SimVLA: Zero-Shot Sim-to-Real VLA Learning for Mobile Manipulation

Kyoungin Baik[1] **and Youngwoon Lee**[2]
[1]Yonsei University , [2]Seoul National University
dennybaik@yonsei.ac.kr, youngwoon@snu.ac.kr

Project page: https://kyounginbaik.github.io/simvla/

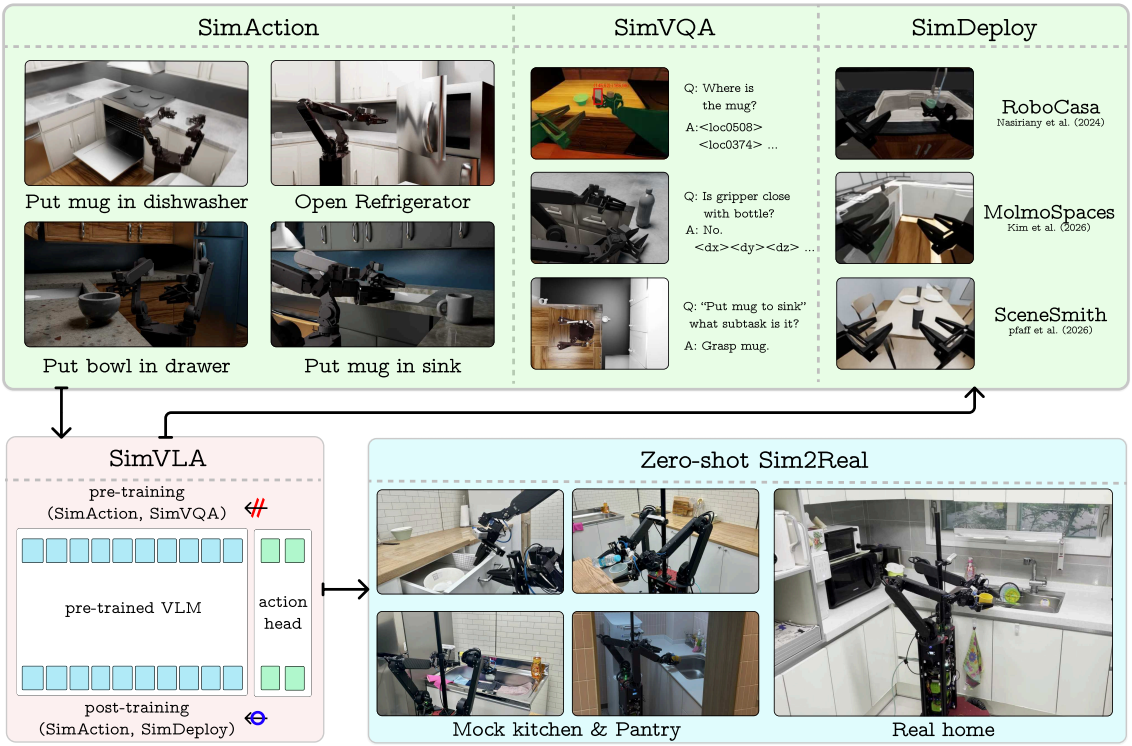


Figure 1. SimVLA: Zero-shot sim-to-real framework for mobile manipulation.

Large-scale, diverse datasets have driven the success of LLMs and VLMs. But VLAs for robotics remain limited by the cost and complexity of real-world data collection. While simulation offers a scalable alternative, its potential for sim-to-real VLA learning in mobile manipulation remains largely underexplored. We introduce **SimVLA**, an end-to-end framework that trains VLAs *entirely on synthetic simulation data* without teleoperation for mobile manipulation. SimVLA is first pre-trained on two complementary simulation-derived datasets: SimAction, a large-scale robot action dataset spanning 35 diverse mobile manipulation tasks, generated by composing atomic skills, and SimVQA, which leverages privileged simulator state to provide spatial, geometric, and subtask-level visual-language supervision. We further post-train SimVLA on a mixture of SimAction and SimDeploy, a dataset collected from policy rollouts across diverse *simulated* environments. We evaluate SimVLA on tasks including restocking, pouring, and cleaning, and show zero-shot transfer to real-world mobile manipulation, including real home environments. SimVLA outperforms policies trained on 50 in-domain real-world demonstrations, suggesting that simulation can enable scalable sim-to-real mobile manipulation. We further demonstrate the value of multiple complementary forms of supervision for effectively leveraging simulation in VLA training.

1. Introduction

Large Language Models (LLMs) and Vision-Language Models (VLMs) have benefited from large-scale, heterogeneous datasets that provide broad coverage over tasks, objects, and environments (Radford et al., 2021; Chen et al., 2022; Alayrac et al., 2022). Recent Vision-Language-Action (VLA) models extend this data-driven paradigm to robotics by conditioning policies on visual observations and language instructions while predicting robot actions (Zitkovich et al., 2023; Octo Model Team et al., 2024; Black et al., 2024, 2025; Kim et al., 2025; NVIDIA et al., 2025). However, unlike web-scale text-image data, robot action data are expensive to collect, embodiment-specific, and limited in task and environmental diversity. This data bottleneck is especially severe for long-horizon mobile manipulation.

Mobile manipulation amplifies the challenges of VLA training in several ways. First, the setting is partially observable (Jauhri et al., 2024; Li* et al., 2025): a robot may need to act beyond its current field of view. Second, mobility increases the dimensionality of the action space and tightly couples manipulation with navigation (Honerkamp et al., 2023; Yang et al., 2024; Wu et al., 2025; Shan et al., 2026). For example, a poor base position can cause task failure even when the manipulation policy is otherwise capable. As a result, learning VLAs for mobile manipulation requires broad coverage over scene layouts, viewpoints, and base–arm coordination. Collecting such coverage through real-world teleoperation is difficult (Fu et al., 2024), motivating simulation as a scalable data source if the learned policies can transfer to real environments. Recent work (Deshpande et al., 2026) has taken initial steps toward mobile manipulation, but real-world evaluation remains limited to a single door-opening task and limited performance, highlighting the need for a more effective pipeline.

We introduce **SimVLA**, a zero-shot sim-to-real framework for training VLA policies on long-horizon mobile manipulation tasks. Unlike prior simulation-based data pipelines that primarily generate or augment action trajectories, SimVLA uses simulation as a source of three complementary forms of supervision: action trajectories, privileged visual-language supervision, and deployment-oriented policy rollouts, as illustrated in Figure 1. First, **SimAction** provides large-scale, diverse synthetic robot action data spanning 35 mobile manipulation tasks in procedurally generated kitchen environments, without teleoperation. SimAction generates task trajectories by composing atomic skill APIs, allowing the same task structure to be instantiated across diverse layouts, objects, and initial configurations. Second, **SimVQA** leverages simulation ground-truth to provide spatial, geometric, and subtask-level supervision for VLA co-training. Third, **SimDeploy** consists of policy rollouts in diverse simulated environments (Nasiriany et al., 2024; Kim et al., 2026; Pfaff et al., 2026), exposing the model to deployment-oriented trajectories with broader scene layouts. Together, these data sources allow SimVLA to exploit simulation beyond action generation alone.

SimVLA is trained in two stages. The model is pre-trained on SimAction for action supervision and SimVQA for visual-language supervision. We then roll out the policy in diverse simulated environments to collect SimDeploy, and post-train SimVLA on a mixture of SimAction and SimDeploy, which we find critical for generalization to unseen real-world environments.

We evaluate SimVLA on long-horizon mobile-manipulation tasks in the real world, including mock kitchens, a pantry, and a real home. Without any real-world data for training, SimVLA transfers to unseen real-world environments. It outperforms a $\pi_{0.5}$ policy fine-tuned on 50 real-world demonstrations in the demonstration environment. It also maintains robust performance in a pantry and a real home, where both baselines degrade sharply. These results suggest that combining action, visual-language, and deployment-oriented supervision in simulation can enable zero-shot sim-to-real mobile manipulation.

Table 1. Comparison of simulation data generation methods. "V-L sup." indicates visual-language supervision beyond action trajectories, such as VQA-style annotations. "Policy rollouts" denotes data collected by rolling out a learned policy in simulation.

Dataset	# Task	Mobility	V-L sup.	Data generation	Zero-shot Sim-to-Real
MimicGen (Mandlekar et al., 2023)	18	◯	✗	Teleoperation & Augmentation	✗
RoboGen (Wang et al., 2024)	106	◯	✗	Motion planning & Optimization & RL	✗
SkillMimicGen (Garrett et al., 2025)	18	✗	✗	Teleoperation & Augmentation	✗
DexMimicGen (Jiang et al., 2025)	9	◯	✗	Teleoperation & Augmentation	◯
GraspVLA (Deng et al., 2025)	1	✗	◯	Motion planning	◯
RoboTwin 2.0 (Chen et al., 2026)	50	✗	✗	Motion planning	◯
InternData-A1 (Tian et al., 2026)	70	✗	✗	Motion planning	◯
MoMaGen (Li* et al., 2025)	4	◯	✗	Teleoperation & Augmentation	✗
RoboCasa365 (Nasiriany et al., 2026)	365	◯	✗	Teleoperation & Augmentation	✗
MolmoB0T (Deshpande et al., 2026)	8	◯	✗	Motion planning	◯
SimVLA (ours)	35	◯	◯	Motion planning & Policy rollouts	◯

2. Related Work

Data for Training VLAs. VLA models rely on large-scale, heterogeneous datasets that pair observations, language instructions, and actions producing generalist robot policies (Collaboration et al., 2023; Black et al., 2024, 2025; NVIDIA et al., 2025). However, data remains a central bottleneck for VLAs. Robot action data are primarily collected through teleoperation, making it expensive to scale.

Sim-to-Real. Simulation is widely used for robot learning due to its scalable data generation. However, zero-shot sim-to-real remains challenging because of the sim-to-real gap (Li* et al., 2025; Maddukuri et al., 2025; Nasiriany et al., 2026). To tackle this, sim-to-real methods often rely on two complementary strategies: system identification (Lin et al., 2025b; Sobanbabu et al., 2025) to reduce dynamics mismatch and domain randomization (Singh et al., 2024; Chen et al., 2026) to expose policies to a broad range of physical and visual conditions. We combine hybrid system identification, which calibrates both joint-space and task-space behavior, with broad domain randomization.

Synthetic Data Generation. Synthetic robot data is often generated by augmenting a small set of source demonstrations in simulation (Mandlekar et al., 2023; Garrett et al., 2025; Jiang et al., 2025; Li* et al., 2025). While this improves data quantity, the resulting distributions remain constrained by the source demonstrations. For example, demonstrations for opening a cabinet by pulling may not transfer to cabinets that require different articulated motions, such as rotation. Motion-planning-based approaches address this by generating synthetic data without relying on teleoperation, and recent work has shown strong sim-to-real transfer (Deng et al., 2025; Tian et al., 2026; Deshpande et al., 2026). However, existing approaches largely treat simulation only as a source of action data.

SimVLA treats simulation as a source of multiple complementary forms of supervision. SimAction provides action trajectories generated by composing atomic skill APIs. SimVQA provides spatial, geometric, and subtask-level visual-language supervision derived from privileged simulator state. SimDeploy generates rollout trajectories across diverse simulated environments, exposing the model to policy-induced states. Thus, unlike prior approaches that use simulation mainly to scale action data, SimVLA uses simulation to provide action, visual-language, and deployment-oriented supervision for zero-shot sim-to-real VLA learning.

Mobile Manipulation. Compared to stationary, mobile manipulation remains relatively underexplored despite its importance for real-world tasks. Tasks such as cleaning, restocking, and tidying require not only contact-rich manipulation, but also coordinated movement to reach suitable viewpoints and interaction poses. This makes mobile manipulation fundamentally harder than tabletop manipulation, as policy behavior depends jointly on navigation, base–arm coordination, and partial observabil-

ity (Honerkamp et al., 2023; Lin et al., 2025a). Prior work often simplifies mobile manipulation by decoupling navigation and manipulation (Yang et al., 2025), replaying base trajectories (Nasiriany et al., 2026), or assuming favorable initial poses (Chai et al., 2026). While these choices reduce control complexity, they limit the ability to learn coordinated base–arm behaviors (Yang et al., 2024). Recent work has generated synthetic data for mobile manipulation (Deshpande et al., 2026), but real-world deployment remains limited in both task diversity and performance.

In this work, SimVLA targets end-to-end VLA learning under coupled base–arm control across 35 diverse practical tasks and evaluates zero-shot transfer to three unseen real-world environments.

3. SimVLA: Synthetic Data Generation and VLA Training

SimVLA is a zero-shot sim-to-real framework for training VLA policies *solely from synthetic simulation data*. In this work, we focus on kitchen environments as a representative domain for mobile manipulation tasks. The framework first constructs procedurally generated kitchen scenes (Section 3.1). From these scenes, SimVLA derives three complementary data sources: **SimAction**, a large-scale robot action dataset generated through skill composition and motion planning (Section 3.2); **SimVQA**, a visual question-answering dataset from privileged simulator state to provide spatial, geometric, and subtask-level supervision (Section 3.3); and **SimDeploy**, a deployment-oriented dataset by rolling out the learned policy in diverse simulated environments (Section 3.4). The SimVLA model is first pre-trained on SimAction and SimVQA, and then post-trained on a mixture of SimAction and SimDeploy (Section 3.5).

3.1. Procedural Kitchen Scenes

Large-scale action data requires diverse environments, objects, and initial configurations. We use Scene Synthesizer (Eppner et al., 2024) to procedurally generate kitchen layouts, and simulate all scenes in Isaac Lab (Mittal et al., 2025). We instantiate five common kitchen types: island, L-shaped, peninsula, single-wall, and U-shaped layouts. Within each layout type, we randomize furniture heights, furniture and object arrangements, cabinet configurations, and materials. For each object category, we include diverse shape variations from Objaverse (Deitke et al., 2023) to promote generalization across object geometries. Additional scene and object details are provided in Section A.

3.2. SimAction: Synthetic Action Data

SimAction is a large-scale synthetic action dataset for mobile manipulation. Each trajectory contains observations, a language instruction, and continuous robot actions (end-effector poses of two arms and base velocity) for executing long-horizon tasks in procedurally generated kitchen scenes. SimAction comprises approximately 200K trajectories (2,778 hours of robot data) spanning 35 mobile manipulation tasks across 100 procedurally generated houses. The dataset is generated without teleoperation by composing atomic skills into task programs and converting their subtask goals into collision-free robot trajectories through motion planning.

Task generation with skill APIs. Given a generated kitchen scene, a scripted skill API generates subtask-level goal states for robot trajectory generation. The API provides skills for grasping, placing, pouring, boiling, opening, closing, serving, navigating, and searching, which can be composed into reusable long-horizon task specifications. These specifications can then be instantiated across robots, environments, and object configurations to automatically generate demonstrations at scale. For example, put mug in sink decomposes into searching for and navigating to the mug, grasping it, searching for and navigating to the sink, and placing the mug in the sink.

Searching is essential in mobile manipulation, which is inherently partially observable: the robot needs to locate task-relevant objects before interacting with them. The search skill API enables the policy to actively search for task-relevant objects using only visual input at inference time. The detailed logic of the search skill API is described in Section B.

We automatically generate grasp poses using BODex (Chen et al., 2025a), while additionally providing a GUI for optional user-defined grasp preferences. The same GUI also allows users to define new tasks by specifying a sequence of atomic skills, enabling scalable task generation without writing low-level control code. A figure of the GUI is provided in Section B.

Trajectory generation with motion planning. Before generating trajectories, we perform hybrid system identification to reduce dynamics mismatch between simulation and the real robot. Unlike prior work, which often identifies parameters by matching joint-space trajectories (Lin et al., 2025b), we additionally constrain end-effector tracking since our policy operates in end-effector space. The full procedure is described in Algorithm 1.

With the identified parameters, we instantiate parallel environments and generate SimAction trajectories by executing motion plans. Given subtask goal states from the skill API, we use cuRobo (Sundaralingam et al., 2023) to compute collision-free trajectories. To support embodiment diversity, we generate for three robot embodiments: Anubis (Kang et al., 2025), RB-Y1, and ROBOTIS AI Worker.

3.3. SimVQA: Simulation-Derived Visual-Language Supervision

VLAs require visual-language backbones that understand spatial relations, object geometry, subtask progress, and robot-centric coordinates. However, existing sim-to-real VLA pipelines primarily use simulation to generate robot action trajectories (Tian et al., 2026; Deshpande et al., 2026), leaving much of the privileged information available in simulation unused. SimVQA addresses this gap by converting simulator ground truth into visual question-answering supervision.

For each simulated scene and trajectory, we automatically generate question-answer pairs about object locations, spatial relations, geometric properties, robot state, and subtask progress, yielding approximately 1M SimVQA samples. This provides an intermediate supervision layer between perception and action, encouraging the VLM backbone to encode information relevant for downstream control. Obtaining such spatially grounded annotations from real-world demonstrations would require expensive post-processing (Chen et al., 2025b). In simulation, the same annotations are available from ground-truth state at no additional cost. All VQA templates are provided in Section F, Table 6.

3.4. SimDeploy: Deployment-Oriented Rollout Data

Although SimAction provides scalable robot action data, scripted motion-planning trajectories do not fully capture the states encountered by a learned policy at deployment time. Small prediction errors can lead to changed viewpoints, imperfect base poses, and missed grasps. To expose the model to such policy-induced states, we construct SimDeploy by rolling out the SimVLA policy zero-shot across diverse simulated environments, e.g., RoboCasa (Nasiriany et al., 2024), MolmoSpaces (Kim et al., 2026), and SceneSmith (Pfaff et al., 2026), collecting 22K trajectories (460 hours). Unlike SimAction, which is generated from scripted skill compositions and motion planning, SimDeploy captures the distribution shift induced by the learned policy itself, including diverse viewpoints and scene layouts. We find that post-training on SimDeploy improves robustness and generalization to unseen real-world environments.

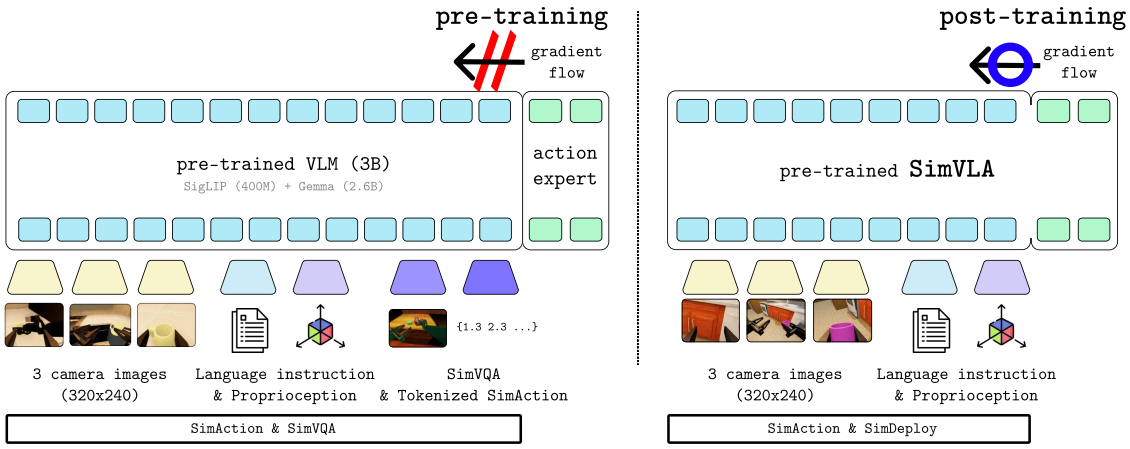


Figure 2. Overview of SimVLA Training. During pre-training, SimVLA jointly optimizes continuous action prediction, tokenized action prediction, and SimVQA supervision. During post-training, SimVLA is trained on a mixture of SimAction and deployment trajectories from SimDeploy.

3.5. SimVLA: Two-Stage VLA Training

SimVLA follows a two-stage training pipeline built on a pre-trained VLM (Beyer et al., 2024) and a flow-matching action expert, following the model structure of $\pi_{0.5}$ (Black et al., 2025). During pre-training, SimVLA jointly optimizes three simulation-derived objectives in a single forward pass: flow matching on continuous actions from SimAction, autoregressive prediction of tokenized actions (Pertsch et al., 2025) from SimAction, and autoregressive prediction of SimVQA answers. This stage aligns the VLM backbone with robot action while preserving its visual-language representation. To prevent unstable gradients from the randomly initialized action head from degrading the pre-trained VLM backbone, we stop gradients from the action head into the backbone during this stage (Driess et al., 2025).

During post-training, SimVLA optimizes the flow-matching objective on a mixture of SimAction and SimDeploy. Since the action head has been warmed up during pre-training, we allow gradients to update the VLM backbone in this stage. We deliberately select $\pi_{0.5}$-style architecture so that the experiments isolate the effect of the proposed simulation data and supervision. Detailed training objectives and configurations are provided in Section G.

4. Experiments

4.1. Simulation Experiments

We compare SimVLA with MolmoB0T (Deshpande et al., 2026) on RB-Y1 in procedurally generated unseen SimVLA kitchens and MolmoSpaces (Kim et al., 2026), both evaluated zero-shot. We additionally evaluate eight long-horizon tasks using the Anubis robot (Kang et al., 2025) (Table 12).

4.2. Real-World Experiments

We evaluate SimVLA on five long-horizon mobile manipulation tasks: *put mug / bowl / bottle in sink*, *put bowl in drawer*, and *pour bottle to mug* (Figure 3). These tasks cover distinct challenges, including navigation to receptacles, articulated-object interaction, base–arm coordination, and bimanual pouring. All real-world experiments use Anubis, a bimanual mobile manipulator with two 6-DoF arms, parallel grippers, and a 3-wheel omnidirectional base.



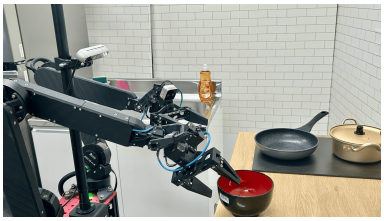
(a) Put mug in sink



(b) Put bowl in drawer



(c) Pour bottle to mug



(d) Put bowl in sink



(e) Put bottle in sink

Figure 3. Five long-horizon mobile manipulation tasks used for real-world evaluation.

We evaluate in three real-world environments: a mock kitchen, a pantry, and a real home (Figure 4). In the mock kitchen, we consider three conditions: In-Distribution (ID), where data were collected for baseline models; OOD-pos, where objects and furniture are repositioned; and OOD-vis, where backgrounds and lighting are randomized (Figure 4).

For baseline methods, we fine-tune $\pi_{0.5}$ on 50 or 200 mock kitchen data per task. All methods share the same observation/action spaces, prompts, and trial protocol. We report task progress, i.e., the percentage of predefined subgoals completed. Each task–policy pair is evaluated over 20 trials in the mock kitchen and pantry and 10 trials in the real home, for a total of 2,120 rollouts. Detailed definitions of subgoals and task progress, along with binary success rates, are provided in Section H.



(a) Mock kitchen



(b) Pantry



(c) Real home



(d) In-Distribution



(e) OOD-pos



(f) OOD-vis

Figure 4. Real-world evaluation setup. **Top:** three evaluation environments: (a) mock kitchen, (b) pantry, and (c) real home. **Bottom:** mock kitchen evaluation conditions: (d) In-Distribution, (e) OOD-pos, and (f) OOD-vis.

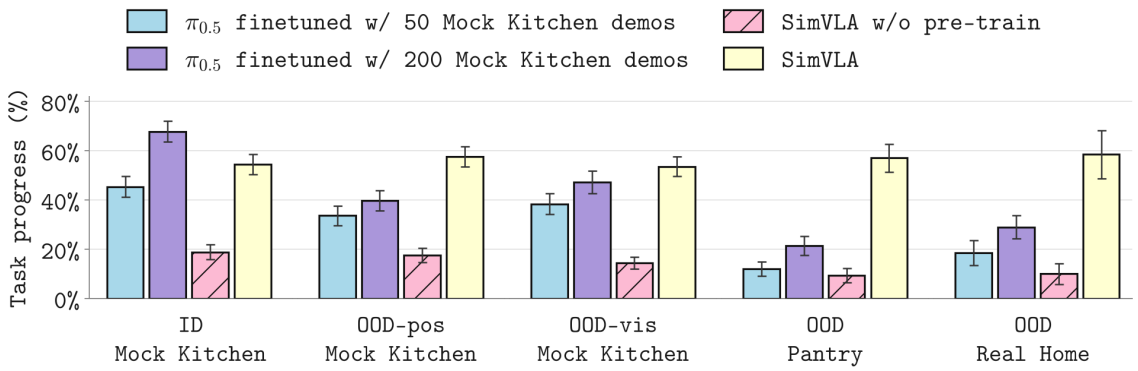


Figure 5. Averaged task progress in real-world environments. Mock kitchen includes all five tasks; pantry includes *put mug / bowl / bottle in sink*; real home includes *put mug in sink / pour bottle to mug*. SimVLA maintains robust performance in OOD environments, while the fine-tuned policy degrades substantially. Removing pre-training leads to poor performance, demonstrating its importance for robust generalization.

4.3. Can SimVLA Match Policies Trained with In-Domain Demonstrations?

Yes, in our evaluated settings. We compare SimVLA to $\pi_{0.5}$ fine-tuned on 50 or 200 in-domain real-world demonstrations per task. SimVLA is trained entirely in simulation. As shown in Figure 5, SimVLA achieves 54.4% average task progress, outperforming the $\pi_{0.5}$ policy fine-tuned on 50 in-domain real-world demonstrations in our evaluated settings. Under OOD-pos and OOD-vis, in-domain baselines degrade substantially, and increasing real-world demonstrations from 50 to 200 gives limited gain. This suggests that additional in-domain data alone may not provide sufficient coverage for robust mobile manipulation in OOD settings, while SimVLA benefits from broader synthetic variation.

4.4. Does SimVLA Need Pre-Training?

Yes. We compare SimVLA with a variant that skips pre-training and is trained only with the post-training objective on the same SimAction–SimDeploy mixture. As shown in Figure 5, removing pre-training substantially degrades performance. This finding is consistent with prior work that leverages large-scale real-world robot data and multimodal web data for pre-training. In contrast, SimVLA obtains its pre-training data, SimAction/SimVQA, entirely from simulation.

Although $\pi_{0.5}$ (Black et al., 2025) has demonstrated strong generalization to unseen environments, adapting the pre-trained checkpoint to a new task or robot embodiment typically requires additional data collection and fine-tuning. Because these data are collected in a particular environment, the fine-tuned policy can exhibit substantial performance degradation under environment changes. In our experiments, the $\pi_{0.5}$ model fine-tuned with 200 real-world demonstrations achieves 67.8% task progress in mock kitchen-ID, but only 21.3% and 29% in pantry and real home, respectively. In contrast, SimVLA maintains consistent performance across all three real-world environments.

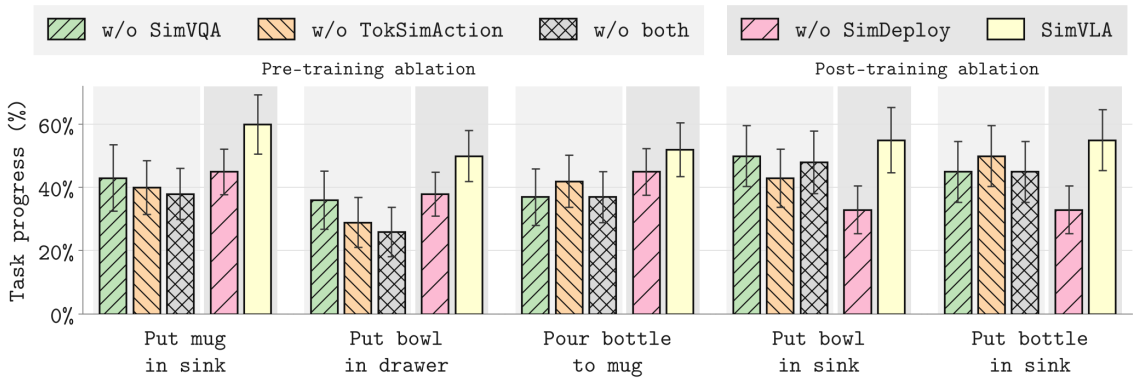


Figure 6. Averaged task progress across tasks in mock kitchen-ID. Removing either SimVQA or Tokenized SimAction degrades performance across all tasks. SimDeploy yields task-dependent gains in zero-shot deployment performance.

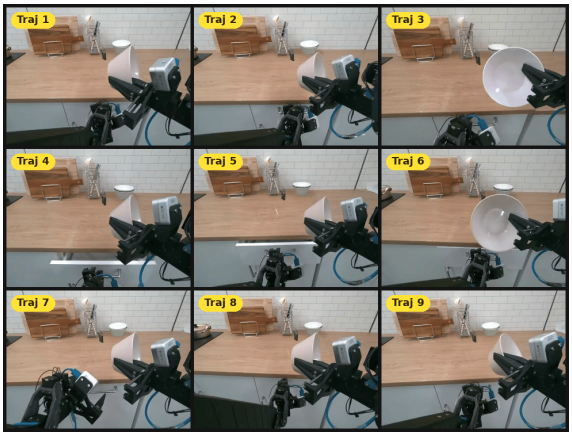
4.5. Pre-training Data Ablation

We compare SimVLA with variants that remove SimVQA, TokSimAction, or both during pre-training to evaluate their contributions. As shown in Figure 6, removing either component consistently degrades performance across all five tasks, with the full model achieving the best results. Qualitatively, the model pre-trained without SimVQA often fails due to imprecise grasping, suggesting that SimVQA provides valuable spatial supervision for downstream manipulation.

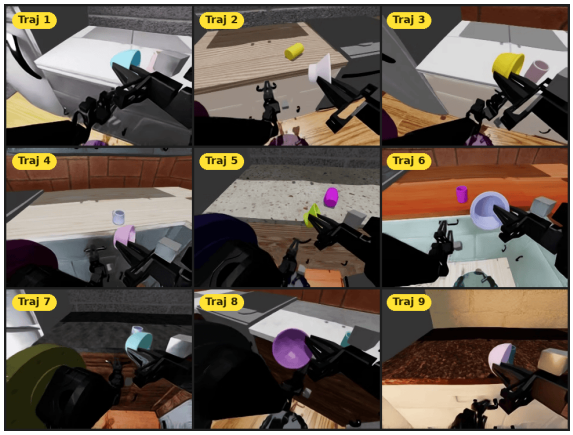
To further investigate this effect, we evaluate SimVLA on 1,369 held-out SimVQA pairs. As shown in Table 11, the full model achieves a median 3D position error of only $0.06\,\mathrm{m}$ and a cross-view IoU of 0.61, substantially outperforming the ablated variants. These results indicate that SimVQA pre-training provides the model with accurate spatial reasoning capabilities beyond action prediction.

4.6. Post-training Data Ablation

We compare SimVLA against a variant without SimDeploy during post-training in Figure 6. SimDeploy substantially improves *put mug / bowl / bottle in sink* and *put bowl in drawer* as it adds deployment trajectories with diverse layouts, viewpoints, and retry behaviors. However, it gives relatively smaller gains on *pour bottle to mug*. This is consistent with the variation required by each task: pouring mainly requires object-level generalization, already covered by SimAction randomization, whereas sink and drawer tasks require broader scene-level variation.



(a) Real-world teleoperation demonstrations



(b) Simulation demonstrations

Figure 7. Sampled drawer-grasping frames from real-world teleoperation and simulation. Teleoperation trajectories are visually homogeneous, while simulation exhibits greater diversity.

4.7. What Makes Mobile Manipulation Hard in OOD Situations?

Limited trajectory coverage. Teleoperated demonstrations often cover narrow viewpoints and approach trajectories, especially for precise interactions such as drawer-handle grasping (Figure 7a). With action chunking, small errors can push the policy into OOD states; for example, slight over-rotation in the *put bowl in drawer* can misalign the gripper with the handle and cause grasp failure. As a result, in mock kitchen OOD-pos (Figure 4e), policies trained on real-world demonstrations mostly failed to grasp the handle. In contrast, SimVLA generates more diverse viewpoints and spatial configurations through subgoal domain randomization (Figure 7b), resulting in more robust performance. This suggests that limited trajectory coverage is a key bottleneck for OOD generalization, and that large-scale simulation can mitigate this limitation.

4.8. How does SimVLA compare to existing zero-shot sim-to-real mobile manipulation baselines?

SimVLA enables more generalizable mobile manipulation. We compare SimVLA with MolmoB0T using the RB-Y1 robot across five held-out SimVLA kitchens and five MolmoSpaces environments. We evaluate three picking and two pick-and-place tasks with 200 trials per task. While MolmoB0T's performance degrades substantially in SimVLA kitchens, SimVLA maintains stable performance on MolmoSpaces, demonstrating stronger zero-shot transfer.

We also evaluate mobile manipulation when targets are not visible from the initial viewpoint. MolmoB0T achieves 0% success across all tasks, whereas SimVLA succeeds through learned active vision behaviors. Together, these results demonstrate that SimVLA achieves stronger zero-shot mobile manipulation, generalizing to unseen environments and locating targets under partial observability.

Table 2. Zero-shot mobile manipulation. Success rate (%); *Pick* {*mug, bottle, apple*}, *PnP* {*apple in bowl, mug next to bowl*}. SimVLA is post-trained without MolmoSpaces SimDeploy. SimVLA remains robust under partial observability. None of the environments are included in the training dataset.

	MolmoB0T	SimVLA
Pick		
MolmoSpaces	36.3 ± 2.8	$\mathbf{71.0 \pm 2.9}$
SimVLA kitchens	9.7 ± 1.7	$\mathbf{91.2 \pm 2.8}$
PnP		
MolmoSpaces	1.0 ± 0.7	$\mathbf{49.0 \pm 3.4}$
SimVLA kitchens	0.5 ± 0.5	$\mathbf{79.5 \pm 3.1}$
Partial observation		
MolmoSpaces	0.0 ± 0.0	$\mathbf{51.8 \pm 3.1}$
SimVLA kitchens	0.0 ± 0.0	$\mathbf{81.0 \pm 2.9}$

5. Conclusion

We presented SimVLA, a zero-shot sim-to-real framework that trains VLAs entirely from simulation data for mobile manipulation. By combining procedural kitchen scene generation, hybrid system identification, domain randomization, and SimVQA supervision, SimVLA produces large-scale, diverse synthetic training data without teleoperation. SimDeploy data collected through policy rollouts across simulation environments further reduces the distribution gap by exposing the policy to scene-level diversity and policy-induced states beyond procedural demonstrations. Experiments across five tasks and three real-world environments show that SimVLA, trained purely in simulation, outperforms policies trained on 50 in-domain real-world data in our evaluation settings. SimVLA also transfers to practical settings, including a pantry and a real home, without environment-specific data. Comparison with a current sim-to-real VLA baseline demonstrates SimVLA's generalization across mobile manipulation settings. Furthermore, SimVLA demonstrates that simulation can serve not only as a source of action trajectories, but also as a source of visual-language supervision and deployment-oriented rollout data for zero-shot sim-to-real VLA learning in mobile manipulation.

6. Limitations & Future Work

Although SimVLA demonstrates zero-shot sim-to-real transfer for mobile manipulation, it has several limitations. First, we do not isolate the contributions of individual domain randomization factors. Second, our tasks focus on rigid-object manipulation in kitchen-like environments; extending to deformable objects, dexterous manipulation, and broader household settings remains future work. Third, since SimVLA is a simulation-based data generation framework rather than a cross-embodiment transfer method or a new VLA architecture, our evaluation is limited to a subset of embodiments and architectures. Broader evaluation would be needed to assess its generality. Finally, SimAction generation relies on human-defined skill APIs; future work could leverage LLMs to expand skills as code-based policies, enabling a more autonomous pipeline.

7. Acknowledgements

This work was supported in part by the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (RS-2024-00333634, RS-2025-25396144). LG Electronics provided the Anubis robot, which was used for the experiments. The authors would like to thank Changwoo Yoo for the assistance with the preliminary experiments and real-world data collection.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz,

Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

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, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Kaixin Chai, Hyunjun Lee, and Joseph J Lim. N2m: Bridging navigation and manipulation by learning pose preference from rollout. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=yW9uNbRgOh.

Jiayi Chen, Yubin Ke, and He Wang. Bodex: Scalable and efficient robotic dexterous grasp synthesis using bilevel optimization. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–08. IEEE, 2025a.

Kaiyuan Chen, Shuangyu Xie, Zehan Ma, Pannag R Sanketi, and Ken Goldberg. Robo2vlm: Improving visual question answering using large-scale robot manipulation data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025b.

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 *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=itonej9GIV.

Xi Chen, Xiao Wang, Soravit Changpinyo, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu,

Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.

Shengliang Deng, Mi Yan, Songlin Wei, Haixin Ma, Yuxin Yang, Jiayi Chen, Zhiqi Zhang, Taoyu Yang, Xuheng Zhang, Heming Cui, Zhizheng Zhang, and He Wang. Graspvla: a grasping foundation model pre-trained on billion-scale synthetic action data. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine*

Learning Research, pages 1004–1029. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/deng25a.html.

Abhay Deshpande, Maya Guru, Rose Hendrix, Snehal Jauhri, Ainaz Eftekhar, Rohun Tripathi, Max Argus, Jordi Salvador, Haoquan Fang, Matthew Wallingford, Wilbert Pumacay, Yejin Kim, Quinn Pfeifer, Ying-Chun Lee, Piper Wolters, Omar Rayyan, Mingtong Zhang, Jiafei Duan, Karen Farley, Winson Han, Eli VanderBilt, Dieter Fox, Ali Farhadi, Georgia Chalvatzaki, Dhruv Shah, and Ranjay Krishna. Molmob0t: Large-scale simulation enables zero-shot manipulation. In *ICRA 2026 Workshop: From Data to Decisions: VLA Pipelines for Real Robots*, 2026. URL https://openreview.net/forum?id=bJRNV4Ddvz.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.

Clemens Eppner, Adithyavairavan Murali, Caelan Garrett, Rowland O'Flaherty, Tucker Hermans, Wei Yang, and Dieter Fox. scene_synthesizer: A python library for procedural scene generation in robot manipulation. *Journal of Open Source Software*, 2024.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

Caelan Reed Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2750–2790. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/garrett25a.html.

Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. N^2m^2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments. *IEEE Transactions on Robotics*, 39(5):3601–3619, 2023. doi: 10.1109/TRO.2023.3284346.

Snehal Jauhri, Sophie Lueth, and Georgia Chalvatzaki. Active-perceptive motion generation for mobile manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1413–1419. IEEE, 2024.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16923–16930, 2025. doi: 10.1109/ICRA55743.2025.11127809.

Taewoong Kang, Joonyoung Kim, Shady Nasrat, Dongwoon Song, Gijae Ahn, Minseong Jo, Seonil Lee, and Seung-Joon Yi. Anubis: A compact, low-cost, compliant humanoid mobile manipulation robot. In *24th International Conference on Humanoid Robots (Humanoids)*, 2025. URL https://ras.papercept.net/conferences/scripts/rtf/ICHR25_ContentListWeb_2.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Yejin Kim, Wilbert Pumacay, Omar Rayyan, Max Argus, Winson Han, Eli VanderBilt, Jordi Salvador, Abhay Deshpande, Rose Hendrix, Snehal Jauhri, et al. Molmospaces: A large-scale open ecosystem for robot navigation and manipulation. *arXiv preprint arXiv:2602.11337*, 2026.

Chengshu Li*, Mengdi Xu*, Arpit Bahety*, Hang Yin*, Yunfan Jiang, Huang Huang, Josiah Wong, Sujay Garlanka, Cem Gokmen, Ruohan Zhang, Weiyu Liu, Jiajun Wu, Roberto Martín-Martín, and Li Fei-Fei. Momagen: Generating demonstrations under soft and hard constraints for multi-step bimanual mobile manipulation, 2025. URL https://arxiv.org/abs/2510.18316.

Min Lin, Xiwen Liang, Bingqian Lin, Liu Jingzhi, Zijian Jiao, Kehan Li, Yuhan Ma, Yuecheng Liu, Shen Zhao, Yuzheng Zhuang, and Xiaodan Liang. Echovla: Robotic vision-language-action model with synergistic declarative memory for mobile manipulation, 2025a. URL https://arxiv.org/abs/2511.18112.

Toru Lin, Kartik Sachdev, Linxi Fan, Jitendra Malik, and Yuke Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids. In *Conference on Robot Learning*, pages 4926–4940. PMLR, 2025b.

Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, Scott Reed, Ken Goldberg, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. In *Robotics: Science and Systems (RSS)*, June 2025.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1820–1864. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/mandlekar23a.html.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026.

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

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo:

An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. Scenesmith: Agentic generation of simulation-ready indoor scenes. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=WwS8CTpUA6.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Ziyu Shan, Yuheng Zhou, Gaoyuan Wu, Ziheng Ji, Zhenyu Wu, and Ziwei Wang. Dockanywhere: Data-efficient visuomotor policy learning for mobile manipulation via novel demonstration generation. *IEEE Robotics and Automation Letters*, 2026.

Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024.

Nikhil Sobanbabu, Guanqi He, Tairan He, Yuxiang Yang, and Guanya Shi. Sampling-based system identification with active exploration for legged sim2real learning. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 578–598. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/sobanbabu25a.html.

Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Reed Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. Curobo: Parallelized collision-free robot motion generation. In *ICRA*, 2023.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, Yaping Li, Ping Wang, Junhao Cai, Jia Zeng, Hao Dong, and Jiangmiao Pang. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 976–985, June 2026.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: towards unleashing infinite data for automated robot learning via generative simulation. In *Proceedings of the 41st International Conference on Machine Learning*, pages 51936–51983, 2024.

Zhenyu Wu, Angyuan Ma, Xiuwei Xu, Hang Yin, Yinan Liang, Ziwei Wang, Jiwen Lu, and Haibin Yan. Moto: A zero-shot plug-in interaction-aware navigation for general mobile manipulation. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 2933–2948. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/wu25c.html.

Jingyun Yang, Isabella Huang, Brandon Vu, Max Bajracharya, Rika Antonova, and Jeannette Bohg. Mobi-π: Mobilizing your robot learning policy. In *Conference on Robot Learning*, pages 3516–3536. PMLR, 2025.

Ruihan Yang, Yejin Kim, Rose Hendrix, Aniruddha Kembhavi, Xiaolong Wang, and Kiana Ehsani. Harmonic mobile manipulation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3658–3665. IEEE, 2024.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A. Kitchen Scene

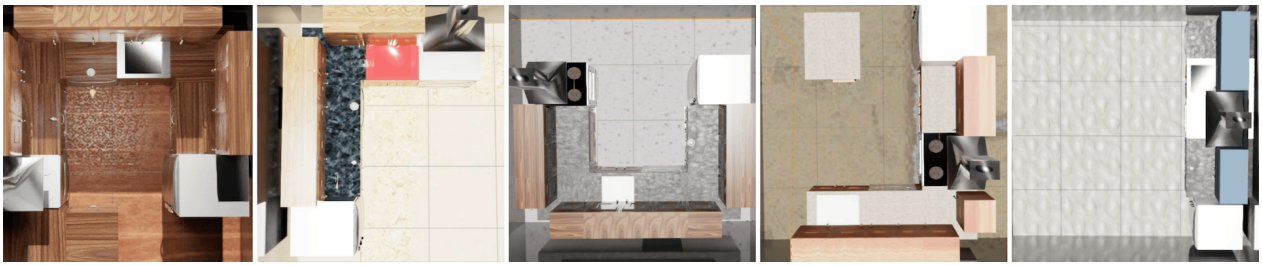


Figure 8. The five kitchen layout types used in SimVLA: peninsula, L-shaped, U-shaped, island, and single-wall. Each layout is procedurally generated with randomized furniture configurations, object placements, and surface materials.

Kitchen layout. To promote structural and visual diversity in our simulation environments, we use five common kitchen layouts from Scene Synthesizer (Eppner et al., 2024): island, L-shaped, peninsula, single-wall, and U-shaped, as illustrated in Figure 8. Each layout type represents a distinct spatial configuration commonly found in real-world kitchens. Each layout presents distinct navigation and manipulation challenges: *island* layouts require navigation around a central counter, *L-shaped* layouts create challenging corners, *peninsula* layouts form partially enclosed workspaces, *single-wall* layouts provide a compact linear workspace, and *U-shaped* layouts require precise base positioning across multiple surfaces. Within each layout, furniture heights, object arrangements, cabinet configurations, and surface materials are randomized to increase visual and geometric diversity.

Objects. To ensure robust generalization across diverse object geometries, SimVLA incorporates a large variety of object instances for each task-relevant category. As shown in Figure 9, diverse instances for each category (e.g., mugs, bottles, tables, bowls, food items, and utensils) are sourced from Objaverse (Deitke et al., 2023), spanning a wide range of shapes, sizes, aspect ratios, and surface materials. During scene generation, objects are sampled with randomized poses, scales, and material properties. This object-level diversity is critical, as policies trained on a narrow set of object instances tend to overfit to specific geometries and fail under even mild shape variation.



Figure 9. 40 bowl variants used in SimVLA, spanning diverse shapes, sizes, depths, and surface materials for robust sim-to-real generalization.

Scene Generator. To streamline scene construction, SimVLA provides a lightweight GUI for kitchen scene generation, as shown in Figure 10. Users specify the kitchen layout type (e.g., L-shaped), select object categories (e.g., bowl, bottle), and assign each object to a semantic placement location (e.g., "Above cabinet," "Above dishwasher"). The populated object list is displayed before committing, allowing users to review and adjust placements prior to generation. Clicking *Generate Kitchen Scene* instantiates the full scene with randomized object instances, poses, and materials. This interface enables rapid expansion of the training distribution.

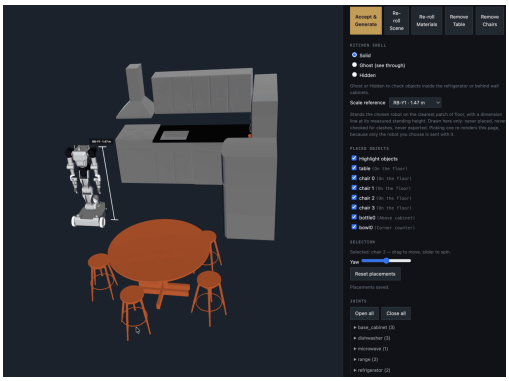


Figure 10. Kitchen scene generation UI.

B. Skill API

Atomic Skill Interface. SimVLA defines a library of atomic skills — including grasping, placing, pouring, opening, closing, navigating, and searching — each implemented as a Python function decorated with `@skill_register`. This decorator automatically registers the skill into the task composition framework, making it immediately available for use in long-horizon task definitions. An example of the grasping skill interface is shown in Listing 1.

```python
@skill_register(name="grasp", arm="right")
def grasp(env, object_name, arm="right"):
    pose = env.grasp_db.sample(object_name, arm)    # Sample from BODex
    move_arm_to_pose(env, arm, pose.intermediate)   # Intermediate grasp
    move_arm_to_pose(env, arm, pose.grasp)          # Final grasp
    close_gripper(env, arm)                         # Close gripper
    return SkillStatus.SUCCESS
```

Listing 1. Atomic grasping skill.

Search Skill API. Given a target object, the scripted expert uses its ground-truth pose and the simulator's collision geometry to synthesize a search trajectory in three stages. Ground-truth poses are used only during data generation; at inference, SimVLA reproduces the learned search behavior from visual observations alone.

1. *Rotate.* Turn in place until facing a goal position in front of the object.
2. *Approach.* Advance through waypoints toward a standoff point, each waypoint laterally perturbed and collision-checked to diversify approach paths.
3. *Align.* Perturb the final heading within the field of view, so the object appears at varied image locations rather than always centered.



Figure 11. Real-world search behavior. Given only a language prompt and RGB observations, the policy executes searching motions until the sink is visible, then proceeds to complete the task.

Grasp Pose Selection. For grasping skills, SimVLA integrates BODex (Chen et al., 2025a) to generate diverse candidate grasp poses for each object. As shown in Figure 12, a GUI presents up to 200 candidates, organized by gripper configuration and arm side (left/right). Users can optionally select preferred grasp poses through the GUI, while the automatically generated candidates provide broad coverage. This human-in-the-loop selection helps ensure that the resulting demonstrations reflect realistic grasping behaviors rather than arbitrary contact configurations.

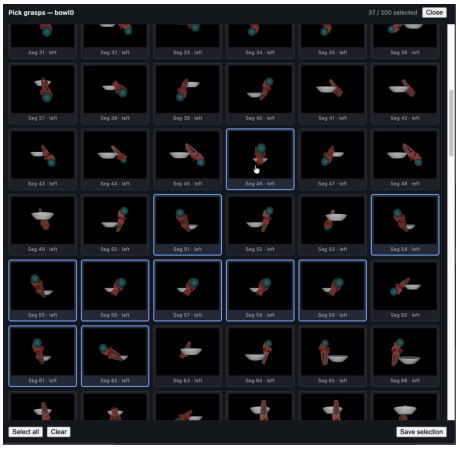


Figure 12. BODex grasp pose generation and selection GUI.

C. Hybrid System Identification

Accurate sim-to-real transfer requires that the simulated robot dynamics closely match the real hardware. We perform hybrid system identification to calibrate physical parameters by jointly minimizing end-effector position error, end-effector rotation error, and joint-space MSE between simulation and the real robot. Unlike prior work that matches only joint-space trajectories (Lin et al., 2025b), our hybrid objective additionally constrains end-effector tracking, since SimVLA operates in end-effector space. To accelerate the search, we run 4096 parallel simulation environments in Isaac Lab simultaneously, as illustrated in Figure 13, reducing wall-clock time for parameter calibration from hours to minutes.



Figure 13. Massive parallel environments (4096 instances) running simultaneously in Isaac Lab for hybrid system identification.

Algorithm 1 describes the full procedure. A population of K candidate simulation environments is sampled from the parameter space, and each candidate is evaluated against real robot tracking signals collected on N calibration sequences. A candidate is accepted only if all three error metrics — EEF position (e_p), EEF rotation (e_r), and joint MSE (e_j) — remain below their respective thresholds τ_p, τ_r, and τ_j across all timesteps.

Algorithm 1 Hybrid System Identification

Require: E: parameter space; M: robot model; K: number of parallel candidate environments; N: number of calibration sequences; R: real robot; $\tau_p = 0.03\,\text{m}$, $\tau_r = 0.2\,\text{rad}$, $\tau_j = 0.0025\,\text{rad}^2$: error thresholds

procedure AUTOTUNE($E, M, K, N, R, \tau_p, \tau_r, \tau_j$)
  $\mathcal{S} \leftarrow \{\textsc{SimEnv}(M, p_k) \mid p_k \sim \textsc{Sample}(E)\}_{k=1}^{K}$
  $\mathcal{U} \leftarrow \textsc{GenSeqs}(N)$ ▷ N calibration sequences
  $\mathbf{R} \leftarrow \textsc{GetSignals}(R,\ \mathcal{U})$ ▷ Real $\{pos, rot, q\}$
  $J^\star \leftarrow \infty$; $p^\star \leftarrow$ **null**; $\mathbf{e}^\star \leftarrow$ **null**
  for $k = 1, \ldots, K$ **do**
    $\mathbf{S} \leftarrow \textsc{GetSignals}(S_k,\ \mathcal{U})$ ▷ Sim $\{pos, rot, q\}$
    $e_p \leftarrow \max_{n,t} \|S^{pos}_{n,t} - R^{pos}_{n,t}\|_2$ ▷ EEF position error
    $e_r \leftarrow \max_{n,t}\ \textsc{QuatAngleError}(S^{rot}_{n,t}, R^{rot}_{n,t})$ ▷ EEF rotation error
    $e_j \leftarrow \text{MSE}(S^q, R^q)$ ▷ Joint tracking error
    $J \leftarrow e_p/\tau_p + e_r/\tau_r + e_j/\tau_j$ ▷ Normalized hybrid score
    if $J < J^\star$ **then** ▷ Keep running minimum
      $J^\star \leftarrow J$; $p^\star \leftarrow p_k$; $\mathbf{e}^\star \leftarrow [e_p, e_r, e_j]$
    end if
  end for
  if $\mathbf{e}^\star \preceq [\tau_p, \tau_r, \tau_j]$ **then** ▷ Element-wise threshold check
    return $p^\star$
  end if
  return null ▷ Best candidate still exceeds a threshold
end procedure

Ablation study. We compare four calibration settings: no identification (default URDF gains), joint-space only (e_j), EEF-space only (e_p, e_r), and our hybrid objective. For each setting, we replay ten held-out real trajectories spanning three manipulation tasks and measure the EEF position discrepancy

between the real and simulated robots, averaged over both arms of the Anubis robot.

Table 3. System-identification ablation. EEF position discrepancy over ten held-out trajectories on both arms; $\pm$ denotes standard deviation across trajectories.

Objective	**L2 mean (mm)** $\downarrow$	**L2 max (mm)** $\downarrow$
None (URDF defaults)	83.4 ± 50.2	120.7 ± 96.0
Joint-space	14.7 ± 2.3	46.2 ± 9.6
EEF-space	13.4 ± 1.7	40.8 ± 8.4
Hybrid (joint + EEF)	$\mathbf{13.3 \pm 1.6}$	$\mathbf{40.6 \pm 8.1}$

Calibration has a substantial effect: hybrid identification cuts mean EEF error by $6.3\times$ ($83.4 \rightarrow 13.3$ mm) and maximum error by $3.0\times$ ($120.7 \rightarrow 40.6$ mm) over default gains, while sharply reducing variance across trajectories. The choice of objective has a much smaller effect. Hybrid is best on both metrics but gains only 1.4 mm in mean error over joint-space identification and is within noise of EEF-space identification, suggesting that the primary benefit comes from performing system identification itself.

D. SimAction

SimAction comprises long-horizon manipulation tasks and atomic manipulation skills in kitchen environments. These tasks involve navigation, object grasping, articulated manipulation, object placement, and sequential task execution. SimVLA includes 35 tasks, comprising 22 long-horizon compositions and 13 atomic skills, with each atomic skill also released as a standalone task. Each task provides three RGB streams (front, left wrist, and right wrist) at 240×320 resolution, recorded at 20 fps. The robot action space is 23-D, comprising a 10-D end-effector representation for each arm (3-D position, 6-D rotation and 1-D gripper), together with the base velocities (v_x, v_y, ω). We additionally provide the proprioceptive state and the initial robot pose. The SimAction tasks are listed below.

Table 4. Task definitions.

ID	Task	Description
1	Move to mug	The robot navigates and positions itself near the mug.
2	Move to bowl	The robot navigates and positions itself near the bowl.
3	Move to bottle	The robot navigates and positions itself near the bottle.
4	Move to drawer	The robot navigates and positions itself in front of the drawer.
5	Put mug in sink	The robot navigates to the mug, grasps it, moves to the sink, and places it into the sink.
6	Put bowl in sink	The robot navigates to the bowl, grasps it, moves to the sink, and places it into the sink.
7	Put bottle in sink	The robot navigates to the bottle, grasps it, moves to the sink, and places it into the sink.
8	Put mug on table	The robot navigates to the mug, grasps it, moves to the table, and gently places it on the table.
9	Put bowl on table	The robot navigates to the bowl, grasps it, moves to the table, and gently places it on the table.
10	Put bottle on table	The robot navigates to the bottle, grasps it, moves to the table, and gently places it on the table.
11	Put apple on table	The robot navigates to the apple, grasps it, moves to the table, and gently places it on the table.
12	Put apple in bowl	The robot navigates to and grasps the apple, moves to the bowl, and places the apple into the bowl.
13	Put apple beside bowl	The robot navigates to and grasps the apple, moves to the bowl, and places the apple beside the bowl.
14	Put apple beside mug	The robot navigates to and grasps the apple, moves to the mug, and places the apple beside the mug.
15	Serve bowl	The robot navigates to and grasps a bowl, transports it to a serving location, and places it down.
16	Serve mug	The robot navigates to and grasps a mug, transports it to a serving location, and places it down.
17	Put pot on stove	The robot navigates to the pot, grasps it, moves to the stove, and places it onto the stove.
18	Put ramen in pot	The robot grasps ramen ingredients and places them into the pot.
19	Place bowl in drawer	The robot places a grasped bowl into the opened drawer.
20	Open drawer	The robot grasps the drawer handle and opens the drawer.
21	Close drawer	The robot closes the drawer after manipulation.
22	Open microwave	The robot moves to the microwave, grasps the handle, and opens it by moving its arm.
23	Close microwave	The robot moves to the microwave, grasps the handle, and closes it by moving its arm.
24	Open refrigerator	The robot navigates to the refrigerator, grasps the handle, and opens the refrigerator door.
25	Close refrigerator	The robot grasps the handle and closes the refrigerator door.
26	Open dishwasher	The robot moves to the dishwasher, grasps the handle, and opens it by moving its arm and base.
27	Close dishwasher	The robot moves to the dishwasher, grasps the handle, and closes it by moving its arm and base.
28	Pour water into pot	The robot grasps a water container and pours water into the pot.
29	Push chair	The robot navigates to the chair and pushes it toward the table using both arms.
30	Put mug and bowl in sink	The robot sequentially navigates to and grasps both the mug and the bowl, moves to the sink, and places them into the sink.
31	Put bowl in drawer	The robot navigates to and grasps the bowl, moves to the drawer, opens the drawer, places the bowl inside, and closes the drawer.
32	Put mug in drawer	The robot navigates to and grasps the mug, moves to the drawer, opens the drawer, places the mug inside, and closes the drawer.
33	Pour from bottle to mug	The robot navigates to and grasps the bottle and the mug, then pours from the bottle into the mug.
34	Boil sweet potato	The robot places sweet potatoes into a pot and performs the cooking procedure.
35	Prepare snack	The robot sequentially manipulates multiple food-related objects to prepare a snack setup.

E. Domain Randomization

To bridge the sim-to-real gap, we apply extensive domain randomization across six categories: camera perturbations, kitchen scene properties, object physical and visual properties, robot dynamics, and lighting conditions. Rather than relying solely on system identification to match real-world dynamics, domain randomization exposes the policy to a broad distribution of physical and visual conditions during training, encouraging it to learn behaviors that are robust to the inevitable discrepancies between simulation and reality. The full randomization setup is summarized in Table 5.

Table 5. Domain Randomization Setup.

Domain	**Randomness**
Camera: Position Noise (m)	$\sim \mathcal{N}(0, 0.03^2)$
Camera: Rotation Noise (deg)	$\sim \mathcal{U}(-5, 5)$
Kitchen: Types	5
Kitchen: Tables	50
Kitchen: Chairs	31
Kitchen: Cabinet Materials	21
Kitchen: Wall Materials	33
Kitchen: Floor Materials	18
Kitchen: Cabinet Height (m)	$\sim \mathcal{N}(0.85, 0.03^2)$
Kitchen: Articulation Furniture PD Controller: P Gain	$\sim \mathcal{U}(1, 25)$
Kitchen: Articulation Furniture PD Controller: D Gain	$\sim \mathcal{U}(0, 1)$
Kitchen: Handle Static Contact Friction Coefficient	$\sim \mathcal{U}(0.3, 0.8)$
Kitchen: Handle Dynamic Contact Friction Coefficient	$\sim \mathcal{U}(0.2, 0.6)$
Kitchen: Handle Position	$\sim \mathcal{U}(-0.05, 0.05)$
Object: Mass (kg)	$\sim \mathcal{U}(0.1, 0.5)$
Object: Static Contact Friction Coefficient	$\sim \mathcal{U}(0.2, 0.6)$
Object: Dynamic Contact Friction Coefficient	$\sim \mathcal{U}(0.1, 0.5)$
Object: Restitution Coefficient	$\sim \mathcal{U}(0.3, 0.7)$
Object: Shape	$\times \sim \mathcal{N}(1, 0.07^2)$
Object: Initial Position Noise (m)	$\sim \mathcal{U}(-0.1, 0.1)$
Object: Initial z-orientation (deg)	$\sim \mathrm{Cat}(\{0, 30, 60, \ldots, 330\})$
Object: Center of Mass (x-axis) (m)	$\sim \mathcal{U}(-0.005, 0.005)$
Object: Center of Mass (y-axis) (m)	$\sim \mathcal{U}(-0.005, 0.005)$
Object: Center of Mass (z-axis) (m)	$\sim \mathcal{U}(-0.02, 0.02)$
Object: Color (RGB)	$\sim \mathcal{U}(0.0, 1.0)$
Object: Mug Types	80
Object: Bowl Types	40
Object: Bottle Types	20
Robot: Initial Position	Anywhere in the room without collision
Robot: Initial Rotation (deg)	$\sim \mathcal{U}(0, 360)$
Robot: Initial Joint Position Noise	$\sim \mathcal{U}(-0.1, 0.1)$
Robot: Initial Joint Velocity Noise	$\sim \mathcal{U}(-0.01, 0.01)$
Robot: Static Contact Friction Coefficient	$\sim \mathcal{U}(0.7, 1.0)$
Robot: Dynamic Contact Friction Coefficient	$\sim \mathcal{U}(0.5, 1.0)$
Robot: Restitution Coefficient	$\sim \mathcal{U}(0.3, 0.5)$
Robot: PD Controller (P Gain)	$\times \sim \mathcal{N}(1, 0.1^2)$
Robot: PD Controller (D Gain)	$\times \sim \mathcal{N}(1, 0.1^2)$
Robot: Color (RGB range [0,1])	$\sim \mathcal{U}(0.0, 0.1)$
Light: Intensity scale	$\sim 2^{\mathcal{U}(-1.5,\, 1.5)}$ $(I = 3000 \cdot \text{scale})$
Light: Color temperature (K)	$\sim \mathrm{Cat}\{2700, 3200, 4000, 5000, 6500, 8000, 9000\}$
Light: Color (RGB)	$\sim \mathrm{clip}\big(\mathbf{1} + \mathcal{N}(0, 0.08^2),\, [0.7, 1.3]\big)$

F. SimVQA

SimVQA is a visual question-answering dataset automatically generated from simulation to inject geometric and subtask-level supervision into the VLM backbone. By leveraging ground-truth simulator state, SimVQA provides precise spatial labels that would be prohibitively expensive to obtain from real-world data. Table 6 lists the full set of VQA templates used, covering high-level task planning, subtask sequencing, object detection, 3D spatial reasoning, cross-view correspondence, reachability estimation, graspability assessment, and gripper state queries. Our dataset contains approximately 1M SimVQA visual question-answering samples.

Table 6. SimVQA Templates

VQA Function	Question Prompt Prototype	Answer Prototype
`high_level_subtask`	To complete the task "{task}", what should the robot do now?	{subtask}
`next_subtask`	After current subtask {subtask}, what is the next subtask?	{subtask}
`object_detection`	Is the {object} visible in the front camera?	yes. bbox: {y1,x1,y2,x2} / no
`object_3d_information`	How far is the {object} from the robot base?	dx {x} dy {y} dz {z}
`view_correspondence`	Given front bbox {bbox}, what are the wrist camera bboxes?	left {bbox} right {bbox}
`object_reachable`	Is the robot close enough to manipulate the {object}?	yes / no. move base dx {x} dy {y} dyaw {yaw}
`object_graspable`	Is the {object} graspable with left gripper?	yes / no. move gripper dx {x} dy {y} dz {z} dquat {quat}
`left_gripper_open`	Is the robot's left gripper closed?	yes / no
`right_gripper_open`	Is the robot's right gripper closed?	yes / no

G. SimVLA Training

SimVLA follows a two-stage training pipeline built on a pre-trained PaliGemma VLM backbone and a flow-matching action expert. The two stages differ in which losses are active and whether gradients are allowed to flow back into the VLM backbone.

Pre-training: Knowledge-isolated co-training. We denote the input tokens of an example by $x = (x^p, x^f, x^c, x^a)$, where x^p are the *prefix* tokens (image patches encoded by SigLIP, language instruction, and proprioceptive state), x^f are FAST-tokenized action tokens (Pertsch et al., 2025), x^c are SimVQA language tokens, and $x^a = a_{1:H}^{\tau,\omega}$ are noisy continuous-action tokens with $a_i^{\tau,\omega} = \tau a_i + (1-\tau)\omega_i$, where $\tau \sim p(\tau)$ and $\omega \sim \mathcal{N}(0, I)$. The flow-matching action expert f_θ^a predicts the vector field from the noisy action x^a toward the clean action trajectory, conditioned on the prefix and interpolation time τ. For SimVLA, we use $H{=}50$ and $d{=}23$.

Each example b is assigned three Bernoulli masks $M_b^a, M_b^f, M_b^c \in \{0, 1\}$, which determine whether the flow-matching, FAST, and VQA losses, respectively, are applied to that example. Our work follows the knowledge insulation framework (Driess et al., 2025). The three objectives are optimized in a single forward pass:

$$\mathcal{L}_{\text{flow}}(\theta) = \mathbb{E}_{\mathcal{D},\tau,\omega}\Big[M^a \,\|\, (a_{1:H} - \omega) - f_\theta^a(x^a;\, x^p, \tau) \,\|^2\Big], \tag{1}$$

$$\mathcal{L}_{\text{fast}}(\theta) = \mathbb{E}_{\mathcal{D}}\Big[- M^f \sum_{j=1}^{n_f} \log p_\theta\Big(x_j^f \mid x^p, x_{<j}^f\Big)\Big], \tag{2}$$

$$\mathcal{L}_{\text{vqa}}(\theta) = \mathbb{E}_{\mathcal{D}}\Big[- M^c \sum_{j=1}^{n_c} \log p_\theta\Big(x_j^c \mid x^p, x_{<j}^c\Big)\Big], \tag{3}$$

$$\mathcal{L}_{\text{pre}}(\theta) = \mathcal{L}_{\text{flow}}(\theta) + \alpha_f\, \mathcal{L}_{\text{fast}}(\theta) + \alpha_c\, \mathcal{L}_{\text{vqa}}(\theta). \tag{4}$$

Here, n_f and n_c denote the lengths of the FAST-action and SimVQA token sequences, respectively, and $x_{<1}^f = x_{<1}^c = \emptyset$. Thus, the first FAST or VQA token is predicted directly from the prefix x^p, while subsequent tokens are predicted autoregressively from the preceding postfix tokens.

To prevent the randomly initialized action expert from corrupting the pre-trained VLM backbone, we apply knowledge insulation. Specifically, gradients from the flow loss are stopped at the VLM backbone parameters θ_b, while the FAST and VQA losses update the backbone normally. The attention mask is additionally modified to prevent cross-attention between continuous action tokens and FAST/VQA tokens, ensuring that the three supervision streams remain decoupled within the same forward pass.

Post-training. Once the action expert is warmed up during pre-training, we lift the stop-gradient so that the flow loss can back-propagate into the VLM backbone θ_b. The FAST and VQA losses are dropped at this stage, and the model is fine-tuned exclusively on mobile-manipulation tasks matching the downstream evaluation setting. The post-training objective is

$$\mathcal{L}_{\text{post}}(\theta_b, \theta_a) = \mathbb{E}_{\mathcal{D}_{\text{post}},\tau,\omega}\Big[\big\|(a_{1:H} - \omega) - f_{\theta_b,\theta_a}^a(x^a;\, x^p, \tau)\big\|^2\Big], \tag{5}$$

where θ_a and θ_b denote the action-expert and VLM backbone parameters, respectively, and gradients from the flow loss are allowed to update both parameter sets.

We train on $\mathcal{D}_{\text{post}} = \mathcal{D}_{\text{cont}} \cup \mathcal{D}_{\text{deploy}}$, where $\mathcal{D}_{\text{cont}}$ is the original continuous-action corpus used in pre-training and $\mathcal{D}_{\text{deploy}}$ is the mobile-manipulation deployment dataset collected from policy rollouts across diverse simulated environments and matched to the downstream evaluation setting. Because

the action expert is already warmed up, allowing gradients to flow into the backbone during this stage enables task-specific adaptation while avoiding the instability that could arise from jointly training a randomly initialized action expert with the pre-trained VLM representations.

Hyperparameters and Compute. SimVLA is trained in two stages using AdamW with cosine learning rate scheduling. Tables 7–9 summarize the full hyperparameter configuration and compute resources used for pre-training and post-training. In total, the reported experiments required approximately 1,272 GPU-hours. Beyond the experiments included in the paper, additional compute was used for preliminary experiments, hyperparameter tuning, failed runs, and debugging. These additional runs required substantially more compute than the final reported experiments.

Table 7. Key hyperparameters for SimVLA training.

Hyperparameter	Pretrain	Post-train
Global batch size	256	256
Precision	BF16	BF16
Gradient clipping (L2)	1.0	1.0
Peak learning rate	5×10^{-5}	5×10^{-5}
End learning rate	5×10^{-5}	5×10^{-5}
LR scheduler	cosine	cosine
Warm-up steps	1,000	1,000
Warm-up ratio	0.00	0.01
Decay steps	180k	50k
Total steps	180k	50k
Optimizer	AdamW	AdamW
Adam β_1, β_2	0.9, 0.95	0.9, 0.95
Adam ϵ	1×10^{-8}	1×10^{-8}
Weight decay	1×10^{-10}	1×10^{-10}
EMA decay	0.999	0.999
Vision backbone frozen	false	false
Action chunk size	50	50
Sampling steps	10	10
Data workers	32	32
Weight init	Paligemma base	SimVLA pretrain ckpt

Table 8. Co-training and FAST auxiliary-loss hyperparameters used during SimVLA pretraining.

Hyperparameter	Value
FAST loss weight	0.1
VQA loss weight	0.1
FAST sampling ratio	0.50
Co-train sampling ratio	0.01
Discrete state input	false
Delta joint actions	true

Table 9. Compute resources used for SimVLA experiments. All experiments were conducted on an internal GPU cluster. Reported times are approximate wall-clock durations and may vary depending on storage and data-loading throughput.

Stage	Hardware	CPU	System RAM	Storage	Per-run compute	Total compute
Pre-training	4× NVIDIA B200 (183GB)	128 CPU cores	1.5 TB	NVMe SSD	∼12 days	∼1152 GPU-hours
Post-training	4× NVIDIA B200 (183GB)	128 CPU cores	1.5 TB	NVMe SSD	∼30 hours	∼120 GPU-hours

H. Experimental Setup

Robot Hardware Spec We conduct our real-world experiments using Anubis robot (Kang et al., 2025). The platform features a teleoperation system inspired by the Mobile ALOHA setup (Fu et al., 2024). Each arm has 6 DoF and is equipped with a parallel gripper and a wrist-mounted camera. At the center of the robot, an Intel RealSense camera is mounted on a height-adjustable mechanism, serving as the ego-centric view camera. Details are described in Table 10. Anubis is equipped with a 3-wheel omni-directional base that supports planar locomotion.

Table 10. Anubis Robot Hardware Specifications.

Component	Specification
Base Type	3-wheel omni-directional chassis
Mobility DOF	3 (X, Y, Yaw)
Arm DOF	2 × (6 DOF + gripper) = 14
Total DOF	14 arm + 3 base = 17
Policy action space	2 × (9D EEF + 1D gripper) + 3D base = 23-D
Front Camera	Intel RealSense D455
Wrist Cameras	Intel RealSense D405
Gripper	Parallel transparent gripper (hole design, ALOHA-style)
Power System	3 × Greenworks 40V 5.0Ah batteries (PC, wheels & leader/follower)
Frame	3D-printed custom components



Figure 14. The Anubis robot.

Task Progress Description. We define discrete task progress labels for each task to provide finer-grained evaluation beyond binary success metrics.

- **Task 1: Put mug in sink**
  - **0**: Failed grasp.
  - **1**: Failed to place the mug into the sink.
  - **2**: Success.
- **Task 2: Put bowl in drawer**
  - **0**: Failed grasp.
  - **1**: Failed to grasp the drawer handle.
  - **2**: Failed to open the drawer (e.g., slipped during opening).
  - **3**: Failed to place the bowl into the drawer.
  - **4**: Failed to close the drawer.
  - **5**: Success.
- **Task 3: Pour from bottle to mug**
  - **0**: Failed to grasp the bottle or mug.
  - **1**: Failed to grasp both objects.
  - **2**: Failed to tilt the bottle for pouring.
  - **3**: Success.
- **Task 4: Put bowl in sink**

  - **0**: Failed grasp.
  - **1**: Failed to place the bowl into the sink.
  - **2**: Success.

- **Task 5: Put bottle in sink**
  - **0**: Failed grasp.
  - **1**: Failed to place the bottle into the sink.
  - **2**: Success.

Table 11. Held-out 1,369 SimVQA evaluation. Ablation study on visual reasoning capabilities. ↑ indicates higher is better and ↓ indicates lower is better.

Model	Gripper ↑ L / R	Subtask ↑ Curr. / Next	3D Pos. ↓ (m)	Grasp ↓ (m)	Cross-view IoU ↑
SimVLA w/o SimVQA	–*	–*	–*	–*	–*
SimVLA w/o TokSimAction	0.86 / 0.69	0.18 / 0.39	0.50	0.16	0.04
SimVLA w/o both	0.85 / 0.68	–†	–†	–†	–†
SimVLA	**1.00 / 0.89**	**0.69 / 0.77**	**0.06**	**0.05**	**0.61**

* Emits tokenized actions; metrics are undefined. † Emits invalid answers; metrics are undefined.

Long-Horizon Simulation Evaluation. To provide evidence beyond the real-world tasks, we post-trained and evaluated SimVLA on eight long-horizon tasks in simulation using Anubis robot. We report the success rate over 50 rollouts per task across five kitchen setups (10 rollouts per setup). The results are summarized in Table 12. These tasks cover several distinct skill compositions, including visual navigation, grasping different object types (mug, bowl, and bottle), placement into sinks and drawers, articulated drawer opening and closing, and bimanual pouring. The lower success rate for *Put mug and bowl to sink* (19/50) is expected, as the task chains two complete navigation and pick-and-place sequences, increasing the number of opportunities for failure.

Table 12. SimVLA success rates on eight long-horizon simulation tasks using Anubis robot. Each task is evaluated over 50 rollouts across five unseen kitchen setups.

Task	Success Rate
Put mug to sink	76%
Put bowl in drawer	70%
Pour from bottle to mug	76%
Put mug and bowl to sink	38%
Put bowl to sink	62%
Put bottle to sink	64%
Put mug in drawer	44%
Put bottle in drawer	58%

Table 13. Mock Kitchen — ID. The $\pi_{0.5}$ model fine-tuned with 200 demonstrations achieves the best performance on all tasks. SimVLA outperforms $\pi_{0.5}$ fine-tuned on 50 real-world demonstrations in task progress across all tasks, with comparable or higher binary success. Task progress $\pm$ s.d. over 20 trials; binary successes in parentheses.

Model	Task 1	Task 2	Task 3	Task 4	Task 5
$\pi_{0.5}$ fine-tuned w/ 50 demos	0.55 ± 0.43 (8/20)	0.36 ± 0.39 (5/20)	0.48 ± 0.38 (6/20)	0.45 ± 0.46 (7/20)	0.43 ± 0.47 (7/20)
$\pi_{0.5}$ fine-tuned w/ 200 demos	$\mathbf{0.75 \pm 0.44}$ (15/20)	$\mathbf{0.66 \pm 0.42}$ (11/20)	$\mathbf{0.70 \pm 0.36}$ (10/20)	$\mathbf{0.65 \pm 0.43}$ (11/20)	$\mathbf{0.63 \pm 0.46}$ (11/20)
SimVLA (w/o pre-train)	0.28 ± 0.38 (3/20)	0.15 ± 0.16 (0/20)	0.15 ± 0.25 (1/20)	0.18 ± 0.34 (2/20)	0.18 ± 0.29 (1/20)
SimVLA (w/o SimDeploy)	0.45 ± 0.32 (3/20)	0.38 ± 0.31 (3/20)	0.45 ± 0.33 (4/20)	0.33 ± 0.34 (2/20)	0.33 ± 0.34 (2/20)
SimVLA	0.60 ± 0.42 (9/20)	0.50 ± 0.36 (6/20)	0.52 ± 0.38 (6/20)	0.55 ± 0.46 (9/20)	0.55 ± 0.43 (8/20)

Table 14. Mock Kitchen — OOD-Pos. SimVLA is best on task progress across all tasks, and best-or-comparable on binary success. Task progress $\pm$ s.d. over 20 trials; binary successes in parentheses.

Model	Task 1	Task 2	Task 3	Task 4	Task 5
$\pi_{0.5}$ fine-tuned w/ 50 demos	0.40 ± 0.38 (4/20)	0.33 ± 0.37 (4/20)	0.40 ± 0.37 (4/20)	0.30 ± 0.44 (5/20)	0.25 ± 0.41 (4/20)
$\pi_{0.5}$ fine-tuned w/ 200 demos	0.45 ± 0.39 (5/20)	0.43 ± 0.40 (6/20)	0.48 ± 0.35 (4/20)	0.35 ± 0.46 (6/20)	0.28 ± 0.44 (5/20)
SimVLA (w/o pre-train)	0.23 ± 0.34 (2/20)	0.17 ± 0.18 (0/20)	0.13 ± 0.17 (0/20)	0.10 ± 0.26 (1/20)	0.25 ± 0.38 (3/20)
SimVLA (w/o SimDeploy)	0.45 ± 0.36 (4/20)	0.36 ± 0.31 (3/20)	0.43 ± 0.38 (5/20)	0.38 ± 0.32 (2/20)	0.50 ± 0.32 (4/20)
SimVLA	$\mathbf{0.58 \pm 0.41}$ (8/20)	$\mathbf{0.50 \pm 0.36}$ (6/20)	$\mathbf{0.52 \pm 0.37}$ (6/20)	$\mathbf{0.60 \pm 0.45}$ (10/20)	$\mathbf{0.68 \pm 0.41}$ (11/20)

Table 15. Mock Kitchen — OOD-Vis. SimVLA achieves the best performance on Tasks 1, 3, 4 and 5, whereas the $\pi_{0.5}$ model fine-tuned with 200 demonstrations performs best on Task 2. Relative to OOD-Pos, $\pi_{0.5}$ holds up better under OOD-Vis shifts, likely due to image augmentation during its fine-tuning. Task progress $\pm$ s.d. over 20 trials; binary successes in parentheses.

Model	Task 1	Task 2	Task 3	Task 4	Task 5
$\pi_{0.5}$ fine-tuned w/ 50 demos	0.35 ± 0.46 (6/20)	0.37 ± 0.39 (5/20)	0.40 ± 0.37 (4/20)	0.50 ± 0.49 (9/20)	0.30 ± 0.41 (4/20)
$\pi_{0.5}$ fine-tuned w/ 200 demos	0.45 ± 0.48 (8/20)	$\mathbf{0.45 \pm 0.42}$ (7/20)	0.45 ± 0.39 (5/20)	0.53 ± 0.47 (9/20)	0.48 ± 0.47 (8/20)
SimVLA (w/o pre-train)	0.20 ± 0.34 (2/20)	0.13 ± 0.15 (0/20)	0.13 ± 0.17 (0/20)	0.18 ± 0.29 (1/20)	0.08 ± 0.18 (0/20)
SimVLA (w/o SimDeploy)	0.38 ± 0.36 (3/20)	0.29 ± 0.28 (2/20)	0.37 ± 0.36 (4/20)	0.43 ± 0.34 (3/20)	0.43 ± 0.29 (2/20)
SimVLA	$\mathbf{0.63 \pm 0.39}$ (9/20)	0.40 ± 0.34 (4/20)	$\mathbf{0.47 \pm 0.38}$ (6/20)	$\mathbf{0.58 \pm 0.41}$ (8/20)	$\mathbf{0.60 \pm 0.38}$ (8/20)

Table 16. Pretraining Ablation, same tasks as submitted, 20 rollouts for each evaluation in mock kitchen-ID. Removing either SimVQA or TokSimAction lowers task progress on every task relative to full SimVLA; removing SimVQA degrades control mainly through less precise grasping. Task progress $\pm$ s.d.; binary successes in parentheses.

Model	Task 1	Task 2	Task 3	Task 4	Task 5
SimVLA w/o SimVQA	0.43 ± 0.47 (7/20)	0.36 ± 0.41 (5/20)	0.37 ± 0.40 (5/20)	0.50 ± 0.43 (7/20)	0.45 ± 0.43 (6/20)
SimVLA w/o TokSimAction	0.40 ± 0.38 (4/20)	0.29 ± 0.35 (3/20)	0.42 ± 0.37 (5/20)	0.43 ± 0.41 (5/20)	0.50 ± 0.43 (7/20)
SimVLA w/o both	0.38 ± 0.36 (3/20)	0.26 ± 0.35 (3/20)	0.37 ± 0.36 (4/20)	0.48 ± 0.44 (7/20)	0.45 ± 0.43 (6/20)
SimVLA	$\mathbf{0.60 \pm 0.42}$ (9/20)	$\mathbf{0.50 \pm 0.36}$ (6/20)	$\mathbf{0.52 \pm 0.38}$ (6/20)	$\mathbf{0.55 \pm 0.46}$ (9/20)	$\mathbf{0.55 \pm 0.43}$ (8/20)

Table 17. Pantry and Real Home. Task progress $\pm$ s.d.; binary successes in parentheses. Pantry is evaluated on Tasks 1, 4 and 5 with 20 trials each; Real Home on Tasks 1 and 3 with 10 trials each.

	Pantry (20 trials)			Real Home (10 trials)	
Model	Task 1	Task 4	Task 5	Task 1	Task 3
$\pi_{0.5}$ fine-tuned w/ 50 Mock Kitchen demos	0.10 ± 0.21 (0/20)	0.08 ± 0.18 (0/20)	0.18 ± 0.24 (0/20)	0.20 ± 0.26 (0/10)	0.17 ± 0.18 (0/10)
$\pi_{0.5}$ fine-tuned w/ 200 Mock Kitchen demos	0.23 ± 0.30 (1/20)	0.23 ± 0.30 (1/20)	0.18 ± 0.29 (1/20)	0.35 ± 0.24 (0/10)	0.23 ± 0.16 (0/10)
SimVLA w/o pre-train	0.10 ± 0.21 (0/20)	0.10 ± 0.26 (1/20)	0.08 ± 0.18 (0/20)	0.10 ± 0.21 (0/10)	0.10 ± 0.16 (0/10)
SimVLA	$\mathbf{0.60 \pm 0.38}$ (8/20)	$\mathbf{0.53 \pm 0.47}$ (9/20)	$\mathbf{0.58 \pm 0.44}$ (9/20)	$\mathbf{0.70 \pm 0.42}$ (6/10)	$\mathbf{0.47 \pm 0.42}$ (3/10)