models (WAMs) have advanced robot control by predicting how observations and actions evolve over time. Despite this progress, RGB and action based future prediction does not explicitly address the spatial understanding needed for robot manipulation. Existing efforts often add a limited set of spatial prediction tasks through specialized heads or branches, leaving both the range of spatial supervision and the model architecture fragmented. We introduce XGenAct, a world action model that represents RGB observations, robot actions, metric depth, surface normals, and functional role segmentation as RGB videos through deterministic codecs. By sampling perception and action tasks during training, XGenAct uses one video diffusion transformer and one objective to learn temporal prediction across these spaces without modality specific learned heads. On held out RLBench tasks, structured perception training improves average closed loop success over RGB only training, and XGenAct achieves 52% success in the five task external comparison, versus 26% for the strongest evaluated baselines. It also predicts future depth and segmentation more accurately than the evaluated pipelines that generate RGB first and then apply a frozen perception expert.

## 1 INTRODUCTION

World action models (WAMs) have emerged as a promising paradigm for generative robot control by jointly predicting future RGB videos and robot actions (Ye et al., 2026; Zhu et al., 2025; Zhen et al., 2026). Compared with vision language action models that directly predict actions, WAMs use video generation to explicitly model how a scene evolves over time. This generative objective provides dense supervision for visual dynamics and temporal context, which can in turn improve action prediction. WAMs therefore offer a direct way to address the temporal reasoning required for



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.

robot control. However, their progress has been primarily temporal rather than spatial. Robots operate in a three dimensional world that evolves over time, while most WAMs represent the predicted world only as two dimensional RGB video. Although RGB observations contain implicit geometric cues, appearance prediction alone does not explicitly capture metric depth, surface orientation, or other spatial structures required for physical interaction. This limited spatial understanding remains a major bottleneck to further improving WAMs.

Recent work has begun to address this limitation by introducing spatial information into WAMs. One possible direction is to predict explicit three dimensional or four dimensional representations, such as point clouds or point trajectories. (Zhen et al., 2025; Dharmarajan et al., 2025) While these representations provide rich geometric structure, incorporating dense reconstruction directly into a generative control model requires additional representations and decoding procedures, making efficient joint prediction with robot actions nontrivial. A more practical alternative is to supervise spatial understanding through image aligned signals, including depth, optical flow, geometric features, and semantic maps. Several recent WAMs have adopted this strategy to enrich RGB based future prediction (Ma et al., 2026; Yuan et al., 2026a; Li et al., 2026; Zhao et al., 2026). However, these spatial signals are typically introduced through auxiliary prediction heads, gated residual modules, or modality specific branches, each with its own representation and training objective. Their coverage also remains fragmented. Depth receives the most attention, while few methods incorporate complementary information such as surface orientation and task dependent functional regions within a single model. We argue that spatial supervision should not remain an auxiliary component attached to an RGB based WAM. Instead, multiple geometric and semantic prediction tasks should be expressed within the same generative space used to model observations and actions, allowing structured spatial knowledge to directly shape the shared action generator.

Inspired by ACTIONIMAGES Zhen et al. (2026), which represents 7 DoF robot actions as multi-view RGB videos, we consider image space as a common interface for both perception and control. This representation is particularly suitable for structured visual signals because metric depth, surface normals, and segmentation can also be expressed as decodable RGB images (Gabeur et al., 2026; Wang et al., 2026). Based on this insight, we introduce XGENACT, a geometry enhanced world action model that represents RGB observations, metric depth, surface normals, functional role segmentation, and 7 DoF robot actions as RGB videos through deterministic codecs. Every stream is encoded by the same frozen video VAE into a shared latent space. During training, each sample selects a perception action template that pairs one perception stream with the action stream, rather than combining all perception modalities simultaneously. All templates are processed by the same diffusion transformer, output head, and flow matching objective, without modality specific learned heads, auxiliary losses, or manually balanced loss coefficients. By varying the perception task across training samples, this cross task formulation exposes the shared action generator to complementary geometric and semantic supervision. Experiments on held-out RLBench tasks demonstrate that XGENACT achieves the highest average closed-loop success among eight external policy baselines while retaining the ability to predict future RGB, depth, surface normals, and segmentation.

Our contributions are summarized as follows:

- **Geometry enriched image space.** We extend image space world action modeling beyond RGB and actions by encoding depth, surface normals, and functional role segmentation in the same visual modality, adding complementary geometric and semantic information.
- **Head free and extensible task interface.** Task templates and conditioning plans enable all perception and action tasks to share the same frozen VAE, DiT, output head, and flow matching objective, without modality specific learned heads or auxiliary losses.
- **Improved robot control through cross-task generation.** Experiments on held-out RLBench tasks show that structured geometric and semantic generation improves average closed-loop success by up to 28.75 percentage points over RGB-only training at matched compute and action budgets, while retaining future RGB generation.

## 2 RELATED WORK

**Structured world prediction for action learning.** World action models (WAMs) jointly learn structured visual future prediction and robot action generation, often using a shared video diffu-

sion backbone (Ye et al., 2026; Zhu et al., 2025; Zhen et al., 2026; Zhou et al., 2026; Chen et al., 2026b). While most WAMs predict RGB futures, recent methods introduce geometric or semantic supervision. PAD (Guo et al., 2024) extends joint image and action denoising to depth using separate modality encoders and decoders. GeoSem-WAM (Ma et al., 2026), DreamWAM (Yuan et al., 2026a), WAM4D (Li et al., 2026), and X-WAM (Guo et al., 2026) incorporate geometry through auxiliary branches, residual modules, or training-only readouts. RynnWorld-4D (Zhao et al., 2026) generates RGB, depth, and optical flow using a three branch architecture, then predicts actions with a downstream inverse dynamics head. These methods demonstrate the value of structured future supervision, but rely on modality specific components, objectives, or action modules.

SA-WAM (Lopetegui Gonzalez et al., 2026) is closest to our representation. It encodes RGB and metric depth with the same frozen video tokenizer and jointly denoises them with actions in one latent sequence. However, actions occupy dedicated latent positions, and the model uses a fixed RGB, depth, and action layout. In contrast, XGENACT represents both structured perception and action as video streams and samples multiple perception action templates within one shared generative model.

**Perception as generation.** A complementary line of work formulates visual understanding through image or video generation. Painter (Wang et al., 2023) and DICEPTION (Zhao et al., 2025) express multiple perception tasks in a common visual space. Vision Banana (Gabeur et al., 2026) and GenCeption (Wang et al., 2026) generate depth, segmentation, surface normals, and correspondence from task prompts. SUV (Yuan et al., 2026b) applies a similar formulation to driving, but retains a separate action expert. These methods establish generation as a common interface for perception, but do not examine whether a mixture of structured perception tasks can improve a robot action generator expressed in the same output space.

**Visual representations of action.** RT-Trajectory (Gu et al., 2023) and TraceVLA (Zheng et al., 2025) augment observations with visual motion cues but still predict action tokens. PIVOT (Nasiriany et al., 2024) visualizes candidate actions for a vision language model to select. Action Images (Zhen et al., 2026) instead represents a 7-DoF end effector pose as multiview RGB canvases generated by a video model. XGENACT extends this image space beyond RGB observations and actions by incorporating metric depth, surface normals, and functional role segmentation. This connects perception as generation with action as generation, allowing multiple geometric and semantic tasks to supervise one action generator without modality specific heads or auxiliary losses.

# 3 METHOD

## 3.1 OVERVIEW

We build XGENACT on the Action Images formulation (Zhen et al., 2026) using a Wan2.2-TI2V-5B video diffusion backbone (Wan Team, 2025). Following Action Images, a 7-DoF robot action is projected into multiple camera views and rendered as an RGB video that can be decoded back into continuous control. We retain this action representation without modification. The video VAE remains frozen, and only the diffusion transformer is fine-tuned.

Our key idea is to extend this shared video space with geometry-related perception tasks. RGB observations, metric depth, surface normals, functional role segmentation, and robot actions are converted into RGB videos through deterministic codecs and encoded by the same frozen VAE. This places appearance, geometry, semantics, and control in a common latent space without introducing modality specific learned encoders or output heads.

For each training sample, we select one perception modality $m$ and pair it with the action stream from two camera views:

$$\Pi_m = \left[m^{(1)},\, a^{(1)},\, m^{(2)},\, a^{(2)}\right], \qquad m \in \{\mathrm{RGB}, \mathrm{depth}, \mathrm{normal}, \mathrm{segmentation}\}. \tag{1}$$

Thus, a sample contains one perception modality and action, rather than all perception modalities simultaneously. The template $\Pi_m$ determines which streams the model processes, while a conditioning plan determines which frames are observed and which frames must be generated. All templates share the same diffusion transformer, output head, and training objective.

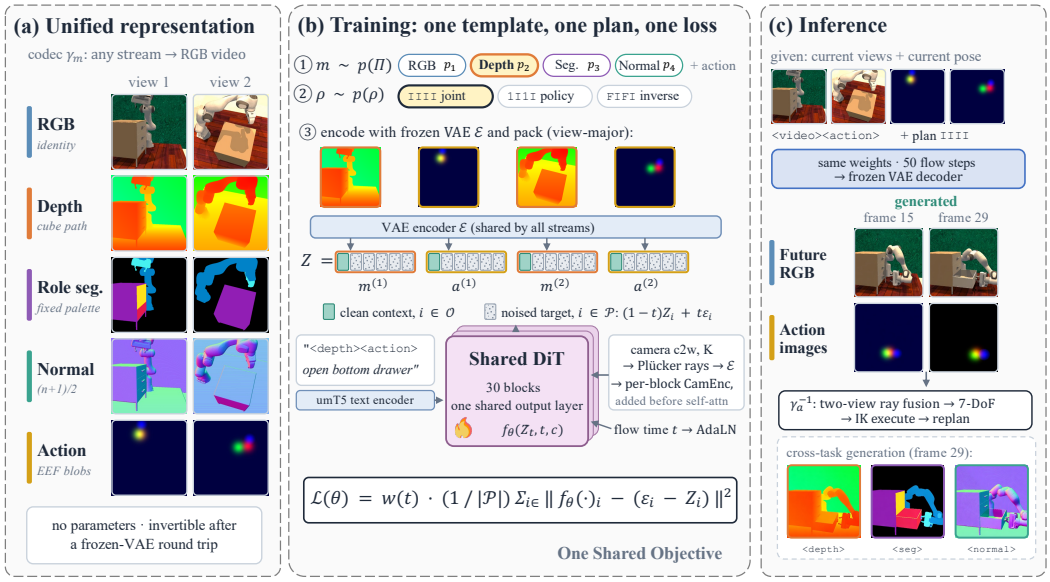



Figure 2: **Training and inference, shared architecture, and conditioning plans.**

### 3.2 Geometry Enriched Video Space

Let $x^{(m,v)}$ denote a video stream of modality $m$ from camera view $v$. We define a deterministic codec $\gamma_m$ that converts each stream into a three channel video accepted by the frozen video VAE:

$$\tilde{x}^{(m,v)} = \gamma_m\left(x^{(m,v)}\right), \qquad z^{(m,v)} = \mathcal{E}\left(\tilde{x}^{(m,v)}\right), \tag{2}$$

where $\mathcal{E}$ is shared across all modalities. These codecs contain no learnable parameters. They preserve the information required to recover the original signal while producing videos that remain stable under VAE encoding and decoding.

**RGB and action.** RGB observations use the identity mapping. Robot actions use the Action Images codec (Zhen et al., 2026), which projects each 7-DoF end effector pose into the camera views and renders the resulting trajectories as RGB videos. We use this representation unchanged.

**Metric depth.** Raw metric depth has an unbounded and highly nonuniform value distribution, making direct grayscale encoding inefficient. We first compress depth values with a monotonic power transform and then map the resulting scalar values onto a continuous path through the RGB cube. At decoding time, generated colors are projected back onto this path before the scalar transform is inverted. This encoding distributes metric depth across an RGB path instead of a single grayscale axis. Full definitions and decoding rules are provided in Appendix C.

**Surface normals.** A surface normal $\mathbf{n} \in [-1, 1]^3$ is mapped directly to RGB as

$$\gamma_{\text{normal}}(\mathbf{n}) = \frac{\mathbf{n} + \mathbf{1}}{2}. \tag{3}$$

The inverse mapping recovers the components and renormalizes them to unit length.

**Functional role segmentation.** We assign each functional role a fixed RGB color code shared across tasks, episodes, camera views, and frames. The labels describe how a region participates in the task, such as target object, goal region, robot, tool, fixture, distractor, or background, rather than its semantic object category. This provides task relevant structural supervision while remaining compatible with the same video representation.

Before training, each codec is evaluated both directly and after a frozen VAE round trip. A new perception task can therefore be added through a deterministic codec, without introducing a learned modality specific encoder, decoder, or output head.

### 3.3 Task Templates and Conditioning Plans

Each training sample selects one perception modality $m$ and forms the template introduced in Eq. 1. The four segments correspond to perception and action streams from two camera views. Their order remains fixed across modalities. Sampling different templates changes the perception task associated with action generation, while the model architecture remains unchanged.

A conditioning plan determines which parts of the packed sequence are observed. For each segment, we use one of three states: $F$ provides the full segment, $I$ provides its initial frame and leaves the rest to be generated, and 1 includes only the initial frame. This yields three settings:

$$\begin{aligned} \rho_{\text{joint}} &= (I, I, I, I), \\ \rho_{\text{policy}} &= (1, I, 1, I), \\ \rho_{\text{inverse}} &= (F, I, F, I). \end{aligned} \tag{4}$$

The joint plan predicts future perception and actions together. The policy plan predicts future actions from the current perception. The inverse dynamics plan predicts actions from a fully observed perception sequence. The same plans apply to every perception modality.

During training, both the template and conditioning plan are sampled for each example. All templates use the same plan distribution. We also occasionally remove the action segments, allowing the model to learn future perception generation from an initial frame alone. Sampling probabilities are provided in the experimental setup.

### 3.4 Shared Training Objective

Let $Z_\Pi$ be the packed latent sequence produced by template $\Pi$. A conditioning plan divides its latent positions into an observed set $\mathcal{O}$ and a prediction set $\mathcal{P}$. Observed positions remain clean, while prediction positions are interpolated with Gaussian noise:

$$\tilde{Z}_{t,i} = \begin{cases} Z_i, & i \in \mathcal{O}, \\ (1-t)Z_i + t\epsilon_i, & i \in \mathcal{P}, \end{cases} \qquad \epsilon \sim \mathcal{N}(0, I). \tag{5}$$

The diffusion transformer predicts the flow from the clean latent to noise. We apply the flow matching loss only to positions that must be generated:

$$\mathcal{L}(\theta) = \mathbb{E}_{\Pi,\rho,t,\epsilon}\left[\frac{w(t)}{|\mathcal{P}|}\sum_{i\in\mathcal{P}}\left\| f_\theta(\tilde{Z}_t, t, \rho, \mathbf{c})_i - (\epsilon_i - Z_i)\right\|_2^2\right], \tag{6}$$

where $w(t)$ is the flow-scheduler weighting and $\mathbf{c}$ contains the language instruction and camera conditioning. Observed positions provide context but do not contribute to the reconstruction loss.

This objective is identical for every perception modality, template, and conditioning plan. Cross task training changes only the content of the shared latent sequence and which positions are predicted. It introduces no modality specific output heads, auxiliary objectives, or manually chosen loss weights.

## 4 Experiments

We evaluate XGenAct on three questions: (i) how it compares with existing robot policies in closed loop evaluation (§4.2); (ii) whether generating structured perception during training improves the action policy, and how perception spaces affect the action performance(§4.3); and (iii) whether generating future geometry and semantics is more accurate than generating RGB and applying a frozen geometry perception model (§4.4).

| Method | Close microwave | Toilet seat down | Reach target | Close drawer | Close laptop | Avg. |
|---|---|---|---|---|---|---|
| PAD-Depth (Guo et al., 2024) | 5 | 5 | 10 | 40 | 0 | 12 |
| $\pi_{0.5}$ (Physical Intelligence et al., 2025) | 30 | 5 | **35** | 35 | 25 | 26 |
| MolmoAct (Lee et al., 2025) | 0 | 40 | 10 | 0 | 5 | 11 |
| UWM (Zhu et al., 2025) | 0 | 10 | 5 | 50 | 5 | 14 |
| LaWAM (Chen et al., 2026a) | 15 | **70** | 10 | 25 | 5 | 25 |
| Evo-1 (Lin et al., 2026) | 10 | 10 | 0 | 10 | 5 | 7 |
| VLA-JEPA (Sun et al., 2026) | 0 | 20 | 5 | 30 | 15 | 14 |
| VLANeXt (Wu et al., 2026) | 10 | 50 | 0 | 50 | 20 | 26 |
| **XGenAct (ours)** | **60** | 40 | 20 | **100** | **40** | **52** |

Table 1: **Closed-loop success rate (%) on five unseen RLBench tasks**, 20 trials each. All methods except MolmoAct are fine-tuned on our 16-task RLBench split; MolmoAct is evaluated zero-shot with its output adapted to our controller, following Zhen et al. (2026). Training and evaluation details are in Appendix H. Best result per column in bold.

### 4.1 SETUP

**Training menus.** We name a training configuration by the perception templates in its menu: RGB trains on `video+action` only, and D, N and S add `depth+action`, `normal+action` and `segmentation+action`, respectively. The most inclusive menu, RGB+D+N+S, draws `video+action` with probability $0.4$ and each perception template with probability $0.2$. Every model is fine-tuned from the released Action Images checkpoint (Zhen et al., 2026) with the same backbone, frozen VAE, objective, conditioning-plan distribution and optimizer (effective batch 4, learning rate $5 \times 10^{-7}$; Appendix E), and is evaluated at step 4,000.

**Training groups.** We train three groups of models, one per question. Two training sources are used: a self-generated RLBench tree (James et al., 2020) of 16 tasks and $3{,}960$ episodes at $512^2$, and a ManiSkill3 tree (Tao et al., 2025) of 7 tasks and $1{,}750$ episodes (Appendix B). (i) For the comparison with robot policies, XGENACT (RGB+D+N+S) and all baselines except the zero-shot MolmoAct are fine-tuned on the same RLBench tree. (ii) For the menu ablation, eight models, one per menu, are trained on RLBench and ManiSkill3 sampled at a $70/30$ ratio. (iii) For the perception study, a separate set of menu models is trained on the same $70/30$ mixture; an RLBench-only RGB+D+N+S model is included as unmatched context. All groups are evaluated on RLBench tasks absent from training; ManiSkill3 serves only as a training source.

**Evaluation.** Closed-loop control runs 20 trials per task with receding-horizon replanning under the `IIII` plan, capped at $1.5\times$ the demonstration length; success is RLBench's task-success signal, and methods run in our rollout harness are paired by scene seed. Future perception is scored against the simulator state produced by each route's own decoded actions: LPIPS for RGB (Zhang et al., 2018), absolute relative error for depth, mean cosine similarity for normals, and mIoU for segmentation.

### 4.2 COMPARISON WITH ROBOT POLICIES

We compare against eight policies fine-tuned on the same 16-task RLBench split (MolmoAct is evaluated zero-shot, with its released checkpoint adapted to our controller as in Action Images; Zhen et al., 2026): vision-language-action policies ($\pi_{0.5}$, Evo-1, VLANeXt), a VLA that reasons over depth tokens (MolmoAct), and world action models that predict the future in pixel space (PAD-Depth, UWM) or in a latent space (LaWAM, VLA-JEPA). All baselines except $\pi_{0.5}$ and LaWAM receive per-view camera parameters, since training cameras are randomized per episode.

As shown in Table 1, XGENACT reaches $52\%$ average success, twice the strongest baselines ($26\%$ for $\pi_{0.5}$ and VLANeXt), and is best on three of the five tasks: close drawer ($100\%$ vs. at most $50\%$), close microwave ($60\%$ vs. $30\%$) and close laptop ($40\%$ vs. $25\%$). The two other world action models that generate future pixels, PAD-Depth and UWM, reach only $12\%$ and $14\%$. Baselines lead on toilet seat down (LaWAM, $70\%$) and reach target ($\pi_{0.5}$, $35\%$), yet XGENACT is the only method above $10\%$ on every task.

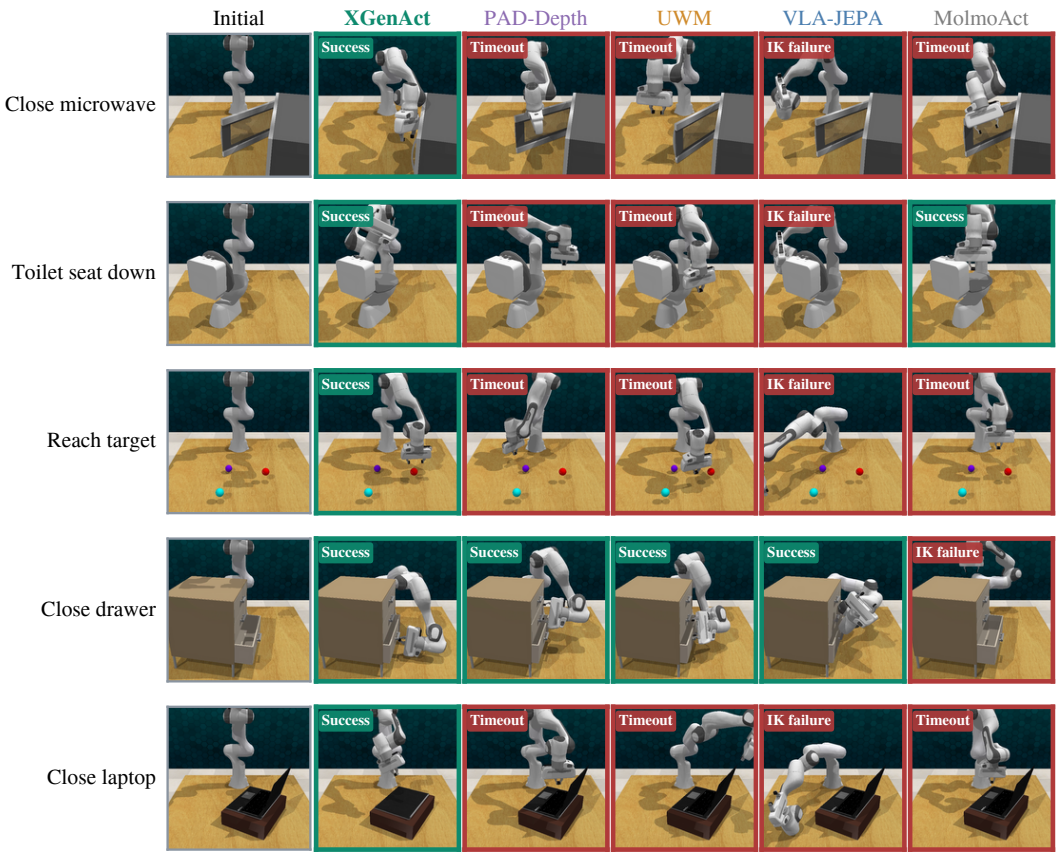



Figure 3: **Paired closed-loop rollouts against external policies.** Final frame of one episode per task and method; all methods start from the same scene (trials are paired by scene seed). The trial shown is the first one XGENACT solves, and the baselines are shown on that trial whatever their outcome. Success rates over 20 trials are in Table 1; per-task key frames and failure cases are in Appendix J.

| Model | RLBench | | | | |
|---|---|---|---|---|---|
| | Close microwave | Toilet seat down | Close drawer | Meat on grill | RL avg. |
| RGB | 5 | 20 | **95** | 10 | 32.50 |
| RGB+D | 45 | 20 | <u>85</u> | 60 | 52.50 |
| RGB+S | **95** | 5 | 60 | **70** | 57.50 |
| RGB+N | 75 | **40** | 70 | 45 | 57.50 |
| RGB+D+S | 70 | 25 | 60 | 60 | 53.75 |
| RGB+N+S | 80 | <u>35</u> | 65 | <u>65</u> | **61.25** |
| RGB+D+N | 75 | <u>35</u> | 60 | 55 | 56.25 |
| RGB+D+N+S | <u>85</u> | <u>35</u> | 55 | 60 | <u>58.75</u> |

Table 2: **Closed-loop success (%) across training-time modality menus.** Bold and underlined values are the best and second-best results per column.

The failure modes differ as well (Figure 3). Over 100 trials per method, PAD-Depth and UWM mostly run out of time (88 and 86 timeouts), while VLA-JEPA and MolmoAct often command poses the controller cannot reach and stop on inverse-kinematics (IK) failures (48 and 22 trials). XGENACT never exhausts its stricter budget of 20 IK failures in total; every failure is a timeout.

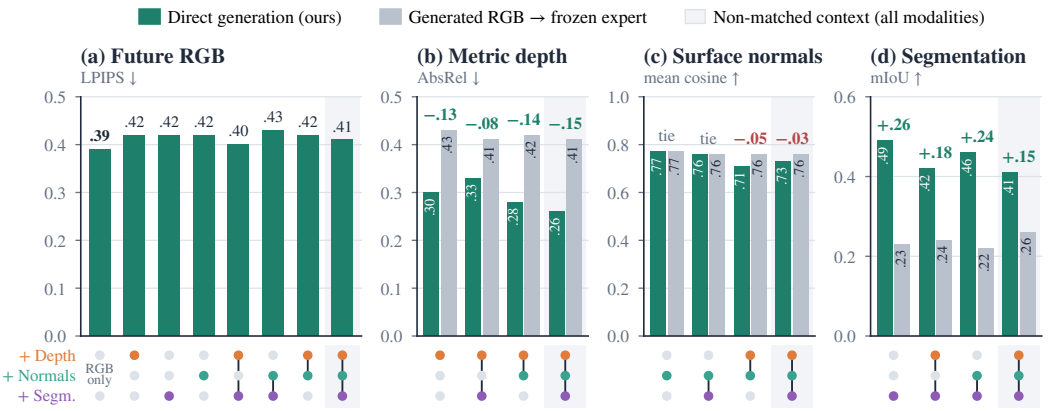



Figure 4: **Direct generation versus generated RGB followed by a frozen expert.** Each column is one training menu; the dots below mark the perception streams it adds to RGB. Teal bars: the model generates the perception space directly. Gray bars: the same model generates future RGB, which Depth Anything V2, Lotus-G or CLIPSeg converts. Numbers above a pair give direct minus expert (green: direct is better). Scores average four held-out RLBench tasks, measured against the simulator state reached by each route's own actions; a task is left out of a panel if its executed poses deviate from the commanded ones by 10 mm or more (Appendix G; per-task values in Table 7).

### 4.3 Structured Perception Improves the Robot Control

Table 2 compares the eight models of group (ii), which differ only in their training menu. Every menu with a perception stream improves on RGB, raising average success from 32.5% to 52.5–61.25%; the best, RGB+N+S, gains 28.75 points. The most useful stream depends on the task. Segmentation helps most where the manipulated part must be separated from its surroundings: RGB+S is best on close microwave (95%) and meat on grill (70%). Normals help rotating a thin planar part: RGB+N is best on toilet seat down (40%), and the three runners-up all include normals. Close drawer is the exception, where RGB already reaches 95%. The results show that more perception tasks is not monotonically better. Adding depth to RGB+N+S lowers the average from 61.25% to 58.75%, although depth alone is clearly useful (52.5% for RGB+D). Under a fixed model capacity and training budget, the streams therefore overlap or compete, and the best menu is the one whose geometry or semantics resolve a task's main ambiguity. These trends are correlational; paired executions of RGB and XGenAct on the same scenes are shown in Figure 10.

### 4.4 Future Perception under Simulator Replay

Every XGenAct model can produce a future perception map in two ways: generate it directly with the corresponding template (e.g. `depth+action`), or generate future RGB with `video+action` and pass it to a frozen expert (Depth Anything V2 for depth, Lotus-G for normals, CLIPSeg for segmentation). We compare the two routes on four held-out tasks using the perception-study models (group iii). Since the robot executes each route's own decoded actions, both are scored against the simulator state those actions reach, not against a demonstration the rollout may have left (Figure 7).

Figure 4 shows that generating geometry and semantics directly is clearly better than reading them off generated RGB by external experts: depth AbsRel drops from 0.41–0.43 to 0.28–0.33, and segmentation mIoU roughly doubles, from 0.22–0.24 to 0.42–0.49. For surface normals the two routes are on par (0.76–0.77 cosine), except with RGB+D+N, where the direct route trails (0.71 vs. 0.76). Adding perception streams costs little in future RGB: LPIPS rises only from 0.39 for RGB to 0.40–0.43. The RLBench-only RGB+D+N+S model, shown as unmatched context, follows the same pattern (0.26 AbsRel, 0.41 mIoU, 0.73 cosine).

Because the depth stream is metric, a generated frame can be used directly as geometry. Figure 5 back-projects the last generated depth frame on three held-out tasks: the table, fixtures and object layout coincide with the ground-truth cloud, and the visible differences lie on the arm, whose predicted pose runs ahead of or behind the demonstration.

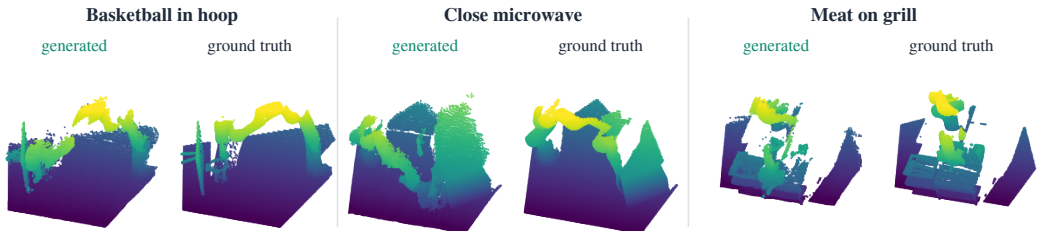



Figure 5: **Generated metric depth lifted to 3-D on held-out tasks.** The last generated depth frame (view 1, open loop) and the demonstration's depth at the same frame are back-projected with the camera intrinsics and rendered from the same novel viewpoint and axis limits; colour encodes height. Pixels on depth discontinuities are dropped in both clouds. Further qualitative results are in Appendix J.

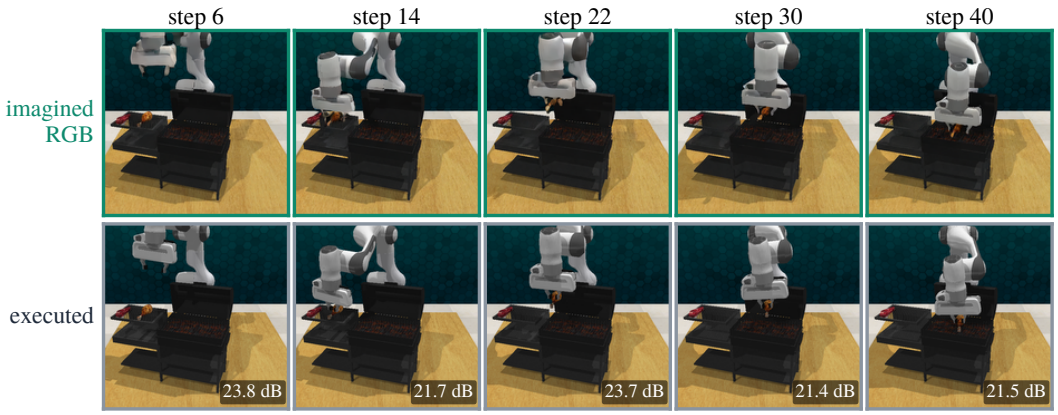



Figure 6: **Imagined versus executed future on a held-out task** (meat on grill, view 1, first replanning step). Top: RGB frames XGENACT generated for step $k$. Bottom: the simulator after executing the decoded actions up to $k$ (corner: PSNR to the imagined frame). More in Appendix J.3.

The imagined future also matches what the robot then does. Figure 6 pairs the RGB frames generated in one replanning step with the simulator frames reached by executing the decoded actions. Across the 40-step chunk the two rows show the same grasp, transport and placement of the chicken in nearly the same arm poses (whole-frame PSNR 21–24 dB): the plan is carried out as imagined, not replaced by a different motion at execution time.

## 5 CONCLUSION

We presented XGENACT, a world action model that expresses robot actions, future RGB video, metric depth, surface normals, and functional role segmentation through a shared video generation interface. Deterministic codecs allow these tasks to use the same frozen video VAE, diffusion transformer, output head, and training objective. By sampling a perception and action template for each training example, the model incorporates structured visual supervision without adding modality-specific learned heads or auxiliary losses. On held-out RLBench tasks, structured-perception training improves average closed-loop success over RGB-only training in our matched modality ablation. The resulting model also directly predicts future depth and segmentation more accurately than the evaluated pipelines that first generate RGB and then apply a frozen perception expert. The best training menu varies across control tasks and perception outputs, indicating that adding more modalities does not necessarily improve every metric under a fixed training budget. Future work will examine larger and more diverse evaluation suites, cross-modal consistency, and how to choose or balance perception tasks during training.

## AI USE STATEMENT

Generative AI tools were used for literature retrieval and discovery, research ideation, and experimental support. During the early brainstorming stage, we started from a broad idea of transferring Vision Banana to robotic manipulation. Generative AI tools were used to search for and summarize related literature on World Action Models, which helped us identify relevant prior work, including ActionImages. During research execution, generative AI tools were used to assist with understanding, implementing, and debugging baseline methods. They were also used as a discussion aid for conditioning strategies, ablation studies, and experimental design, and to help draft preliminary experiment plans. During Paper writing, AI is used to draft related Work, experiments setup, table templates, Appendix and polish method and introduction.

All AI-assisted research activities and generated content were reviewed and verified by the authors. The authors take responsibility for the final methodology, experiments, results, and manuscript.

## REPRODUCIBILITY STATEMENT

We provide the information needed to independently reproduce our method and experiments throughout the paper and appendix. Section 3 and Appendix A specify the model, objective, and action representation; Appendices B–F document data construction and disjoint splits, deterministic modality codecs and admission checks, the complete training mixture, optimization hyperparameters, and camera conventions; and Appendices G and H define the evaluation and external-baseline protocols. Upon publication, we will publicly release the complete code for data generation and preprocessing, training, inference, closed-loop rollout, and evaluation, together with pinned environment specifications, launch commands, configuration files, task and split definitions, and the random seeds used in the reported experiments. We will also release the trained checkpoints for every reported XGENACT variant. For simulator assets and third-party checkpoints governed by their own licenses, we will provide versioned download and reconstruction instructions rather than redistribute them. The release will map every table and figure to its configuration and checkpoint, with scripts that reproduce the reported metrics from raw simulator outputs.

## REFERENCES


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.

Jonathan T. Barron. A power transform. *arXiv preprint arXiv:2502.10647*, 2025.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, et al. Lawam: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026a.

Yixiang Chen, Peiyan Li, Yuan Xu, Qisen Ma, Jiabing Yang, Kai Wang, Jianhua Yang, Dong An, He Guan, Gaoteng Liu, et al. Flowwam: Optical flow as a unified action representation for world action models. *arXiv preprint arXiv:2607.13017*, 2026b.

Karthik Dharmarajan, Wenlong Huang, Jiajun Wu, Li Fei-Fei, and Ruohan Zhang. Dream2flow: Bridging video generation and open-world manipulation with 3d object flow. *arXiv preprint arXiv:2512.24766*, 2025.

Valentin Gabeur, Shangbang Long, Songyou Peng, Paul Voigtlaender, Shuyang Sun, Yanan Bao, Karen Truong, Zhicheng Wang, Wenlei Zhou, Jonathan T Barron, et al. Image generators are generalist vision learners. *arXiv preprint arXiv:2604.20329*, 2026.

Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, et al. RT-Trajectory: Robotic task generalization via hindsight trajectory sketches. *arXiv preprint arXiv:2311.01977*, 2023.

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.

Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. *Advances in Neural Information Processing Systems*, 2024.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

Ying Li, Xiaobao Wei, Jiajun Cao, Hao Wang, Xiaowei Chi, Chengyu Bai, Qianpu Sun, Jiajun Li, Xiaojie Zhang, Peidong Jia, Jian Tang, Sirui Han, and Shanghang Zhang. WAM4D: Fast 4d world action model via spatial register tokens. *arXiv preprint arXiv:2606.14048*, 2026.

Tao Lin, Yilei Zhong, Yuxin Du, Jingjing Zhang, Jiting Liu, Yinxinyu Chen, Encheng Gu, Ziyan Liu, Hongyi Cai, Yanwen Zou, et al. Evo-1: Lightweight vision-language-action model with preserved semantic alignment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13397–13406, 2026.

Javier Alejandro Lopetegui Gonzalez, Paul Pacaud, and Cordelia Schmid. Spatially aware world action model via geometric latent diffusion. *arXiv preprint arXiv:2609.02531*, 2026.

Fulong Ma, Daojie Peng, Wenjun Yue, Jiahang Cao, Bintao Wang, Qiang Zhang, and Jun Ma. Geosem-wam: Geometry-and semantic-aware world action models. *arXiv preprint arXiv:2606.03188*, 2026.

Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, Quan Vuong, Tingnan Zhang, et al. PIVOT: Iterative visual prompting elicits actionable knowledge for vision-language models. *arXiv preprint arXiv:2402.07872*, 2024.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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. *arXiv preprint arXiv:2504.16054*, 2025.

Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.

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.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Letian Wang, Chuhan Zhang, Rishabh Kabra, Jasper Uijlings, Steven Waslander, Andrew Zisserman, Joao Carreira, Kaiming He, Misha Andriluka, Eduard Gabriel Bazavan, et al. Video generation models are general-purpose vision learners. *arXiv preprint arXiv:2607.09024*, 2026.

Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist painter for in-context visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Xiao-Ming Wu, Bin Fan, Kang Liao, Jian-Jian Jiang, Runze Yang, Yihang Luo, Zhonghua Wu, Wei-Shi Zheng, and Chen Change Loy. Vlanext: Recipes for building strong vla models. In *Proceedings of the International Conference on Machine Learning*, 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.

Shanglin Yuan, Weiheng Zhao, Xin Shi, Haoyi Jiang, Xianda Guo, Liu Liu, Wenyu Liu, Wei Sui, and Xinggang Wang. Dreamwam: Beyond rgb future prediction for world action models. *arXiv preprint arXiv:2608.04996*, 2026a.

Yibo Yuan, Jiacheng Fu, Jiangtong Zhu, Yi Li, Jianhua Han, Meng Tian, Zhuohan Liu, Zhiwei Xiong, Hang Xu, Jianwu Fang, et al. Suv: Future scene understanding as video generation for end-to-end driving. *arXiv preprint arXiv:2608.03084*, 2026b.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.

Canyu Zhao, Yanlong Sun, Mingyu Liu, Huanyi Zheng, Muzhi Zhu, Zhiyue Zhao, Hao Chen, Tong He, and Chunhua Shen. DICEPTION: A generalist diffusion model for visual perceptual tasks. *arXiv preprint arXiv:2502.17157*, 2025.

Haoyu Zhao, Xingyue Zhao, Siteng Huang, Xin Li, Deli Zhao, and Zhongyu Li. Rynnworld-4d: 4d embodied world models for robotic manipulation. *arXiv preprint arXiv:2607.06559*, 2026.

Haoyu Zhen, Qiao Sun, Hongxin Zhang, Junyan Li, Siyuan Zhou, Yilun Du, and Chuang Gan. Tesseract: learning 4d embodied world models. *arXiv preprint arXiv:2504.20995*, 2025.

Haoyu Zhen, Zixian Gao, Qiao Sun, Yilin Zhao, Yuncong Yang, Yilun Du, Pengsheng Guo, Tsun-Hsuan Wang, Yi-Ling Qiao, and Chuang Gan. Action images: End-to-end policy learning via multiview video generation. *arXiv preprint arXiv:2604.06168*, 2026.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *International Conference on Learning Representations*, 2025.

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, Chenhao Qiu, et al. $\tau_0$-wm: A unified video-action world model for robotic manipulation. *arXiv preprint arXiv:2606.01027*, 2026.

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.

# A Architecture and action representation

## A.1 Latent video-diffusion world action model

We build on Wan2.2-TI2V-5B (Wan Team, 2025), a text-conditioned latent video diffusion transformer whose causal 3D VAE encoder and decoder, $\mathcal{E}$ and $\mathcal{D}$, stay frozen throughout. A clip of $T$ frames at resolution $H \times W$ is encoded as

$$\mathcal{E} : \mathbb{R}^{3\times T\times H\times W} \longrightarrow \mathbb{R}^{C_l\times T_l\times H_l\times W_l}, \qquad T_l = 1 + \frac{T-1}{4}, \quad H_l = \frac{H}{16}, \quad W_l = \frac{W}{16}. \tag{7}$$

For our 41-frame, $512^2$ clips this is an $11 \times 32 \times 32$ grid with $C_l = 48$ channels. Only the diffusion transformer $f_{\boldsymbol{\theta}}$ is fine-tuned. Each sample carries $V = 2$ camera views, conditioned as in Zhen et al. (2026) through Plücker ray embeddings (Sitzmann et al., 2021; Bai et al., 2025), collected in a tensor $\mathbf{C}$ (Appendix F).

## A.2 Actions as video

Following Action Images (Zhen et al., 2026), actions live in the same RGB-video space as observations. For an end-effector state $(\boldsymbol{p}_n, \boldsymbol{R}_n, g_n)$ at frame $n$ — position, orientation, gripper openness — view $v$ projects three image points,

$$\boldsymbol{q}_n^{(1)} = \pi_v(\boldsymbol{p}_n), \qquad \boldsymbol{q}_n^{(c)} = \pi_v(\boldsymbol{p}_n + \ell\, \boldsymbol{R}_n \boldsymbol{e}_c)\,, \quad c \in \{2, 3\}, \tag{8}$$

with fixed tool length $\ell = 0.1$ m and tool-frame axes $\boldsymbol{e}_2 = (1, 0, 0)^\top$ and $\boldsymbol{e}_3 = (0, 0, -1)^\top$. They are rendered into three RGB channels as Gaussian blobs,

$$\boldsymbol{A}^{(v)}_{c,n,\boldsymbol{x}} = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{q}_n^{(c)}\|_2^2}{2\sigma^2}\right), \qquad \sigma = 0.05 \min(H, W). \tag{9}$$

To the frozen VAE and the transformer the action stream is therefore an ordinary RGB video. Gripper encoding and the trajectory decoder follow Zhen et al. (2026) (Appendix C).

# B Training data and evaluation splits

| Source | Training tasks |
|---|---|
| RLBench | `open_drawer`, `slide_block_to_target`, `sweep_to_dustpan`, `meat_off_grill`, `turn_tap`, `put_item_in_drawer`, `close_jar`, `reach_and_drag`, `stack_blocks`, `light_bulb_in`, `put_money_in_safe`, `stack_wine`, `put_groceries_in_cupboard`, `place_shape_in_shape_sorter`, `push_buttons`, `insert_onto_square_peg` |
| ManiSkill3 | `peg_insertion_side`, `pick_cube`, `plug_charger`, `pull_cube_tool`, `push_cube`, `stack_cube`, `stack_pyramid` |

Table 3: Training tasks of the two sources. The external comparison trains on the RLBench tasks only; the menu ablation and the perception study train on both sources at a $70/30$ ratio. Names follow the identifiers used by the data loader.

**RLBench source.** The RLBench source contains 16 tasks, four camera views (front, overhead, left shoulder, and right shoulder), and $512^2$ observations recorded at 20 Hz, with per-frame metric depth, instance masks, and camera intrinsics and extrinsics written alongside every episode. Variation 0 is the only split used for training and holds 3,960 episodes. We targeted 250 seeds per task; `light_bulb_in` and `place_shape_in_shape_sorter` yielded 249 each, and `sweep_to_dustpan` yielded 212. Every episode is regenerated from its seed under Colosseum-style randomization (Pumacay et al., 2024) of camera pose, table color, table texture, background texture, and light color. Object color, object texture, distractors, and object physics are deliberately *not* randomized: color is part of the instruction semantics ("close the *red* jar"). The randomization seed is derived from `crc32(task|variation|seed)` rather than from Python's hash, so a tree is reproducible pixel-for-pixel across processes.

**ManiSkill3 source.** The ManiSkill3 source contains seven tasks and $1{,}750$ training episodes. We produce it by replaying the official ManiSkill demonstrations (Tao et al., 2025) frame by frame through `set_state_dict` and re-rendering them, rather than by collecting new trajectories. Each task contributes 250 episodes, and each episode is rendered from four cameras sampled on a sphere around the workspace center, at $512^2$ and 20 Hz, with one rendered frame per control step. Camera parameters are converted from the OpenCV convention to the RLBench convention of Appendix F, and each episode is admitted only if unprojecting and reprojecting its own rendered depth closes to under $0.51$ px, if no mask pixel falls outside the segmentation vocabulary, and if the fraction of pixels outside the depth codec range stays under $3\%$; failures resample the cameras and, if needed, retry the episode. The byte-level layout is identical to the RLBench tree, so both sources are read by the same loader.

**Mixture.** For the menu ablation and the perception study, training draws a source with probability $0.7$ for RLBench and $0.3$ for ManiSkill3, then an episode uniformly within that source. Over the $24{,}000$ samples of a 6k-step run this gives $4.24$ exposures per RLBench episode and $4.11$ per ManiSkill3 episode, and $4.4\%$ versus $4.3\%$ of samples per task — so the two sources are balanced both per episode and per task rather than only in aggregate. Windows are 41 frames; the RLBench source uses stride 3, which covers $6.0$ s at an effective $6.7$ Hz, and the ManiSkill3 source uses stride 1, $2.0$ s at 20 Hz, because its episodes are only 49–103 frames long and a larger stride would degenerate into frozen tail frames.

**Evaluation suites.** No evaluation episode is drawn from either training tree. The perception study in Table 7 uses four RLBench tasks absent from training: `close_microwave`, `toilet_seat_down`, `close_box`, and `meat_on_grill`. Each task is evaluated over five simulator rollouts at training step 4,000. The internal closed-loop ablation in Table 2 uses `close_microwave`, `toilet_seat_down`, `close_drawer`, and `meat_on_grill`, with 20 trials per task. The external-policy comparison in Table 1 evaluates `close_microwave`, `toilet_seat_down`, `reach_target`, `close_drawer`, and `close_laptop_lid`, also with 20 trials per task. The three protocols use the three separately trained groups of Section 4.1 and should not be pooled.

**Fresh closed-loop scenes.** The rollout harness generates a fresh demonstration per trial from a seed derived from `(task, variation, trial)`, disjoint from the seeds used to render either training tree, so a closed-loop trial is a new scene rather than a replay of a stored episode.

## C Modality codecs and admission-gate fidelity

**Depth decoding.** For a valid metric depth $d \in (0.05, 10)$ meters, we first apply the monotone transform

$$y(d) = 1 - \left(1 - \frac{d}{\lambda c}\right)^{\lambda+1}, \qquad \lambda = -3, \quad c = \frac{10}{3}. \tag{10}$$

which is the power transform of Barron (2025). We then trace a piecewise-linear Hamiltonian path through the eight vertices of the RGB cube, following Gabeur et al. (2026):

$$\gamma_{\text{depth}}(d) = (1-\alpha)\mathbf{v}_k + \alpha \mathbf{v}_{k+1}, \qquad k = \lfloor 7y(d) \rfloor, \quad \alpha = 7y(d) - k, \tag{11}$$

where consecutive vertices differ in one RGB coordinate. At decoding time, each generated RGB pixel is projected onto its nearest path edge. The position along the path recovers $y$, after which the Barron transform is inverted to obtain metric depth. Pixels farther than a fixed tolerance from every path edge are marked *invalid* instead of being assigned an arbitrary depth value. This provides an explicit detector for generations that leave the valid measurement manifold.

**Surface normals.** Normals are computed from metric depth using camera-aware finite differences:

$$\widetilde{\boldsymbol{n}} = \left(-\frac{\partial d}{\partial u}\frac{|f_x|}{d}, -\frac{\partial d}{\partial v}\frac{|f_y|}{d}, 1\right), \qquad \boldsymbol{n} = \frac{\widetilde{\boldsymbol{n}}}{\|\widetilde{\boldsymbol{n}}\|_2}.$$

Each unit normal is then encoded as

$$\gamma_{\text{normal}}(\boldsymbol{n}) = \frac{\boldsymbol{n} + \mathbf{1}}{2}. \tag{12}$$

Decoding first maps RGB values back to $[-1, 1]^3$ and then renormalizes the result to unit length.

**Segmentation.** Segmentation uses a fixed lookup table from functional-role labels to RGB values, shared across tasks, episodes, frames, and camera views. The eight training roles are `background`, `target`, `goal`, `robot_arm`, `gripper`, `distractor`, `tool`, and `fixture`. A ninth `unknown` color is reserved as an annotation diagnostic and is rejected during data validation rather than used as a training class. The labels are functional roles a task assigns — what is being manipulated, what it is being manipulated towards, what merely populates the scene — so one color keeps one meaning across every task in the tree. Unlike a referring-mask representation, the prompt does not specify a per-episode color map; each color has a stable functional interpretation throughout training.

**Action images.** For the action representation in equation 9, the gripper center and two orientation-axis endpoints occupy the three RGB channels. Following Zhen et al. (2026), the low-response background of the blue channel additionally stores a binary gripper-openness pedestal. Decoding extracts the image-space peaks in each camera and fuses the corresponding rays to recover a 3D end-effector trajectory.

Table 4: Admission-gate results for each output stream. All streams are evaluated after a codec round-trip and, where applicable, a round-trip through the frozen VAE before training. Metrics are computed against the decoded ground-truth stream. The RGB control uses the same VAE and clips and provides an approximate upper bound on reconstruction quality.

| **Stream** | **Encoding** | **Codec round-trip** | **VAE round-trip** |
|---|---|---|---|
| Depth | Barron $\to$ RGB-cube path, equation 11 | AbsRel $0.066\%$ | AbsRel **0.362%**; $0.02\%$ off-path |
| Normal | $\frac{1}{2}(\boldsymbol{n}+1)$, equation 12 | mean cosine $0.999998$ | mean cosine **0.9906** ($7.9^\circ$) |
| Segmentation | Role LUT with a 9-color palette | IoU $> 0.98$ (spec.) | `target` IoU median **0.965** |
| Action | Three Gaussian channels, equation 9 | $\approx 4$ mm 3D | Not measured[†] |
| *RGB control (same VAE, same clips)* | | N/A | 29.6–31.5 dB PSNR |

[†] The action stream is rendered online rather than loaded from disk, so an equivalent VAE admission test requires a separate action-decoding harness and is not reported here.

## D TRAINING-MIXTURE DETAILS

The training distribution factorizes as

$$p(\Pi, \rho) = p(\Pi)\, p(\rho \mid \Pi),$$

where $\Pi$ determines the modality template and $\rho$ the conditioning plan.

**Template distribution.** XGENACT draws `video+action` with probability $0.4$ and each of `depth+action`, `segmentation+action`, and `normal+action` with probability $0.2$. The RGB-only control draws `video+action` with probability $1.0$. Every template in either mixture carries an action stream, and every template has the same four-segment shape and the same cost, so the two runs differ in the sampled observation space while retaining the same action-supervision budget.

**Conditioning plans.** An action stream is dropped with probability $0.1$ before the plan is drawn. Such a draw becomes a single-modality sample conditioned on first frames only: generate the whole depth, segmentation, normal or RGB video from its own first latent frame. The remaining $90\%$ of draws use the conditioning distribution of Table 5. The `1I1I` plan is applied only to RLBench samples; a ManiSkill3 sample falls through to the other two plans, which gives it $93.75\%$ `IIII` and $6.25\%$ `FIFI`.

The conditioning-plan distribution, optimization schedule, and action dropout are therefore matched across all runs; their training distributions differ only in the template menu.

Table 5: The shared conditioning distribution used by every trained model, including RGB, conditional on the action stream being retained (RLBench samples).

| | IIII | 1I1I | FIFI |
|---|---|---|---|
| Conditional | 75% | 20% | 5% |

## E FLOW-MATCHING AND OPTIMIZATION DETAILS

We retain the upstream flow-matching time sampler and weighting rule. A base time $\hat{t}$ is drawn uniformly from the discrete 1000-point schedule over $(0, 1]$ and shifted according to

$$t = \frac{s\hat{t}}{1 + (s-1)\hat{t}}, \qquad s = 5. \tag{13}$$

The timestep weight $w(t)$ in equation 6 is obtained by subtracting the minimum of $\exp[-2(t - \frac{1}{2})^2]$ over the discrete schedule and normalizing the resulting weights to have unit mean.

Unless otherwise stated, all runs warm-start from the released Action Images checkpoint at step 125,750. We fine-tune all parameters of the diffusion transformer (the 5B Wan2.2 backbone plus the per-block camera modules, 6.4B parameters in total) in bfloat16 mixed precision with ZeRO-2 and CPU-offloaded Adam, gradient checkpointing, batch size 1 per device on four GPUs and no accumulation — an effective batch of 4 — at learning rate $5 \times 10^{-7}$ with 1000 warmup steps, a constant schedule thereafter, gradient clipping at 1.0, and seed 42. The matched mixed-source runs use 6,000 optimization steps and save a checkpoint every 1,000 steps; the reported perception comparison reads all matched checkpoints at step 4,000. The VAE remains frozen throughout.

## F CAMERA CONDITIONING AND COORDINATE CONVENTIONS

Both data sources store, per view and per frame, a $4 \times 4$ camera-to-world extrinsic matrix and a $3 \times 3$ intrinsic matrix in the RLBench convention, whose camera frame has $x$ to the left, $y$ up and $z$ forward, so that $f_x$ and $f_y$ carry a negative sign. ManiSkill3 reports OpenCV parameters instead, which we convert once at generation time,

$$C_{\text{rlb}} = E_{\text{cv}}^{-1} \operatorname{diag}(-1, -1, 1, 1), \qquad K_{\text{rlb}} = K_{\text{cv}} \operatorname{diag}(-1, -1, 1),$$

so that a single projection routine serves both sources and the action codec of equation 8 needs no per-source branch. Camera conditioning itself follows Zhen et al. (2026): the per-view rays, relative to the first frame of view 1, are embedded as Plücker coordinates (Sitzmann et al., 2021; Bai et al., 2025) and encoded by the frozen VAE into the tensor $\mathbf{C}$. Each transformer block has its own camera encoder, whose output is added to the token features before self-attention; every segment reuses the camera of its view. The absolute extrinsics and intrinsics are used only to project and decode action images.

## G EVALUATION PROTOCOL

**Sampling configuration.** The matched XGENACT checkpoints use 50 denoising steps, classifier-free guidance 7.5, explicit modality tags, temporal stride 3, and seed 42. Their action images are decoded with the same sphere-constrained two-view solver.

**Future perception and the dynamic region.** A perception space of this model is only ever prompted alongside an action, so under `IIII` the anchor is frame 0 of that same modality and the scored frames are the ones that follow it. The baseline that carries the comparison is the *external-expert route*: the checkpoint's own `video+action` stream is sampled under the identical configuration, and a feed-forward specialist is then run on the frames it generated. The reported depth route uses the raw metric output of Depth Anything V2 Metric Indoor-Large with no fitted alignment parameters. Normals are scale-free and receive no fit; for segmentation the external expert is given the label vocabulary of the episode. *Copy-anchor* repeats the anchor at every future time and is retained as a protocol diagnostic, but is omitted from Table 7 for readability. The first four decoded poses in

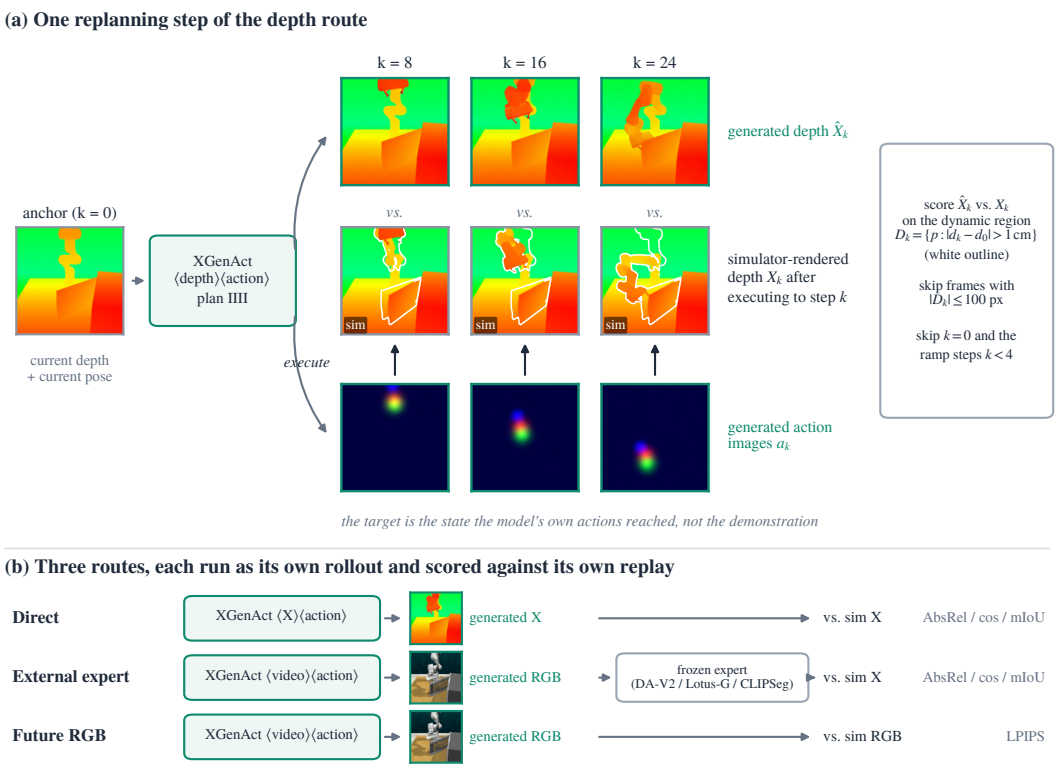



Figure 7: **Perception evaluation by simulator replay.** **(a)** One replanning step of the depth route (close microwave, view 1). From the current depth and pose, XGENACT generates a chunk of depth frames $\hat{X}_k$ and action images $a_k$. The decoded poses are executed, and after each step $k$ the simulator renders the depth $X_k$ of the state that the model's own actions reached (middle row, raw simulator output in the same colour code); $\hat{X}_k$ is scored against $X_k$ rather than against a demonstration. Scores use only the dynamic region $D_k$, outlined in white: pixels whose simulator depth changed by more than 1 cm since the chunk began, which includes both where the arm is and where it was. The anchor $k = 0$ and the ramp steps $k < 4$ are never scored. **(b)** The three routes compared in Table 7 and Figure 4. Each is a separate rollout scored against its own replay: direct generation of $X$, the same checkpoint's generated RGB passed through a frozen expert, and generated RGB itself.

each action chunk are replaced during execution by a ramp from the current pose to the fifth pose. Since these ramp poses are not predictions made by the model, scoring excludes them as well as the anchor at $k = 0$. For every modality, the *dynamic region* is defined from simulator depth as

$$\mathcal{D}_t = \{p : |d_t(p) - d_0(p)| > 0.01\ \text{m}\},$$

where $d_0$ is the simulator depth after the first executed step of the current chunk. The same region is used for RGB, depth, normals, and segmentation, and a frame is assigned a dynamic-region score only when $|\mathcal{D}_t| > 100$ pixels. Because this mask depends only on simulator ground truth, no prediction can influence which pixels are scored. Restricting evaluation to $\mathcal{D}_t$ prevents static background from dominating the metric.

**Main-table reporting details.** Direct rows in Table 7 appear only when the scored modality is present in the training mixture. For every checkpoint, the external-expert route uses its RGB output and then applies Depth Anything V2 Metric Indoor-Large, Lotus-G, or CLIPSeg. Future RGB has no external-expert route because it is already the intermediate representation consumed by the other cascades. It is scored against lossless simulator RGB dumped after every executed step, never against a demonstration future or an MP4-decoded frame. Table 7 reports RLBench evaluation only; ManiSkill3 contributes training data through the fixed 30% source share. A metric block excludes a task from every route's average whenever any shown cell in that block has a median commanded–achieved end-effector gap of at least 10 mm; the affected cells remain visible and are marked $^{\dagger}$. Every task cell uses five trials.

**Segmentation metrics.** **mIoU** is macro-averaged over the named labels present in the scored region. Because the live simulator annotations do not identify which distractor the instruction promotes to `target`, `target` is merged into `distractor` in both prediction and ground truth before scoring.

**Closed loop.** RLBench uses its rollout harness with receding-horizon replanning capped at $1.5\times$ the demonstration length and is evaluated with the corresponding control checkpoint.

## H EXTERNAL POLICY BASELINES

This appendix describes how the external policies in Table 1 are trained and evaluated. Every baseline runs from its official code base. Except for MolmoAct, which is evaluated zero-shot and adapted to our interface as in Action Images (Zhen et al., 2026), each is fine-tuned on the 16 RLBench training tasks of Appendix B, variation 0; none of the five evaluation tasks appears in this split. Table 6 summarizes the settings.

| Method | Training | Cam. | Chunk | Exec. | Action target |
|---|---|---|---|---|---|
| $\pi_{0.5}$ | 20,500 steps, batch 32 | × | 50 | 10 | $\Delta$SE(3) to current pose |
| LaWAM | 20,500 steps, batch 32 | × | 8 | 8 | absolute pose |
| Evo-1 | 20,500 steps, batch 32 | ✓ | 50 | 25 | per-step delta, accumulated |
| VLANeXt | 20,500 steps, batch 32 | ✓ | 8 | 8 | absolute pose |
| MolmoAct | zero-shot | ✓ | 16 | 16 | triangulated 2D trace |
| PAD-Depth | 2,560 steps, batch 256 | ✓ | 3 poses | 16 | absolute pose |
| UWM | 4,600 steps, batch 36 | ✓ | 16 | 8 | $\Delta$ to current pose |
| VLA-JEPA | 2,000 steps, batch 32 | ✓ | 7 | 4 | absolute pose |
| XGENACT | 4,000 steps, batch 4 | ✓ | 41 | 41 | action image |

Table 6: Training and execution settings of the methods in Table 1. *Cam.*: camera parameters are an input. *Chunk*: actions predicted per query; *Exec.*: actions executed before replanning.

**Camera parameters.** The training tree randomizes camera poses per episode (Appendix B), whereas rollouts use RLBench's fixed default cameras. XGENACT receives the camera through Plücker embeddings and its action projection (Appendix F). A baseline marked in Table 6 receives, per view, the $3 \times 4$ extrinsic matrix and the intrinsics $(f_x, f_y, c_x, c_y)$ rescaled to its input resolution, 16 values per view. The $\pi_{0.5}$ and LaWAM rows are camera-free variants.

**$\pi_{0.5}$, LaWAM, Evo-1, and VLANeXt.** These four policies share one training budget: $20{,}500$ steps at an effective batch size of 32 on the 16-task tree ($3{,}960$ episodes, $690{,}893$ frames at 20 fps), with $512^2$ renders resized to $224^2$. Each executes its action chunks as its official inference code does. $\pi_{0.5}$ predicts 50 SE(3) deltas relative to the current pose and executes 10. LaWAM and VLANeXt predict 8 absolute poses and execute the whole chunk. VLANeXt conditions on an 8-frame proprioceptive history, which its server rebuilds from the chunks it returned, because the client state is the last commanded pose. We load its weights with zero missing or unexpected keys; it keeps no EMA copy. Evo-1 executes the first 25 of 50 actions, as in both of its official clients. Its targets are per-step deltas, since each state in the data equals the previous commanded action, so a chunk is composed by accumulating the deltas. Its rotation deltas are computed by subtracting rotation vectors, which wrap at $\pm\pi$; $2.06\%$ of training frames therefore carry a rotation delta near $2\pi$. The reported Evo-1 result consequently reflects this data-conversion defect and should not be interpreted as an estimate of an otherwise correctly trained Evo-1 policy's capability.

**MolmoAct.** Following Zhen et al. (2026), we evaluate the released MolmoAct-7B-D checkpoint zero-shot with the reasoning prompt from its model card and adapt its output to our controller as follows. MolmoAct draws a 2D end-effector trace on its primary input image, so each replan queries it twice, once with each view as the primary image. The two traces are resampled by arc length to 16

corresponding points, triangulated with the true camera matrices, and anchored to the current end-effector position. Orientation is held at its current value; the gripper command is the mean of the two predicted gripper values, thresholded at 0.5. All 16 waypoints are executed before replanning.

**PAD-Depth.** PAD-Depth starts from the released Bridge-pretrained DiT-XL/2 (681M parameters). It uses the frozen SD VAE and CLIP text encoder and is fine-tuned on the `front` view at $256^2$ together with metric depth and the front camera parameters. It predicts $k = 3$ future frames and poses spaced 16 control steps apart, at global batch size 256 and learning rate $10^{-4}$. We evaluate the step-$2{,}560$ checkpoint, which has seen $655{,}360$ samples, the same exposure as the $20{,}500 \times 32$ budget above. We fixed three issues in the released training code. The depth loader referenced an undefined attribute. The action loss was disabled for the first $50{,}000$ steps; we enable it from step 0 with weight 1. The first future target was the current frame; we shift targets forward by one stride. Sampling uses 75 DDIM steps, as in the paper. Following the paper, only the first predicted pose is executed; it is linearly interpolated (orientation by slerp) into 16 waypoints to match the step budget of other policies.

**UWM.** UWM starts from the released DROID-pretrained checkpoint and is fine-tuned for $4{,}600$ steps at batch size 36 and learning rate $10^{-4}$ with delta actions. It observes the `front` and `overhead` views at $128^2$ over two frames, together with a 42-dimensional state: a 10-dimensional pose (position, 6D rotation, gripper) and 16 camera values per view. It predicts 16 actions and executes 8.

**VLA-JEPA.** VLA-JEPA starts from the released DROID-pretrained checkpoint (`VLA-JEPA-Pretrain`) and is fine-tuned with its LeRobot training code. Our actions are 10-dimensional poses and our state is the 42-dimensional vector used for UWM, whereas the pretrained model uses 7- or 8-dimensional ones, so the action encoder, action decoder and state encoder are re-initialized; all other weights, including the language backbone, are fine-tuned. The latent world-model objective stays enabled with its default weight of 0.1. The `front` and `overhead` views, at $224^2$, fill the two exterior-camera slots of the pretrained model. States are normalized by mean and standard deviation and actions by their range, with statistics computed on our training tree. We train with AdamW (learning rate $10^{-4}$, $\beta = (0.9, 0.95)$, gradient clipping at 1.0) under a cosine schedule with 5,000 warmup steps, in bf16, at an effective batch size of 32, and evaluate the step-$2{,}000$ checkpoint. The policy predicts 7 absolute poses and executes the first 4 before replanning.

**Closed-loop evaluation.** Each task is evaluated on variation 0 over 20 trials, and success is the task-success signal reported by RLBench. MolmoAct, PAD-Depth, UWM, and VLA-JEPA run in the same rollout harness as XGENACT, with scene seeds derived from `(task, variation, trial)`, $512^2$ renders, and RLBench's planning-based end-effector controller. An episode ends at success, at $1.5\times$ the demonstration length, or after 20 consecutive inverse-kinematics failures; XGENACT uses the same budget but ends after 20 inverse-kinematics failures in total. $\pi_{0.5}$, LaWAM, Evo-1, and VLANeXt are served behind a common WebSocket interface and driven by one model-agnostic RLBench client; their episodes end at success, after 200 executed actions, or after 20 inverse-kinematics failures in total.

## I PERCEPTION RESULTS UNDER SIMULATOR REPLAY

Table 7 gives the per-task scores behind Figure 4, obtained with the protocol of Appendix G and Figure 7.

## J ADDITIONAL QUALITATIVE RESULTS

This section collects one or two representative examples for each qualitative result of Section 4. Frames are unedited simulator renders or raw generated canvases; generated frames are outlined in green and simulator or ground-truth frames in grey. Trials are selected by the fixed rule stated in each caption, never by inspecting the baselines' outcomes.

| Training modalities | Route | RLBench | | | | |
|---|---|---|---|---|---|---|
| | | Close microwave | Toilet seat down | Close box | Meat on grill | Gated avg. |
| **Future RGB** — LPIPS ↓ | | | | | | |
| RGB | Direct | **0.41** | 0.55 | **0.38** | 0.49 | **0.39** |
| RGB+D | Direct | 0.46 | 0.50† | 0.39 | 0.53 | 0.42 |
| RGB+S | Direct | 0.45 | 0.51 | 0.39 | 0.46 | 0.42 |
| RGB+N | Direct | 0.46 | 0.50 | 0.38 | 0.54 | 0.42 |
| RGB+D+S | Direct | 0.42 | 0.48 | 0.38 | 0.47 | 0.40 |
| RGB+N+S | Direct | 0.47 | **0.47**† | 0.39 | 0.52 | 0.43 |
| RGB+D+N | Direct | 0.44 | 0.48 | 0.39 | 0.37† | 0.42 |
| RGB+D+N+S | Direct | 0.41 | 0.53† | 0.41 | **0.37** | 0.41 |
| **Metric depth** — dynamic-region AbsRel ↓ | *external expert: Depth Anything V2* | | | | | |
| RGB+D | Direct | 0.34 | 0.57† | 0.27 | 0.38 | 0.30 |
| | External expert | 0.44 | 0.49† | 0.42 | 0.59 | 0.43 |
| RGB+D+S | Direct | **0.27** | 0.52 | 0.39 | 0.40 | 0.33 |
| | External expert | 0.39 | 0.49 | 0.42 | 0.57 | 0.41 |
| RGB+D+N | Direct | 0.32 | 0.35 | **0.25** | 0.33 | 0.28 |
| | External expert | 0.42 | 0.50 | 0.42 | 0.51† | 0.42 |
| RGB+D+N+S | Direct | 0.28 | **0.31**† | 0.25 | **0.23** | **0.26** |
| | External expert | 0.39 | 0.53† | 0.42 | 0.49 | 0.41 |
| **Surface normal** — dynamic-region mean cosine ↑ | *external expert: Lotus-G* | | | | | |
| RGB+N | Direct | 0.76 | 0.68† | 0.78 | 0.70† | **0.77** |
| | External expert | 0.75 | **0.72** | **0.78** | 0.69 | 0.77 |
| RGB+N+S | Direct | 0.74 | 0.70† | 0.78 | 0.69 | 0.76 |
| | External expert | 0.75 | 0.71† | 0.78 | 0.72 | 0.76 |
| RGB+D+N | Direct | 0.72 | 0.68 | 0.71 | 0.74 | 0.71 |
| | External expert | 0.75 | 0.71 | 0.77 | 0.75† | 0.76 |
| RGB+D+N+S | Direct | 0.73 | 0.68 | 0.73 | 0.75 | 0.73 |
| | External expert | 0.76 | 0.70† | 0.77 | **0.77** | 0.76 |
| **Segmentation** — dynamic-region mIoU ↑ | *external expert: CLIPSeg* | | | | | |
| RGB+S | Direct | **0.53** | **0.42** | **0.45** | **0.34**† | **0.49** |
| | External expert | 0.19 | 0.21 | 0.28 | 0.20 | 0.23 |
| RGB+D+S | Direct | 0.42 | 0.38 | 0.42 | 0.33 | 0.42 |
| | External expert | 0.21 | 0.23 | 0.26 | 0.20 | 0.24 |
| RGB+N+S | Direct | 0.49 | 0.41 | 0.44 | 0.30 | 0.46 |
| | External expert | 0.18 | 0.20† | 0.26 | 0.16 | 0.22 |
| RGB+D+N+S | Direct | 0.39 | 0.32 | 0.43 | 0.32 | 0.41 |
| | External expert | 0.26 | 0.21† | 0.27 | 0.25 | 0.26 |

Table 7: **Perception under simulator replay on four unseen RLBench tasks.** Each task cell averages five trials at step 4,000 and is scored on the dynamic region against the simulator replay induced by that route's decoded actions. For checkpoints that natively predict a modality, adjacent rows compare direct generation with the same checkpoint's external- expert route; the latter first predicts RGB and then applies the frozen expert named in the block heading. Within a metric block, the gated average removes a task from every route if any shown cell has a commanded–achieved execution gap of at least 10 mm; those cells remain visible and are marked †. Bold and underline mark the best and second-best values, respectively, using the unrounded scores. Arms 0–6 share the 70/30 RLBench–ManiSkill3 training mixture. The available all-modality arm-7 checkpoint is an RLBench-only contextual result with a different data exposure and training horizon, and is therefore not a strictly matched ablation.

### J.1 Closed-loop rollouts against external policies

Figures 8 and 9 expand Figure 3 with the same runs as Table 1 for XGenAct, PAD-Depth, UWM, VLA-JEPA and MolmoAct; $\pi_{0.5}$, LaWAM, Evo-1 and VLANeXt run behind a separate client (Appendix H) whose episodes were not recorded as video. Each panel fixes one trial, so all methods start from the same scene, and shows frames at $25/50/75\%$ of each method's own episode and its final frame, with the outcome and termination reason. The labels give each method's success rate over the 20 trials.

### J.2 Paired rollouts of RGB and XGenAct

Figure 10 accompanies the menu ablation of Section 4.3: for three unseen tasks it shows an RGB rollout and a XGenAct rollout started from the same task instance and scene seed.

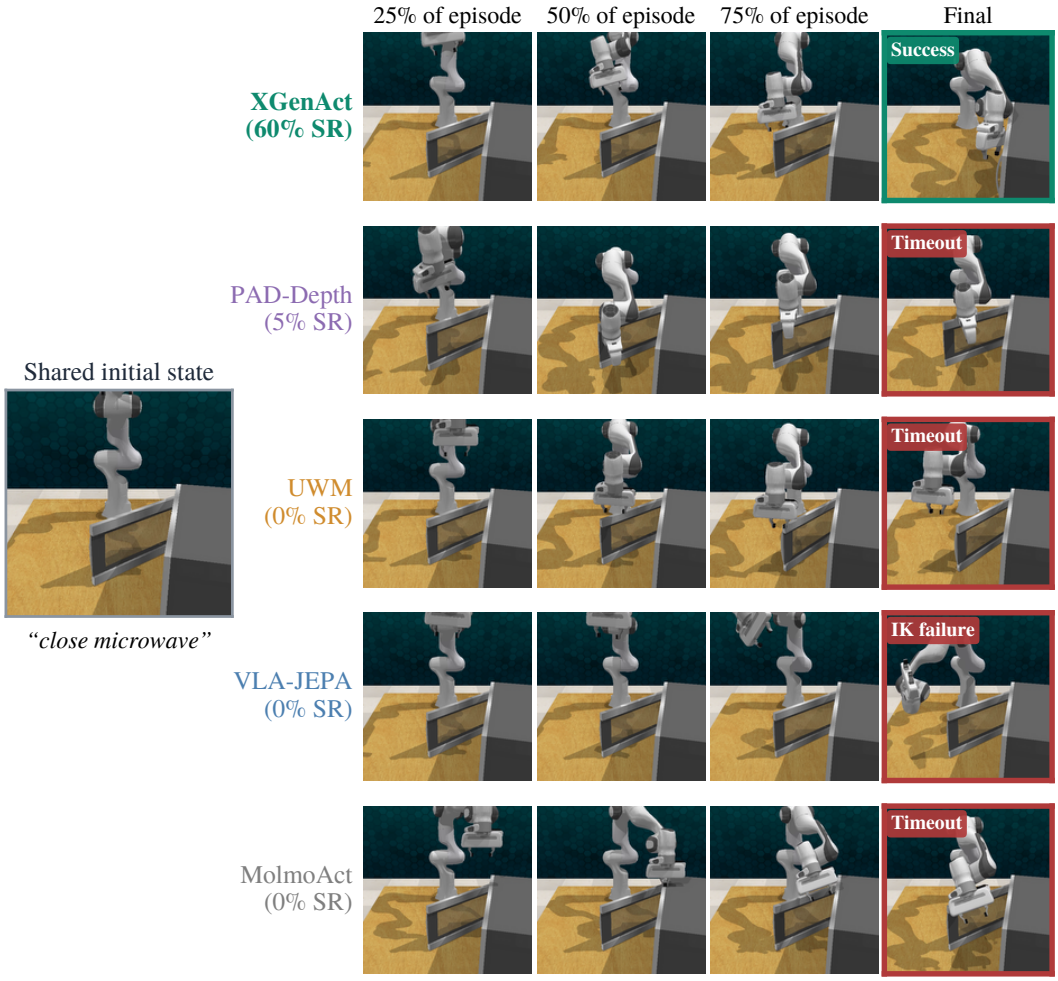
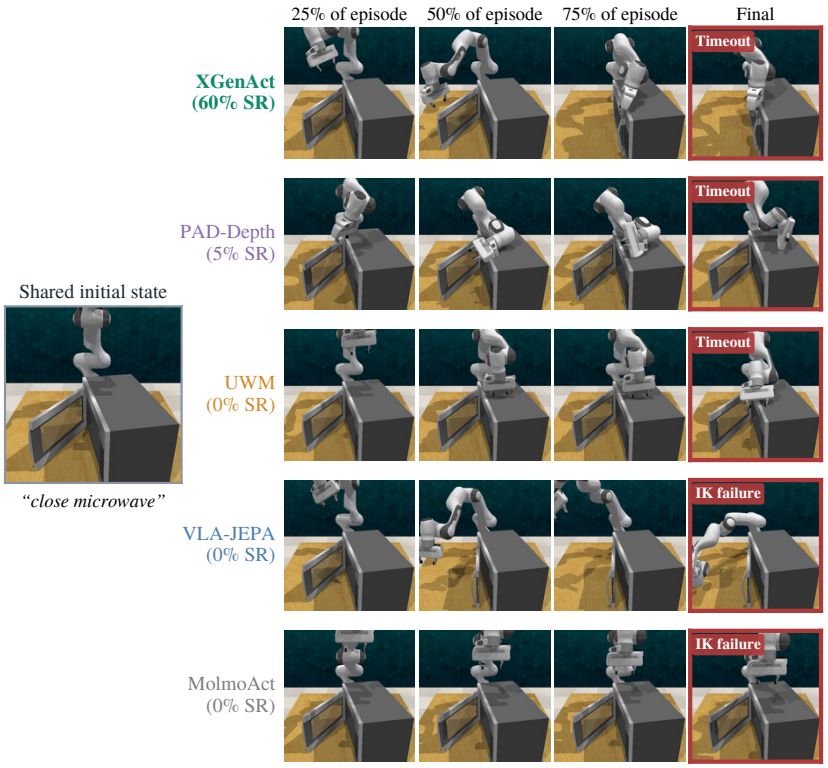
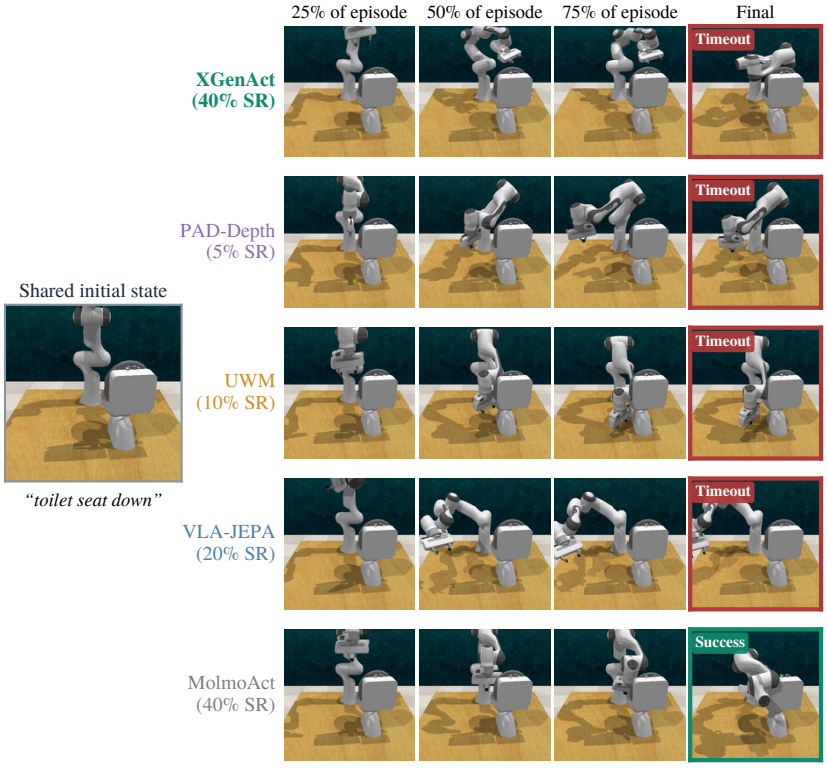



Figure 8: **Close microwave**, trial 0 (the first trial XGENACT solves). XGENACT closes the door; PAD-Depth, UWM and MolmoAct time out, and VLA-JEPA stops on repeated inverse-kinematics failures.




(a) Close microwave, trial 3




(b) Toilet seat down, trial 0

Figure 9: **Failure cases of XGENACT**: the first trial of each task that XGENACT does not solve. Both failures are timeouts; on the same scenes every baseline also fails, except MolmoAct in (b).

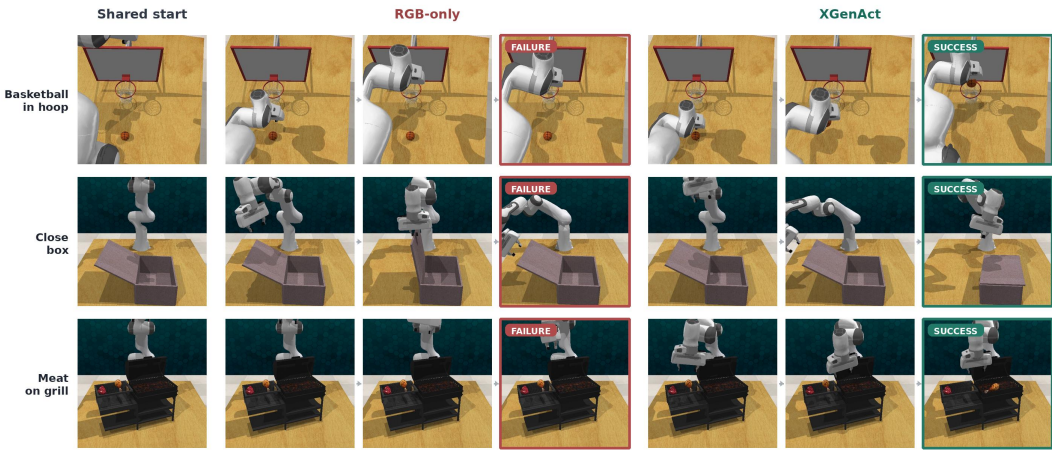



Figure 10: **Paired closed-loop executions on RLBench.** Each row uses the same task instance and scene seed for RGB-only and XGENACT. The basketball row is shown from a recording-only right-shoulder camera; both policies retain the original front and overhead inputs used throughout the closed-loop evaluation.

### J.3 IMAGINED FUTURES VERSUS EXECUTED ROLLOUTS

During a closed-loop rollout, every replanning step generates a chunk of future RGB and action images, the action images are decoded to poses, and the robot executes them. The dumps of Appendix I record, for each executed step $k$, which frame of which generated chunk it came from and what the simulator rendered after executing it, so the two can be compared frame by frame. Figure 11 shows both views of one such chunk, complementing Figure 6; it uses the all-modality checkpoint prompted with `<video><action>`, the first chunk of the lowest-index successful trial, and omits the first four steps, which the rollout replaces by a ramp from the current pose.

### J.4 OPEN-LOOP GENERATION ACROSS TASKS

The last group is open loop. For demonstration episodes of training and held-out tasks, the all-modality checkpoint trained on the 70/30 RLBench–ManiSkill3 mixture (step 4,000) receives only the frame-0 observation and the current end-effector pose of two views. With plan `IIII` it generates 40 future frames, once per template (50 flow steps, guidance 7.5, seed 42); the ground truth is the demonstration. Figure 12 shows the last generated frame in every output space, and Figure 13 decodes the generated action images into end-effector paths on training and held-out tasks.

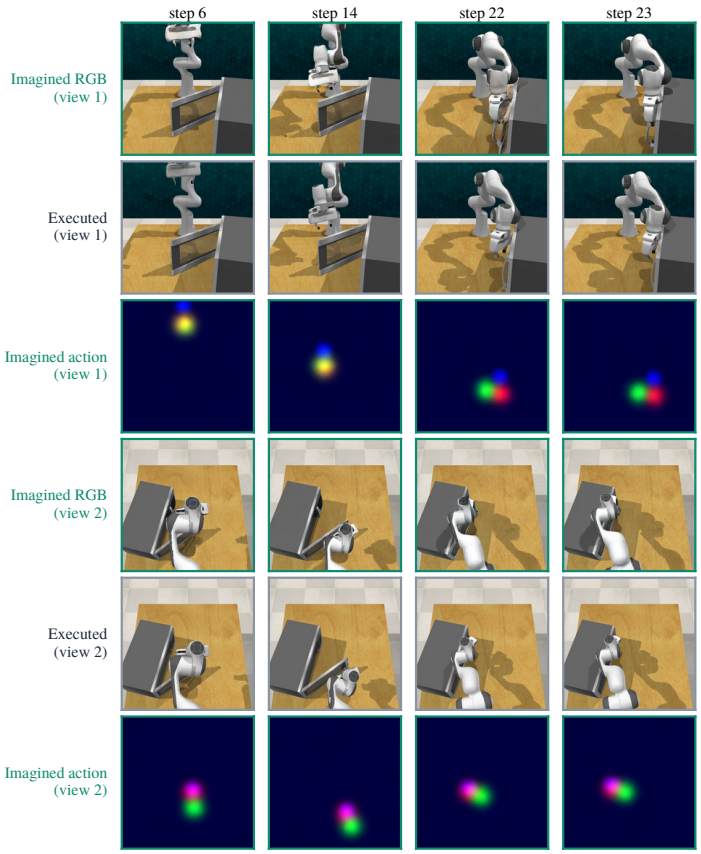



Figure 11: **Imagination versus execution, close microwave** (trial 0, both views). For each view: the generated RGB frame, the simulator after executing the decoded actions up to the same step, and the generated action image that produced them.

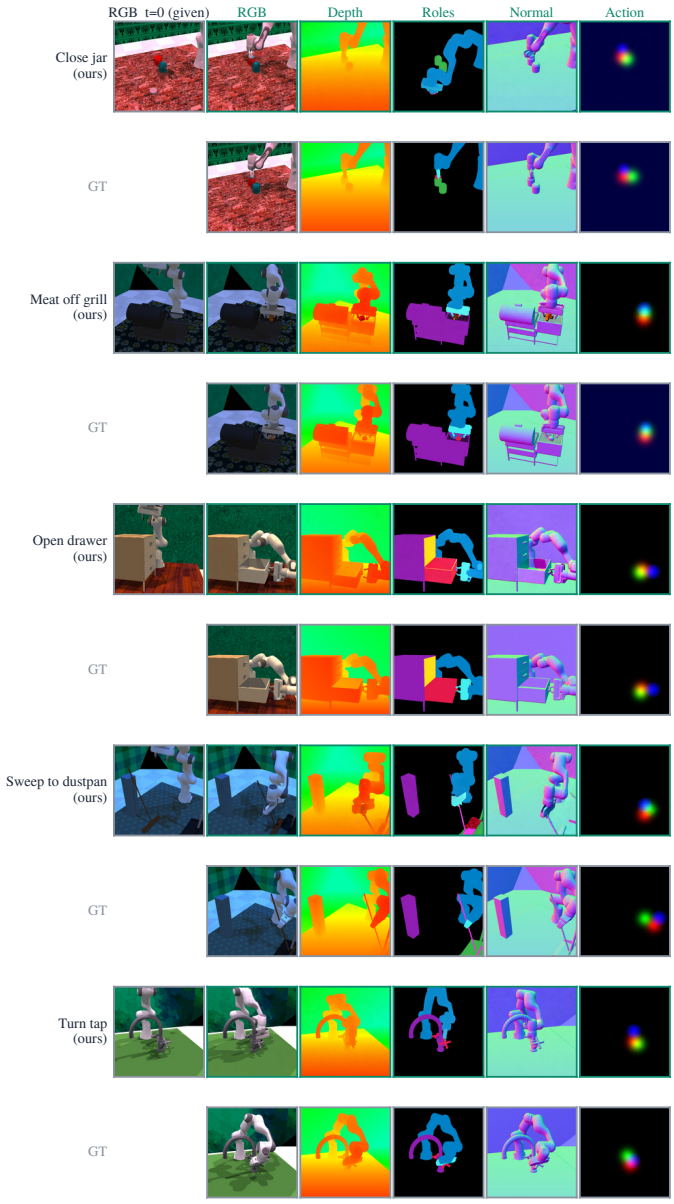



Figure 12: **Open-loop generation across output spaces.** For each episode, the upper row is generated (frame 40 of 40, view 1) and the lower row is the demonstration. Each output space is a separate prompt of the same checkpoint, so the arm pose can differ slightly between columns.




Figure 13: **End-effector paths decoded from generated action images** (red) versus the demonstration (green, dashed), drawn on the frame-0 observation of both views; the top two rows are training tasks and the bottom two held-out tasks. Each point is the centroid of the position blob of one frame; open circle: start, square: end, ring: first frame with the gripper closed. The corner value is the mean 2-D distance between the two paths in pixels ($512^2$ images); it is not the 3-D pose error of the tables.