LEARNING TO ACT THROUGH SPARSE 4D QUERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Successful manipulation requires a policy to understand where things are and how they will move, yet action demonstrations leave this spatiotemporal structure implicit. Existing methods supply it through dense supervision: world action models predict every pixel of future frames, and 3D-aware vision-language-action (VLA) models align every token with teacher features, both at substantial training cost. We ask whether sparse, explicit supervision suffices. We introduce SparQ4D, which casts depth, 2D/3D point motion, multi-view correspondence, and hand-eye relations as a single D4RT-style question (*where is a selected point, at a specified time, in a specified view?*), which is answered by a lightweight query decoder attached to a pretrained VLA. Each training step samples only a small query budget, and the decoder is discarded after training, leaving the VLA's architecture and inference cost unchanged. Across two VLA backbones in simulation and the real world, SparQ4D consistently improves manipulation: it raises the success rate of $\pi_{0.5}$ from 77.6% to 92.6% on RoboTwin 2.0, reducing over 60% failures, and outperforms X-WAM with 20% of its training GPU-hours and 6.6× faster inference. Probing further shows that the trained VLA itself learns recoverable 4D structure.

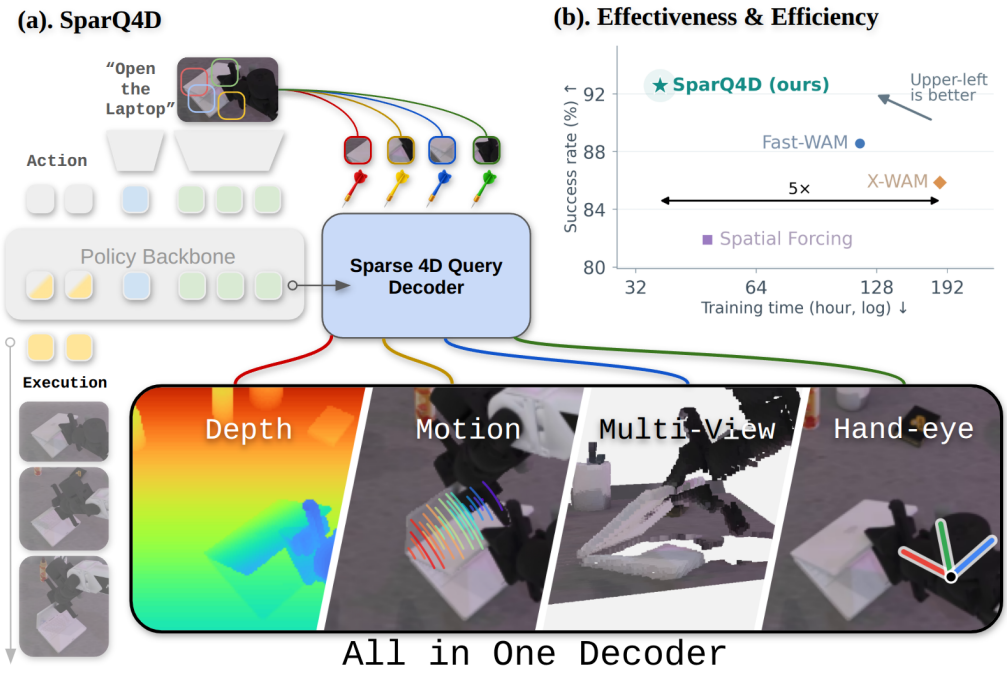


Figure 1: **Overview of SparQ4D. (a)** We supervise a policy with sparse 4D queries (darts), decoded by a lightweight decoder attached to its intermediate representations. **(b)** Success rate versus training time on RoboTwin 2.0. SparQ4D achieves the highest success rate at the lowest training cost, compared with world action models (Fast-WAM, X-WAM) and 3D representation alignment (Spatial Forcing).

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.

1 INTRODUCTION

Successful manipulation is inherently spatiotemporal: it depends on knowing where things are and how they will move. To hang a mug, a robot must grasp it, align its handle with a rack, and relate wrist and external camera views as it moves. Action demonstrations supervise the desired behavior but leave these relationships implicit. Efficiently teaching a policy this spatiotemporal structure is thus a central question.

World action models (WAMs) address this question by jointly learning future visual observations and robot actions (Ye et al., 2026), and X-WAM (Guo et al., 2026) extends this paradigm to multi-view RGB-D prediction. Reconstructing the full future supervises scene dynamics richly, but requires dense prediction over many visual tokens, substantially increasing training cost. Fast-WAM (Yuan et al., 2026) removes future generation at inference yet retains strong performance, suggesting that much of the benefit comes from the training signal itself. Another line injects geometric knowledge into vision-language-action models (VLAs) through pretrained 3D representations: R3DP (Zhang et al., 2026b) incorporates foundation-model features via a fast-slow perception system, and Spatial Forcing (Li et al., 2026a) aligns VLA representations with 3D teacher features during training. Although effective, these methods likewise rely on dense, per-token supervision. This leads to our central question: **can sparse, explicit supervision teach a robot policy both geometry and motion, at substantially lower training cost and with no additional inference computation?**

We introduce SparQ4D, a policy learning framework that supervises geometry and motion through unified sparse queries (Fig. 1). Inspired by D4RT (Zhang et al., 2026a), we cast multiple auxiliary tasks into one question: *where is a selected point at a specified time, in a specified view or coordinate frame?* Each *query* specifies the point, target time, and reference frame, and a lightweight query decoder predicts the point's location. Varying these attributes yields supervision for depth, 2D/3D point motion, multi-view correspondence, and hand-eye relations. We attach the decoder to intermediate representations of a pretrained VLA and jointly optimize auxiliary prediction and action learning, so a single decoder supports diverse geometric and motion supervision. Crucially, this formulation requires no dense reconstruction. Each training step samples only a small number of queries across points, times, and views, and the query budget directly controls the auxiliary decoding cost, providing diverse geometric and temporal supervision without predicting every pixel or future frame. The decoder is discarded after training, leaving the original VLA architecture and inference cost unchanged. SparQ4D thus supervises spatiotemporal structure explicitly while avoiding dense auxiliary generation.

We evaluate SparQ4D on seven RoboTwin 2.0 (Chen et al., 2026b) tasks and five real-world tasks. Built on $\pi_{0.5}$ (Black et al., 2025a), SparQ4D improves the average success rate by **19.3%** (77.6% to 92.6%, reducing 60% failures) on RoboTwin and by **18.3%** (60% to 71%) on the real robot, with especially large gains on tasks requiring precise spatial alignment, such as mug hanging. Compared with X-WAM, it performs better with only **20% of the training GPU-hours** and $6.6\times$ **faster** inference. To understand these gains, we probe the frozen VLA representations and find recoverable 4D structure, including temporal correspondence, cross-view relations, and hand-eye geometry, suggesting that the policy itself internalizes this structure rather than delegating it to the auxiliary decoder.

Our contributions are threefold:

- We introduce SparQ4D, which unifies depth, point motion, multi-view correspondence, and hand-eye relations as explicit supervision for VLAs through a shared query decoder.
- We show that sparse supervision suffices: sampling a small query budget per step controls training-time auxiliary decoding cost, and since the decoder is removed after training, policies can be improved with small additional training cost and no inference computation.
- We demonstrate improved manipulation across two VLA backbones in simulation and the real world, and systematically examine supervision content, density, placement and internal representation through controlled ablations.

2 Related Work

World action models. Video-prediction policies such as UniPi and GR-1 (Du et al., 2023; Wu et al., 2024) first used future-frame generation as a training signal for control, and WAMs scale this idea with large video foundation models (Ye et al., 2026; Li et al., 2026b). Later variants either predict depth or other 4D modalities alongside RGB, as in TesserAct and X-WAM (Zhen et al., 2025; Guo et al., 2026; Bi et al., 2026; Li et al., 2026c; Xu et al., 2026), or remove or reduce generation at inference, as in Fast-WAM and AHA-WAM (Yuan et al., 2026; Cai et al., 2026). Either way, training remains dense prediction over future frames; SparQ4D instead supervises through sparse, explicit queries, reducing training cost and adding no inference cost.

Geometric representations for policies. Another line imports geometry from pretrained models. PerAct and RVT (Shridhar et al., 2023; Goyal et al., 2023; Ze et al., 2024) take depth, point clouds, or voxels as input, and R3DP (Zhang et al., 2026b) feeds 3D foundation-model features to a VLA through a fast-slow perception system; both add inputs or encoders at deployment. Alignment methods, following REPA (Yu et al., 2025), instead distill frozen-teacher features into the VLA's hidden states during training: Spatial Forcing (Li et al., 2026a) aligns with 3D foundation models, other work with video or 4D models (Ding et al., 2026; Yang et al., 2026), and FLARE (Zheng et al., 2025) with predicted future latents. These signals are dense over every token, and alignment supervises implicit features rather than named quantities; SparQ4D supervises explicit geometric and motion quantities at sparse points, adding nothing at deployment.

Table 1: Positioning of SparQ4D against the methods we compare with. ✓: satisfied; ○: partially or setting-dependent; ✗: not satisfied.

Method (paradigm)	Explicit target	No dense supervision in training	Multi-target, one interface	No extra cost at inference
X-WAM (Guo et al., 2026) (WAM + depth)	✓	✗	○	✗
Fast-WAM (Yuan et al., 2026) (WAM + efficiency)	✓	✗	✗	✓
Spatial Forcing (Li et al., 2026a) (alignment)	✗	✗	✗	✓
SparQ4D (ours)	✓	✓	✓	✓

3 Method

SparQ4D adds an auxiliary objective to policy learning: while the policy learns to predict actions, its backbone is also trained to answer sparse *4D queries* about the current observation (Fig. 2). Designing this objective involves three questions: what to ask the policy, how to read out its answers, and how to train both tasks together. Sec. 3.1 answers the first by defining the query that unifies depth, motion, multi-view, and hand-eye supervision, Sec. 3.2 the second with a lightweight decoder on the policy's backbone, and Sec. 3.3 the third.

Background. We build on $\pi_{0.5}$ (Black et al., 2025a), a vision-language-action (VLA) policy π_θ that maps the proprioceptive state $\mathbf{s}_t$, the camera images $\mathbf{I}_t = \{\mathbf{I}_t^c\}_{c\in\mathcal{C}}$, and a language instruction ℓ to an action chunk $\mathbf{A}_t = [\mathbf{a}_t, \dots, \mathbf{a}_{t+H-1}]$, trained with flow matching (Lipman et al., 2023; Black et al., 2025b):

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{\tau,\boldsymbol{\epsilon}} \left\| \mathbf{v}_\theta\big(\tau\mathbf{A}_t + (1-\tau)\boldsymbol{\epsilon};\, \mathbf{s}_t, \mathbf{I}_t, \ell\big) - (\mathbf{A}_t - \boldsymbol{\epsilon}) \right\|^2, \tag{1}$$

where $\boldsymbol{\epsilon}$ is standard Gaussian noise and $\tau \in [0, 1]$. In our bimanual setting, $\mathcal{C}$ consists of a head camera and one wrist camera per arm, and $\mathcal{G}$ denotes the two grippers. Internally, a vision-language backbone encodes $(\mathbf{s}_t, \mathbf{I}_t, \ell)$ into layer-wise prefix tokens $\mathbf{Z}_t$, from which an action expert decodes $\mathbf{A}_t$; SparQ4D leaves $\mathcal{L}_{\text{act}}$ unchanged and adds an auxiliary objective on $\mathbf{Z}_t$.

3.1 4D Queries

Query. D4RT (Zhang et al., 2026a) asks where a point observed in one frame lies at another time and in another camera frame, but all queried frames belong to its input video, so answering

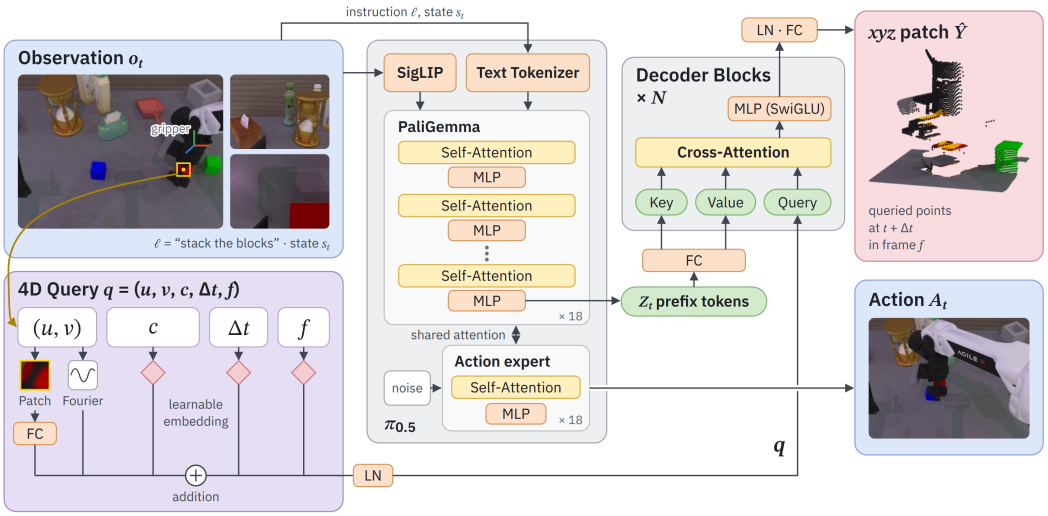


Figure 2: **Overview of SparQ4D.** Alongside action prediction, the backbone of $\pi_{0.5}$ is trained to answer sparse 4D queries $\mathbf{q} = (\mathbf{u}, \mathbf{v}, c, \Delta t, f)$: where the point at pixel $(\mathbf{u}, \mathbf{v})$ of camera c is at time $t + \Delta t$, in frame f. Each query is embedded as one token, and a lightweight decoder cross-attends to the prefix tokens $\mathbf{Z}_t$ to predict its 3D patch $\hat{\mathbf{Y}}_\mathbf{q}$. The decoder is used only during training, so the deployed policy is unchanged.

is reconstruction. A policy instead observes only the current time step and must anticipate what happens next. We therefore define a 4D query at time t as

$$\mathbf{q} = (\mathbf{u}, \mathbf{v}, c, \Delta t, f), \tag{2}$$

which asks: *where **is or will** the point currently seen at pixel* $(\mathbf{u}, \mathbf{v})$ *of camera* c *be at time* $t + \Delta t$*, in reference frame* f*?* Here $c \in \mathcal{C}$ is the source camera, $\Delta t \in \{0, \dots, H-1\}$ is a time offset within the action horizon, and $f \in \mathcal{F} = \mathcal{C} \cup \mathcal{G}$ is a camera or a gripper, taken at the current time t. Camera poses are never given as input, so they must be inferred from the observation. The target is the 3D patch $\mathbf{Y}_\mathbf{q}$ of the point at $t + \Delta t$, expressed in frame f.

One interface for all queries. The same question, with different Δt and f, covers the four kinds of geometry that manipulation relies on (Tab. 2):

- **Depth**: where is this point *now*, in the current camera frame?
- **Motion**: where *will* this point be at $t + \Delta t$, in the current camera frame?
- **Multi-view**: where is this point now, in *another* camera's frame?
- **Hand-eye**: where is this point now, in a *gripper*'s frame?

Table 2: **Query types.** Each type is an instance of $\mathbf{q} = (\mathbf{u}, \mathbf{v}, c, \Delta t, f)$. Hand–eye queries use the head camera as source, since wrist cameras are rigidly mounted on their grippers; the other types use any camera.

Type	Offset Δt	Reference f	Ability
Depth	0	c	metric structure
Motion	$1, \dots, H-1$	c	future motion
Multi-view	0	$c' \in \mathcal{C} \setminus \{c\}$	relative camera pose
Hand-eye	0	$g \in \mathcal{G}$	gripper pose

3.2 Query Decoder

Architecture. Each query is embedded as one token: the sum of a linear embedding of the $p \times p$ RGB patch around $(\mathbf{u}, \mathbf{v})$ in $\mathbf{I}_t^c$, a positional encoding of $(\mathbf{u}, \mathbf{v})$, and learned embeddings of c, Δt,

and f. The decoder $\mathcal{D}_\phi$ is a small transformer (1.6% of the policy's parameters) in which query tokens cross-attend to the prefix tokens $\mathbf{Z}_t$, i.e., the image tokens of all cameras and the language tokens, but not to each other, so each query is decoded independently. A linear layer maps each output token to a predicted patch $\hat{\mathbf{Y}}_\mathbf{q} = \mathcal{D}_\phi(\mathbf{q}; \mathbf{Z}_t)$.

3.3 Training and Inference

During training, we sample M queries per frame, split equally among the four types, with $(\mathbf{u}, \mathbf{v})$ sampled preferentially near depth discontinuities and partly uniformly to cover flat regions (App. A.3). For motion queries, $(\mathbf{u}, \mathbf{v})_{t+\Delta t}$ is obtained from CoTracker3 (Karaev et al., 2025) run offline on the demonstrations, and Δt is sampled uniformly from tracked offsets. We jointly optimize the policy and decoder with

$$\mathcal{L}_{\text{geo}} = \frac{1}{|\mathcal{Q}|} \sum_{\mathbf{q} \in \mathcal{Q}} \frac{1}{3p^2} \left\| \hat{\mathbf{Y}}_\mathbf{q} - \mathbf{Y}_\mathbf{q} \right\|_1, \tag{3}$$

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda \mathcal{L}_{\text{geo}}, \tag{4}$$

where $\mathcal{Q}$ excludes queries with invalid depth or untracked points. The geometry loss averages the absolute error over all valid queries and their $3p^2$ coordinate values. Both losses update the backbone, with λ weighting geometric supervision.

At inference, the decoder is removed, leaving the underlying VLA's inputs, policy architecture, and inference cost unchanged. Since SparQ4D interfaces with the policy only through prefix tokens, it also extends to other VLAs with action heads conditioned on a vision-language backbone; we additionally apply it to π_0 (Black et al., 2025b).

4 Experiments

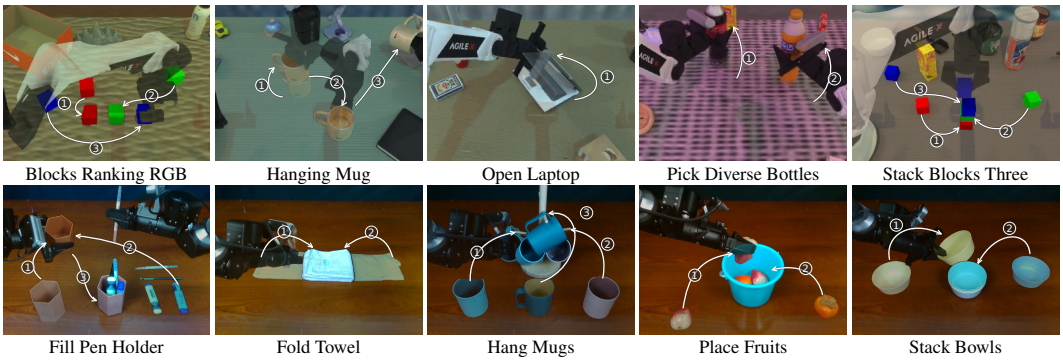


Figure 3: Representative task setups in RoboTwin 2.0 (top row) and the real world (bottom row).

Our experiments test both effectiveness and efficiency, and examine why SparQ4D works. We organize them around five questions:

- **RQ1 (Effectiveness, Simulation):** Does SparQ4D improve manipulation success over strong VLA and WAM baselines in simulation?
- **RQ2 (Effectiveness, Real World):** Does this improvement hold on a physical robot?
- **RQ3 (Efficiency):** Is SparQ4D cheaper to train and faster to deploy than dense video-based and implicit-alignment baselines?
- **RQ4 (Ablation):** Which query types and architectural choices drive this improvement?
- **RQ5 (Analysis):** Does SparQ4D make the VLA representation geometry-aware?

4.1 Experimental Setup

Training. In simulation, we train on RoboTwin 2.0 (Chen et al., 2026b) using both 50 clean-setting and 500 randomized-setting demonstrations per task, for 5 epochs with eight NVIDIA H20 GPUs;

depth queries are supervised with rendered ground-truth depth. On the real robot, we collect 5 tasks $\times$ 50 episodes each and supervise depth queries with MoGe-3 (Kong et al., 2026) estimates (see App. C for more details). Some representative tasks are shown in Fig. 3.

Evaluation. In simulation, we roll out each method 100 times per task under both *Clean* and *Randomized* settings. On the real robot, we roll out 25 episodes per task. We report success rate (SR) throughout. For real-world tasks, we additionally report progress score (PS), a partial-credit metric defined in App. C.3.

Baselines. All baselines are initialized from their publicly released pretrained weights and fine-tuned on RoboTwin under the same protocol as ours. We compare against three groups: (i) π_0 and $\pi_{0.5}$ (Black et al., 2025b;a), action-only VLA baselines; (ii) Spatial Forcing (Li et al., 2026a), an implicit-alignment baseline (Sec. 2) built on π_0, for which we use the implementation from XPolicyLab (XPolicyLab Community et al., 2026); and (iii) X-WAM (Guo et al., 2026) and Fast-WAM (Yuan et al., 2026), two WAM baselines built on the Wan2.2 (Wan et al., 2025) video foundation model.

4.2 RQ1: Does SparQ4D improve manipulation success in simulation?

Table 3 reports success rates on 7 RoboTwin 2.0 tasks under the *Clean* and *Randomized* settings.

SparQ4D improves both backbones. SparQ4D raises the average success rate of π_0 and $\pi_{0.5}$ by roughly 25 and 15 points, respectively, in both settings. Spatial Forcing, which also injects 3D information into π_0, is competitive in simulation but not in the real world (Sec. 4.3). Both action-only VLAs trail the two WAMs we report for reference, which learn spatiotemporal structure by generating future video; with SparQ4D, $\pi_{0.5}$ closes this gap without generating video.

Gains concentrate where 3D alignment matters. We call a task geometry-aware if success requires precise 3D alignment among the gripper, object, and target, such as grasping a bottle deep enough to hold it (`pick_diverse_bottles`). These tasks are the hardest for action-only baselines and benefit most from SparQ4D: with $\pi_{0.5}$, the average gain on them is several times that on the remaining tasks, and `hanging_mug` goes from rarely solved to solved in most trials. The remaining tasks still improve consistently, and with the weaker π_0, gains are large across all tasks. Moreover, most of the failures eliminated by SparQ4D occur during grasping and placing. Explicit geometric supervision thus mainly sharpens where the policy makes contact, rather than how it moves in between.

4.3 RQ2: Does SparQ4D improve manipulation success in real world?

Table 4 reports success rate (SR) and progress score (PS) on 5 real-world tasks (Fig. 3).

SparQ4D improves both backbones, most clearly where precise contact matters. SparQ4D improves both backbones on average in both SR and PS, while Spatial Forcing only matches π_0 in the real world. With $\pi_{0.5}$, SparQ4D achieves the best overall result, raising the average SR from 60% to 71% and improving or matching the backbone on every task. Part of this gain is masked by the strong base policy, which already handles the coarse-placement tasks well (`place_fruits`, `stack_bowls`); still, a consistent improvement on a state-of-the-art VLA, at little extra training cost and no inference cost, is valuable in practice.

4.4 RQ3: Is SparQ4D More Efficient to Train and Deploy?

SparQ4D is cheaper to train and deploy. Compared with the two WAMs, SparQ4D ($\pi_{0.5}$) trains 3–5$\times$ faster and runs inference 1.7-6.6$\times$ faster, with about a quarter of their parameters and 24-41% of their training memory (Tab. 5); since the query decoder is used only during training, it adds no inference overhead to the backbone. Compared with Spatial Forcing, whose costs are otherwise similar, SparQ4D also avoids a frozen VGGT teacher (Wang et al., 2025), whose features must either be cached offline, requiring large storage, or computed online, which lengthens training (42 vs. 32 GPU-hours).

Table 3: Results on 7 RoboTwin tasks. Each per-task cell reports the number of successful trials out of 100 (top: Clean, bottom: Random); Avg columns report the mean success rate (%) over the 7 tasks. **Bold** and *italics* denote the best and second-best result in each column, respectively (ties are all marked). Rows shaded in gray are ours. All models in this table are trained 5 epochs.

		Avg. SR ↑		**7 tasks** (Clean / Rand Success Rate) ↑						
Method	**Backbone (Type)**	Clean	Rand	block ranking rgb	hanging mug	open laptop	pick diverse bottles	rotate qrcode	stack blocks three	place shoe
π_0	π_0 (VLA)	60.71	57.86	70	14	76	49	63	77	76
				73	11	76	31	63	75	76
Spatial Forcing	π_0 (VLA)	82.00	84.43	93	**80**	67	47	**100**	87	**100**
				87	**77**	70	73	**97**	87	**100**
Ours (π_0)	π_0 (VLA)	85.43	85.29	96	63	93	70	90	90	96
				90	56	**100**	79	86	88	98
$\pi_{0.5}$	$\pi_{0.5}$ (VLA)	77.57	75.00	82	18	90	81	89	91	92
				85	17	96	71	87	76	93
Ours ($\pi_{0.5}$)	$\pi_{0.5}$ (VLA)	**92.57**	**90.14**	**100**	*70*	*96*	*86*	**100**	**96**	**100**
				97	57	96	**94**	*95*	*92*	**100**
X-WAM	Wan2.2 (WAM)	85.86	86.14	*99*	46	*96*	**91**	84	88	*97*
				95	55	*97*	*92*	83	82	*99*
Fast-WAM	Wan2.2 (WAM)	*88.57*	*89.43*	**100**	58	**98**	80	*93*	*95*	96
				97	*62*	*97*	85	89	**97**	*99*

Table 4: Real-world results. Avg. columns report success rate (SR, %) and progress score (PS) averaged over all tasks. Each task cell reports SR / PS. Higher is better. **Bold**: best in each backbone group. Rows shaded in gray are ours. All models are trained with the same number of seen samples.

	Avg.		**Tasks** (SR / PS)				
Method	SR↑	PS↑	fill pen-holder	fold towel	hang mugs	place fruits	stack bowls
π_0	34.4	43.4	**12.0** / 23.6	44.0 / 58.0	16.0 / 26.8	48.0 / 54.0	52.0 / 54.4
Spatial Forcing	31.2	37.6	8.0 / 14.8	28.0 / 34.0	20.0 / 33.2	56.0 / 62.0	44.0 / 44.0
Ours (π_0)	**44.8**	**56.0**	**12.0 / 27.8**	**48.0 / 60.0**	**28.0 / 46.4**	**64.0 / 73.6**	**72.0 / 72.0**
$\pi_{0.5}$	60.8	70.5	52.0 / 59.6	44.0 / 62.0	32.0 / 50.0	84.0 / 88.8	**92.0 / 92.0**
Ours ($\pi_{0.5}$)	**71.2**	**79.6**	**56.0 / 67.6**	**60.0 / 76.0**	**52.0 / 65.2**	**96.0 / 97.2**	**92.0 / 92.0**
X-WAM	22.4	29.2	0.0 / 0.0	8.0 / 10.0	12.0 / 30.8	68.0 / 76.4	24.0 / 28.8
Fast-WAM	30.4	38.2	4.0 / 19.4	20.0 / 24.0	28.0 / 39.2	40.0 / 47.2	60.0 / 61.2

4.5 RQ4: Which Design Choices Drive This Improvement?

Table 6 ablates SparQ4D ($\pi_{0.5}$) along three design axes: the query types supervised, the number of queries sampled per step, and the backbone layer that the query decoder attends into.

Any geometric query helps; combining them helps most. Each query type alone improves over the baseline by roughly 10 points in both settings, recovering most of the gain of the full model. The one exception is motion queries under randomization, where they are clearly the weakest. Unlike the other three types, which concern geometry observable in the current frame, motion queries ask where a point will be in a future frame, which the current observation does not fully determine. We attribute its weaker Rand result to this noisier target. Supervising all four jointly gives the best result in both settings, consistent with the signals being complementary. Most of the improvement thus comes from supervising the VLA with explicit geometric queries at all, and the unified interface allows further query types to be added on top without changing the architecture.

Table 5: Training and inference cost. Measured in the real-world setting (10k steps, batch size 64). Training VRAM is reported in GB. Params are given as trainable + frozen. Rows shaded in gray are ours. For WAMs we use gradient accumulation for two batches of 32 samples. Inference latency is measured on one NVIDIA RTX 3090 device.

Method	Backbone (Type)	Params (Trainable + Frozen)	Training VRAM	Training Time ↓ (H20 GPU-hours)	Inference Latency ↓
Ours	$\pi_{0.5}$ (VLA)	3.4B	162G	32h	146ms
Spatial Forcing	π_0 (VLA)	4.2B	144G	42h†	135ms
Fast-WAM	Wan2.2 (WAM)	12.4B	399G	101h	249ms
X-WAM	Wan2.2 (WAM)	13.0B	679G	160h	960ms

†Includes 12.5 H20 GPU-hours of offline feature precomputation with a frozen VGGT (Wang et al., 2025) teacher; training VRAM is measured with cached features and excludes the teacher. Caching these features requires about 7TB of disk at our data scale; running VGGT online instead avoids the cache but increases training time to 54 H20 GPU-hours.

A small query budget suffices. Success improves as the number of queries per step grows from 24 to 96 and plateaus beyond that point: 192 queries bring no further gain. Sparse supervision on a small fraction (about 12.5%) of image patches is therefore sufficient.

Supervising deeper layers helps. Attaching the query decoder to deeper layers improves success monotonically, with the final layer performing best. Because the decoder's gradient reaches only the layers up to its attachment point, a deeper attachment lets the geometric supervision shape more of the backbone that the action expert reads from. Sec. 4.6 examines what the resulting representations encode.

Table 6: Ablation on RoboTwin (success rate, %). Left: query configuration; right: SR. We ablate the query type (green) at the default query number 96 and cross-attention layer 18; the query number (blue) and the cross-attention layer (red) with all query types enabled. Shaded cells mark the factor being varied. **Bold**: best. All models in this table are trained for 5 epochs and tested on 7 tasks, same as Tab. 3.

Query Configuration			SR (%)	
Query Type	Query Number	Cross-Attn Layer	Clean	Random
$\pi_{0.5}$ (Baseline, no queries)			77.57	75.00
$\pi_{0.5}$ (+ depth queries only)	96	18	88.57	87.57
$\pi_{0.5}$ (+ motion queries only)	96	18	89.43	83.71
$\pi_{0.5}$ (+ hand-eye queries only)	96	18	87.57	86.00
$\pi_{0.5}$ (+ multi-view queries only)	96	18	88.00	86.57
Ours (Full)	96	18	**92.57**	**90.14**
Ours (Full)	24	18	84.86	85.43
Ours (Full)	48	18	89.14	88.14
Ours (Full)	96	18	**92.57**	**90.14**
Ours (Full)	192	18	91.43	89.00
Ours (Full)	96	6	82.29	79.14
Ours (Full)	96	12	87.14	88.00
Ours (Full)	96	18	**92.57**	**90.14**

4.6 RQ5: Does SparQ4D Make the VLA Representation Geometry-Aware?

The query decoder recovers all four query types in unseen environments. The decoder of SparQ4D ($\pi_{0.5}$) matches the ground truth for all four query types with millimeter-level errors

(Fig. 4); motion queries err most, as their targets are only partly determined by the current frame (Sec. 4.5).

Query supervision adds geometry beyond the baseline. To isolate the effect of query supervision, we add a control run alongside our full model, in which the policy ($\pi_{0.5}$) is frozen and only the SparQ4D query decoder is trained. Results show that the decoder performs far worse on all four query types with backbone frozen: action supervision alone does not make this geometry decodable, whereas query supervision does. Together with the gains concentrated on geometry-aware tasks (Sec. 4.2), this suggests that SparQ4D improves manipulation by making the geometry these tasks depend on directly available in the representation.

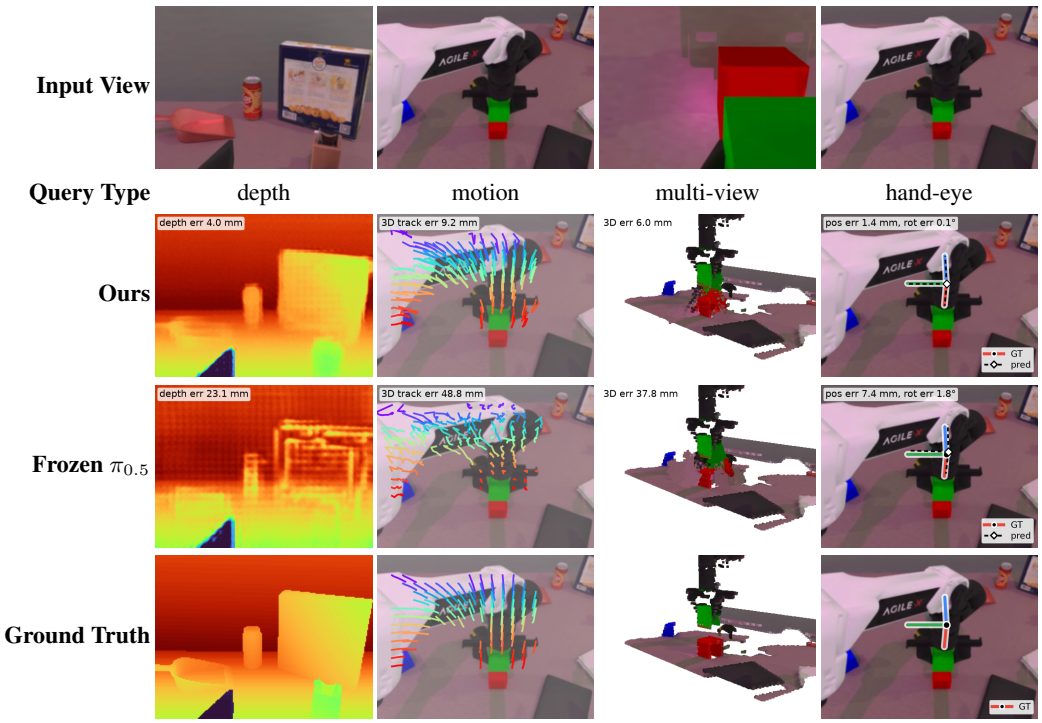


Figure 4: Probing in unseen test environments. The backbone runs once per frame, and the decoder answers non-overlapping queries tiling the image; corner labels give median errors. **Ours**: decoder trained jointly with the backbone. **Frozen** $\pi_{0.5}$: the same decoder trained on a frozen $\pi_{0.5}$ backbone. *Depth*: ground truth is quantized at $\approx$4.7 mm; blocking comes from patch-wise decoding. *Motion*: future 3D tracks projected onto the image. *Multi-view*: left-wrist pixels predicted in the head-camera frame, rendered from a virtual viewpoint over the ground-truth head-camera point cloud; errors include points occluded in that view. *Hand-eye*: head-camera pixels predicted in the gripper frame are rigidly aligned with the predicted depth, without ground truth, to recover the gripper pose (8 cm axes; dashed: prediction, solid: ground truth); the reported error is that of this pose.

5 Conclusion

In this work, we present SparQ4D, a framework that teaches robot policies geometry and motion through unified sparse queries. It casts depth, motion, multi-view, and hand–eye relations as a single question, where a selected point will be at a given time in a given frame, answered by one lightweight decoder on the policy's backbone. Sparse query sampling keeps the auxiliary cost under direct control, and since the decoder is discarded after training, the policy's architecture and inference cost remain unchanged. Across two VLA backbones in simulation and the real world, SparQ4D consistently improves manipulation, especially on tasks requiring precise spatial alignment, at a fraction of the training cost of world action models. These results suggest that sparse, explicit supervision can capture much of the benefit of dense world modeling.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods (writing training and evaluation code based on experimental designs specified by the authors) and to assist with translation. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims or proofs, propose or refine hypotheses, design research methodology or experiments, clean or reformat datasets, support qualitative data analysis, or interpret results. Additionally, we used generative AI tools to edit the paper for readability and grammar, and to format tables (including computing and verifying averaged metrics). We have reviewed all AI-assisted work: all AI-generated code was verified and tested by the authors, and all AI-edited text was checked for accuracy. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 35101–35113, June 2026.

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 Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025b. doi: 10.15607/RSS.2025.XXI.010.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, Ru Ying, Ran Zheng, and Yao Mu. Aha-wam:asynchronous horizon-adaptive world-action modeling with observation-guided context routing, 2026. URL https://arxiv.org/abs/2606.09811.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, Honghao Su, Zhiyang Dou, Kaixuan Wang, Dandan Zhang, Yunze Liu, Yan Qin, Qiwei Liang, Qiwei Wu, Zijian Lin, Wenwei Lin, Yuran Wang, Minghua He, Tianshu Wu, Ruihai Wu, Jingquan Zhou, Kai-Chong Lei, Haibao Yu, Yuanfeng Ji, Weiyang Jin, Guanyu Lin, Xiaofan Li, Qi Xiong, Renjing Xu, Zhongyu Li, Wenhao Chai, Enze Xie, Ziwei Wang, Yao Mu, Hao Dong, Wojciech Matusik, Mingyu Ding, Wenbo Ding, Ping Luo, and Masayoshi Tomizuka. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026a. URL https://arxiv.org/abs/2607.04434.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=itonej9GIV.

Xingyu Ding, Yuzhong Zhao, Chunhai Zhao, Yinghuan Shi, Chaoyang Zhao, and Yifan Zhang. Temporal forcing: 4d representation alignment for vision-language-action models, 2026. URL https://arxiv.org/abs/2608.30643.

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. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 9156–9172. Curran Associates, Inc., 2023. doi: 10.52202/075280-0403. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1d5b9233ad716a43be5c0d3023cb82d0-Paper-Conference.pdf.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 694–710. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/goyal23a.html.

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, 2026. URL https://arxiv.org/abs/2604.26694.

Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6013–6022, October 2025.

Lingyu Kong, Ruicheng Li, Ruicheng Wang, Sicheng Xu, Chengtang Yao, Jianfeng Xiang, and Jiaolong Yang. MoGe-3: Fine-detail monocular geometry estimation with self-guided sparse volumetric refinement, 2026. URL https://arxiv.org/abs/2607.17967.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long ZENG, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 132324–132345, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/d676c05a8fb93ad9231508d95567c498-Paper-Conference.pdf.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Luyao Zhang, Mingrui Yu, Zelin Gao, Nan Xue, Boyu Zhou, Xing Zhu, Mingyu Ding, Yujun Shen, and Yinghao Xu. Causal World Modeling for Robot Control. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026b. doi: 10.15607/RSS.2026.XXII.016.

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, 2026c. URL https://arxiv.org/abs/2606.14048.

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

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 785–799. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/shridhar23a.html.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/55053683268957697aa39fba6f231c68-Paper.pdf.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306, June 2025.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 10641–10662, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/2c37c5bcef24b9541550261dcd63261b-Paper-Conference.pdf.

XPolicyLab Community, Tianxing Chen, Yue Chen, Tian Nian, Zijian Cai, Guangyu Chen, Wenwei Lin, Qiwei Liang, Zanxin Chen, Peicheng Xiang, Kailun Su, Zixuan Li, Junyuan Tang, Yan Qin, Qiangyu Chen, Shaolong Zhu, Tengyue Jiang, Yiqing Wang, Xiang Li, Jiahao Zhang, Weijie Wan, Baijun Chen, Honghao Su, Kehe Ye, Shujia Liu, Kaixuan Wang, Haotian Liang, Yunze Liu, Mingleyang Li, Yuran Wang, Boyu Chen, Hongzhe Bi, Shuhe Huang, Hengkai Tan, Jisong Cai, Yao Mu, Jun Guo, Xiaofeng Wang, Zheng Zhu, Weijie Ke, Hengtao Li, Yuhang Tang, Xiaofan Li, Ganlin Yang, Zhangzheng Tu, Shuai Yang, Wenxuan Song, Pengxiang Ding, Kaidong Zhang, Yu Sun, Junliang Guo, Tong Zhang, Yixing Chen, Rongxu Cui, Zongzheng Zhang, Haoxiang Ma, Junhao Cai, Haoyu Zhang, Senqiao Yang, Jinhui Ye, Pengguang Chen, Shu Liu, Xiu Su, Wenhan Fang, Wenhao Li, Yichao Cao, Chengyao Wang, Qiang Chen, Ping Luo, and Wenbo Ding. XPolicyLab: A unified standard and open ecosystem for robot policy evaluation and deployment. *arXiv preprint arXiv:2608.09892*, 2026. URL https://arxiv.org/abs/2608.09892.

Mutian Xu, Tianbao Zhang, Tianqi Liu, Zhaoxi Chen, Xiaoguang Han, and Ziwei Liu. Kinema4d: Kinematic 4d world modeling for spatiotemporal embodied simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4495–4500, June 2026.

Lishan Yang, Wenxuan Song, Xi Wang, Pingyue Sheng, Zheng Fang, Ziyang Zhou, Junjie He, Haodong Yan, Jiayi Chen, Nan Sun, Qiao Sun, Pengwei Wang, Lingqiao Liu, Yan Wang, Yuxiang Gao, Feras Dayoub, and Haoang Li. 4d-wam: Infusing spatiotemporal awareness into world action models through trajectory fields, 2026. URL https://arxiv.org/abs/2608.08023.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William

Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 87400–87442, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d9e42b4d7163931f3689d6d6fbaa11d0-Paper-Conference.pdf.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination?, 2026. URL https://arxiv.org/abs/2603.16666.

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*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.067.

Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K. Barral, Raia Hadsell, Zoubin Ghahramani, Andrew Zisserman, Junlin Zhang, and Mehdi S. M. Sajjadi. Efficiently reconstructing dynamic scenes one d4rt at a time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7382–7392, June 2026a.

Yuhao Zhang, Wanxi Dong, Yue Shi, Yi Liang, Jingnan Gao, Qiaochu Yang, Yaxing Lyu, Zhixuan Liang, Yibin Liu, Congsheng Xu, Xianda Guo, Wei Sui, Yaohui Jin, Xiaokang Yang, Yanyan Xu, and Yao Mu. R3dp: Real-time 3d-aware policy for embodied manipulation, 2026b. URL https://arxiv.org/abs/2603.14498.

Haoyu Zhen, Qiao Sun, Hongxin Zhang, Junyan Li, Siyuan Zhou, Yilun Du, and Chuang Gan. Learning 4d embodied world models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5337–5347, October 2025.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loïc Magne, Avnish Narayan, You Liang Tan, Guanzhi Wang, Qi Wang, Jiannan Xiang, Yinzhen Xu, Seonghyeon Ye, Jan Kautz, Furong Huang, Yuke Zhu, and Linxi Fan. Flare: Robot learning with implicit world modeling. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3952–3971. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/zheng25a.html.

A Implementation Details of SparQ4D

A.1 Query Decoder

Query tokens. The RGB patch is cut from the 224×224 input image of the source camera after the backbone's image preprocessing, with $p = 14$, which equals the patch size of the SigLIP image encoder in PaliGemma. The positional encoding applies random Fourier features (Tancik et al., 2020) to each pixel coordinate, normalized to $[-1, 1]$; the random frequencies and phases are drawn at initialization and kept fixed. The learned embedding tables have three entries for the source camera, $H = 50$ entries for the time offset, and five entries for the reference frame. The five terms are summed and layer-normalized.

Decoder. The prefix tokens are linearly projected from the backbone width of 2048 to the decoder width of 1024. Each of the three decoder blocks applies pre-normalized multi-head cross-attention and a pre-normalized gated MLP (SwiGLU (Shazeer, 2020)), each with a residual connection; padding tokens in the prefix are masked out. A final layer normalization and a linear layer produce the $14 \times 14 \times 3$ outputs of each query. The decoder has 53.7M parameters.

A.2 Targets

The simulator provides depth maps for all cameras, camera intrinsics, per-frame camera extrinsics (the wrist cameras move with the arms), and gripper poses; the gripper frame is the end-effector frame of each arm. Targets are computed on the 224×224 canvas seen by the backbone, onto which images are resized with padding. Depth maps are resized with nearest-neighbor interpolation, which avoids interpolating depth across object boundaries. We set $s = 1.2\,\mathrm{m}$ and clip normalized coordinates to $[-1, 1]$. For real-world setting, these numbers are acquired through hand-eye calibration and forward kinematics.

A.3 Query Sampling

Allocation. We use $M = 96$ queries per frame, i.e., 24 per query type. The source camera is chosen uniformly among the three cameras, except for hand–eye queries, which use the head camera. A hand-eye query refers to one of the two grippers, and a multi-view query to one of the two cameras other than its source camera, each chosen uniformly at random.

Pixel importance. The importance map is the gradient magnitude of the depth map, smoothed with a 5×5 box filter, averaged over cells of 4×4 pixels, normalized per frame by its 99th percentile, and clipped to $[0, 1]$. A cell i with importance w_i is sampled with probability $\alpha/|\Omega| + (1-\alpha)\, w_i^\gamma / \sum_{j \in \Omega} w_j^\gamma$, where Ω contains the cells whose $p \times p$ patch lies inside the image, $\alpha = 0.1$ and $\gamma = 2$. The pixel is then drawn uniformly within the sampled cell.

Gripper-anchored queries. Two queries per frame are anchored at the projections of the two gripper positions into the head camera, whenever the grippers are visible. If such a query is a motion query, $(\mathbf{u}, \mathbf{v})_{t+\Delta}$ is the projection of the gripper position at time $t + \Delta$; unlike a tracker output, this projection remains valid under occlusion.

Point tracks. We run CoTracker3 (Karaev et al., 2025) offline on the video of each camera, on 50-frame windows that are initialized on a 20×20 grid every 25 frames. For a frame t, we use the most recently initialized window that contains t, which has accumulated the least drift. A motion query selects a point visible at time t, with the importance weights above, and draws $\Delta \in \{1, \ldots, 49\}$ uniformly among the offsets at which the point is visible and its patch lies inside the image.

A.4 Training

We initialize the policy from the $\pi_{0.5}$ base checkpoint and the decoder randomly, and train all parameters jointly with the hyperparameters in Tab. 7. Color augmentation is applied to all cameras, whereas the random crops and rotations that $\pi_{0.5}$ applies to the head camera are all disabled, since

they would move image content relative to the query pixels. For π_0, we use the same settings with the π_0 base checkpoint.

Table 7: **Hyperparameters of SparQ4D.**

Queries	
Queries per frame M	96 (24 per type)
Patch size p	14
Time offsets Δ	$0, \ldots, 49$
Decoder	
Blocks / width	3 / 1024
Attention heads / head dim.	8 / 128
MLP hidden dim.	4096
Parameters	53.7M
Optimization	
Loss weight λ	0.1
Batch size	64
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$)
Learning rate (peak / final)	1.25×10^{-5} / 1.25×10^{-6}
Schedule	4k-step warmup, cosine decay
Gradient clipping	1.0
EMA decay	0.99

B Additional Simulation Results

B.1 Results on All 50 Tasks

As shown in Tab. 8 and Tab. 9, we do another full run for all 50 tasks, on a step of 200k and batch size 64.

C Real-World Experimental Setup

This section describes the physical platform (App. C.1), demonstration data collection procedure (App. C.2), evaluation protocols (App. C.3), and task-specific settings (App. C.4) used in all real-world experiments.

C.1 Robot Platform

All real-world experiments are conducted on a dual-arm PiPER-X mobile manipulator platform, as shown in Fig. 5. Each arm consists of six joints and is equipped with a parallel-jaw gripper controlled by a continuous-valued gripper command. Three synchronized Intel RealSense D435i cameras—one head-mounted camera and one wrist-mounted camera on each arm—are mounted on the platform.

C.2 Data Collection

Our data collection and deployment pipelines are built upon LeRobot (Cadene et al., 2024). For each task, we collect 50 successful teleoperated demonstration trajectories. Each trajectory contains synchronized multi-view RGB observations with size 640×480, robot proprioception and joint-space action streams at 30 Hz. When required by a policy's action representation, we convert joint-space actions to end-effector-space actions using forward kinematics.

C.3 Evaluation Protocols

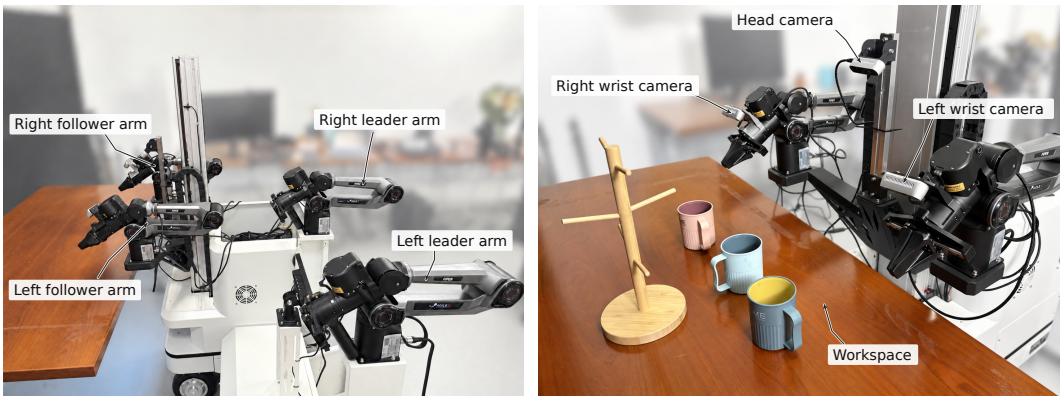
To control variation in scene initialization and enable paired comparison across methods, we construct a fixed set of 25 evaluation configurations for each task and reuse the same configurations for every evaluated method. Before every rollout, the evaluator manually restores the objects to

Table 8: Per-task success rates (%) on all 50 RoboTwin tasks with the π_0 backbone. Ours is trained in a single run for 200k steps ($\sim$2.5 epochs) and evaluated with 30 trials per task. The better result in each setting is shown in **bold**.

Task	π_0		**Ours** (π_0)	
	Clean	Random	Clean	Random
adjust_bottle	70.0	56.7	**100.0**	**100.0**
beat_block_hammer	80.0	73.3	80.0	**83.3**
blocks_ranking_rgb	80.0	83.3	**90.0**	**90.0**
blocks_ranking_size	43.3	60.0	**63.3**	**80.0**
click_alarmclock	**83.3**	73.3	66.7	**80.0**
click_bell	**66.7**	**83.3**	40.0	63.3
dump_bin_bigbin	**100.0**	90.0	90.0	**93.3**
grab_roller	93.3	96.7	**100.0**	96.7
handover_block	33.3	40.0	**80.0**	**76.7**
handover_mic	90.0	**100.0**	**96.7**	96.7
hanging_mug	20.0	20.0	**50.0**	**63.3**
lift_pot	63.3	73.3	**100.0**	**93.3**
move_can_pot	56.7	53.3	**93.3**	**76.7**
move_pillbottle_pad	73.3	66.7	**86.7**	**96.7**
move_playingcard_away	80.0	83.3	**93.3**	**93.3**
move_stapler_pad	36.7	50.0	**83.3**	**70.0**
open_laptop	73.3	70.0	**80.0**	**93.3**
open_microwave	**70.0**	**83.3**	43.3	76.7
pick_diverse_bottles	40.0	53.3	**60.0**	**66.7**
pick_dual_bottles	46.7	50.0	**93.3**	**83.3**
place_a2b_left	60.0	43.3	**83.3**	**83.3**
place_a2b_right	30.0	46.7	**73.3**	**70.0**
place_bread_basket	76.7	46.7	**90.0**	**86.7**
place_bread_skillet	63.3	60.0	**90.0**	**80.0**
place_burger_fries	93.3	93.3	**100.0**	**96.7**
place_can_basket	46.7	40.0	**80.0**	**86.7**
place_cans_plasticbox	83.3	73.3	**96.7**	**93.3**
place_container_plate	93.3	100.0	**96.7**	100.0
place_dual_shoes	63.3	50.0	**80.0**	**76.7**
place_empty_cup	100.0	96.7	100.0	**100.0**
place_fan	56.7	73.3	**86.7**	**93.3**
place_mouse_pad	43.3	40.0	**83.3**	**60.0**
place_object_basket	60.0	73.3	**86.7**	**86.7**
place_object_scale	70.0	43.3	**80.0**	**73.3**
place_object_stand	70.0	76.7	**100.0**	**100.0**
place_phone_stand	70.0	56.7	**83.3**	**86.7**
place_shoe	86.7	80.0	**100.0**	**96.7**
press_stapler	70.0	**80.0**	**73.3**	76.7
put_bottles_dustbin	63.3	83.3	**66.7**	83.3
put_object_cabinet	53.3	43.3	**93.3**	**76.7**
rotate_qrcode	66.7	60.0	**93.3**	**86.7**
scan_object	46.7	23.3	**90.0**	**76.7**
shake_bottle	96.7	90.0	**100.0**	**100.0**
shake_bottle_horizontally	**100.0**	90.0	96.7	**96.7**
stack_blocks_three	50.0	63.3	**86.7**	**83.3**
stack_blocks_two	80.0	93.3	**96.7**	**100.0**
stack_bowls_three	76.7	63.3	76.7	**80.0**
stack_bowls_two	93.3	96.7	**96.7**	96.7
stamp_seal	**60.0**	53.3	53.3	**86.7**
turn_switch	26.7	26.7	**43.3**	**63.3**
Average	67.0	66.4	**83.3**	**85.0**

Table 9: Per-task success rates (%) on all 50 RoboTwin tasks with the $\pi_{0.5}$ backbone. Ours is trained in a single run for 200k steps ($\sim$2.5 epochs) and evaluated with 30 trials per task. The better result in each setting is shown in **bold**.

Task	$\pi_{0.5}$		**Ours** ($\pi_{0.5}$)	
	Clean	Random	Clean	Random
adjust_bottle	63.3	70.0	**100.0**	**96.7**
beat_block_hammer	76.7	93.3	**83.3**	93.3
blocks_ranking_rgb	86.7	86.7	**93.3**	**90.0**
blocks_ranking_size	76.7	26.7	**80.0**	**76.7**
click_alarmclock	93.3	90.0	**96.7**	**100.0**
click_bell	96.7	66.7	**100.0**	**96.7**
dump_bin_bigbin	90.0	96.7	**96.7**	96.7
grab_roller	**100.0**	**100.0**	96.7	96.7
handover_block	83.3	56.7	**90.0**	**76.7**
handover_mic	93.3	96.7	**100.0**	96.7
hanging_mug	23.3	16.7	**46.7**	**60.0**
lift_pot	100.0	86.7	100.0	**100.0**
move_can_pot	73.3	53.3	**83.3**	**83.3**
move_pillbottle_pad	86.7	60.0	**100.0**	**93.3**
move_playingcard_away	96.7	83.3	**100.0**	**100.0**
move_stapler_pad	60.0	43.3	**63.3**	**63.3**
open_laptop	86.7	**96.7**	**96.7**	90.0
open_microwave	**70.0**	**76.7**	53.3	33.3
pick_diverse_bottles	20.0	70.0	**83.3**	**83.3**
pick_dual_bottles	20.0	63.3	**96.7**	**100.0**
place_a2b_left	83.3	83.3	**86.7**	**100.0**
place_a2b_right	83.3	83.3	**93.3**	**90.0**
place_bread_basket	83.3	63.3	**93.3**	**86.7**
place_bread_skillet	86.7	66.7	**93.3**	**83.3**
place_burger_fries	96.7	86.7	96.7	**100.0**
place_can_basket	53.3	63.3	**80.0**	**80.0**
place_cans_plasticbox	90.0	83.3	**100.0**	**100.0**
place_container_plate	90.0	93.3	**100.0**	**96.7**
place_dual_shoes	**93.3**	73.3	73.3	73.3
place_empty_cup	96.7	100.0	96.7	100.0
place_fan	86.7	86.7	**100.0**	**93.3**
place_mouse_pad	63.3	40.0	**73.3**	**76.7**
place_object_basket	60.0	76.7	**86.7**	**83.3**
place_object_scale	83.3	80.0	**93.3**	**93.3**
place_object_stand	90.0	86.7	**96.7**	**96.7**
place_phone_stand	83.3	80.0	**90.0**	**96.7**
place_shoe	96.7	93.3	**100.0**	**100.0**
press_stapler	80.0	83.3	**90.0**	**93.3**
put_bottles_dustbin	76.7	80.0	**100.0**	**93.3**
put_object_cabinet	83.3	80.0	**86.7**	**90.0**
rotate_qrcode	80.0	86.7	**96.7**	**90.0**
scan_object	80.0	66.7	**93.3**	**90.0**
shake_bottle	96.7	96.7	**100.0**	**100.0**
shake_bottle_horizontally	100.0	100.0	100.0	100.0
stack_blocks_three	83.3	76.7	**93.3**	**86.7**
stack_blocks_two	**100.0**	100.0	96.7	100.0
stack_bowls_three	76.7	70.0	**93.3**	**76.7**
stack_bowls_two	93.3	**96.7**	**100.0**	93.3
stamp_seal	73.3	53.3	**86.7**	**100.0**
turn_switch	56.7	53.3	**83.3**	**66.7**
Average	79.9	76.3	**90.7**	**89.1**



(a) Robot setup (b) Scene and cameras

Figure 5: Dual-arm PiPER-X mobile manipulator platform.

the corresponding reference configuration using a visual layout overlay, following the layout-replay protocol of RoboDojo (Chen et al., 2026a).

The evaluation configurations span diverse object positions, orientations, and relative arrangements within the tabletop workspace visible to the head camera. These variations require policies to adapt their grasps and manipulation motions to different initial layouts, including changes in towel orientation, container placement, and the spatial arrangements of mugs and bowls. Fig. 6 shows representative configurations for all five tasks.

We report success rate (SR) and a task-specific progress score (PS). Each trial is assigned a score in $[0, 100]$ according to the task-specific rubrics in Tab. 10. A trial is successful if and only if it receives 100 points. SR is the proportion of successful trials, and PS is the mean per-trial score across the 25 evaluation configurations. Both metrics are computed under the same fixed set of initial configurations for all compared methods.

All methods are deployed using synchronous chunked inference. At each policy query, the model receives the latest observation and predicts an action chunk, which is executed open-loop in its entirety. The next policy query then uses the latest available observation. We use an action horizon of $H = 50$ for the π-series VLA policies and $H = 32$ for WAM-based policies, at the control frequency of 30 Hz.

C.4 Real-World Task Suite

We use five real-world manipulation tasks, spanning complementary sources of difficulty: rigid-object pick-and-place, repeated rigid-object manipulation under a stacking constraint, long-horizon bimanual insertion, deformable-object folding, and precision bimanual hanging, as shown in Fig. 7. Three of the five tasks (Stack Bowls, Fill Pen Holder, and Hang Mugs) follow the corresponding task definitions from the RoboDojo real-world benchmark (Chen et al., 2026a); the remaining two (Place Fruits and Fold Towel) are defined in-house following the same documentation format. For each task, we specify the language prompt, a brief task description, and a graded scoring rubric listed in Tab. 10.

Fill Pen Holder A pen holder and four pens, consisting of two flat highlighters and two slim gel pens, are placed on the table. The robot must pick up the holder and insert all four pens into it. The holder is initialized in the left part of the head camera's field of view, while the pens are initialized in the right part with randomized positions and orientations. The prompt is *Pick up the pen holder and place all the pens into it.*

Fold Towel A flattened towel is placed on the table, and the robot must perform two sequential folds to form a compact rectangular shape using bimanual manipulation. The required procedure consists of two steps. First, the right arm holds down the right side of the towel while the left arm

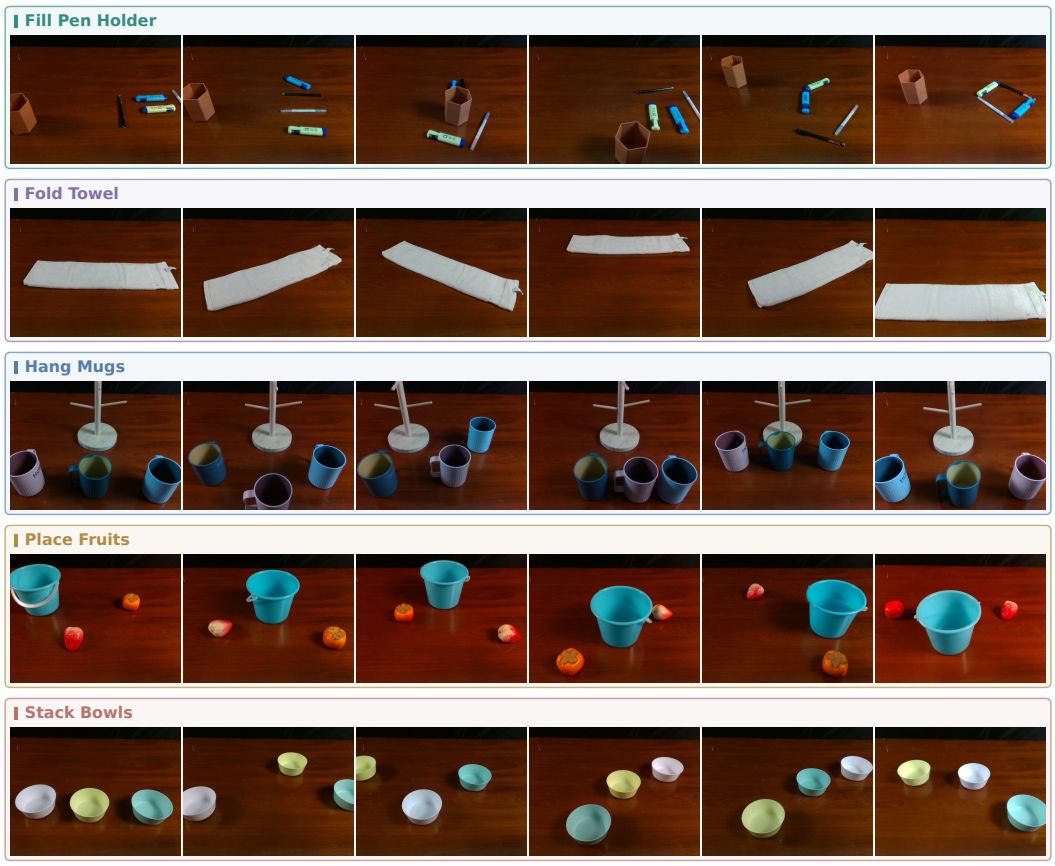


Figure 6: **Diverse initial layouts for real-world evaluation.** Each row shows six example evaluation layouts for one task, illustrating varied object positions, orientations, and relative arrangements within the head camera's view.

grasps its lower-left portion and folds it in half. Second, the left arm remains raised while the right arm lifts slightly and opens its gripper, then grasps the middle of the towel's right side and folds it in half again. The prompt is *Fold the towel.*

Hang Mugs Three mugs and a mug rack are placed on the table, and the robot must pick up the mugs and hang them securely on the rack. The prompt is *Hang the mugs on the mug rack.*

Place Fruits A strawberry and a persimmon are placed on the table, and the robot must place both fruits into a bucket. The prompt is *Place both fruits into the bucket.*

Stack Bowls Three bowls are placed on the table, and the robot must manipulate them one at a time to form a single vertical stack at the center of the workspace. The bowls are initialized with randomized left-to-right ordering and positions within the head camera's field of view. The prompt is *Stack the bowls on the table.*

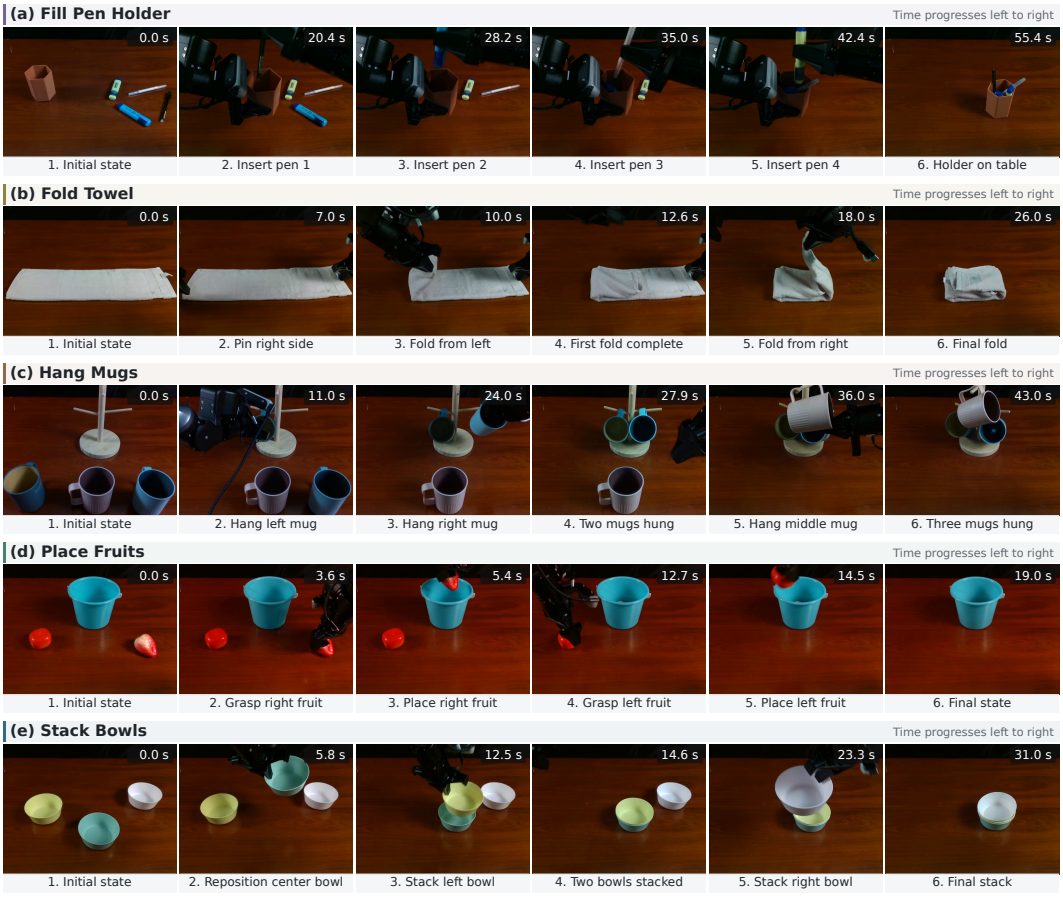


Figure 7: The five real-world tasks. For each task we show representative keyframes from a successful rollout.

Task	Score	Condition
Fill Pen Holder	0	No pen is correctly inserted into the pen holder.
	10	One pen is correctly inserted tip-up into the upright pen holder.
	25	Two pens are correctly inserted into the pen holder.
	40	Three pens are correctly inserted into the pen holder.
	90	All four pens are inserted into the upright pen holder, but the robot has not returned home.
	100	All four pens are inserted, the pen holder is upright, and the robot has returned home.
Fold Towel	0	The towel does not form a rectangular shape, or the two-step folding procedure is not followed.
	50	The two-step folding procedure is followed and the towel is roughly rectangular, but one or more sides are visibly misaligned.
	100	The two-step folding procedure is followed and the towel forms a clean square or rectangular fold with the sides properly aligned.
Hang Mugs	0	No mug is hung on the mug rack.
	20	One mug is hung on the mug rack.
	50	Two mugs are hung on the mug rack.
	100	Three mugs are hung on the mug rack.
Place Fruits	0	No fruit is placed into the bucket.
	30	One fruit is placed into the bucket.
	100	Both fruits are placed into the bucket.
Stack Bowls	0	Fewer than two bowls are stacked in the center.
	30	Two bowls are stacked in the center.
	100	All three bowls are stacked in the center.

Table 10: Per-trial progress-scoring rubrics for real-world tasks (0–100 points).