EVO-DEXBENCH: BENCHMARKING GENERALIZATION ACROSS RELATED TASKS FOR DEXTEROUS MANIPULATION

Anonymous authors
Paper under double-blind review

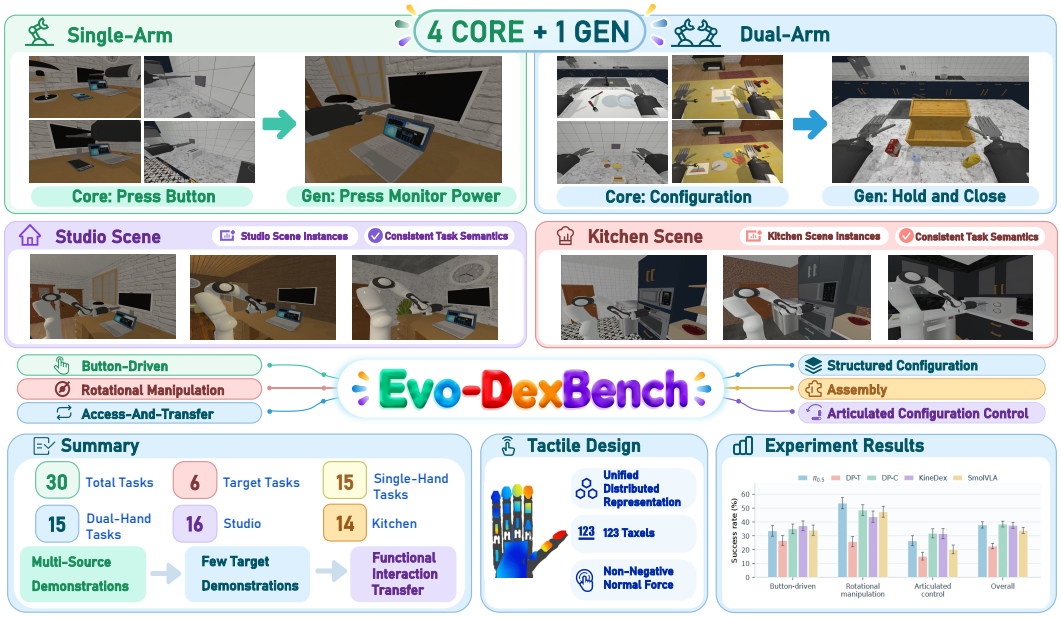


Figure 1: **Overview of the Evo-DexBench.** The benchmark comprises six functional task categories, each containing four Core tasks and one Gen task. Each task randomly switches among three scenes within its corresponding Kitchen or Studio environment during data collection and evaluation. The bottom-right panel reports the performance of representative models.

ABSTRACT

General-purpose dexterous manipulation requires policies to transfer knowledge from related tasks to new tasks with only a few target demonstrations. Existing benchmarks often provide too many target demonstrations. They also leave the differences between source and target tasks uncontrolled. As a result, they cannot clearly distinguish knowledge transfer from task memorization. Our key insight is the **Core-Gen principle**. **Core tasks** are related tasks that share a functional interaction and provide source knowledge, while a **Gen task** is a novel target within the same functional family that has only a few demonstrations. This setup tests whether a policy can reuse functional interaction knowledge. Based on this principle, we propose **Evo-DexBench**, which controls both source-task relatedness and target-task exposure. It comprises 30 tasks in six functional families and a VR teleoperation pipeline. The resulting dataset covers three everyday scenes per task. It contains 1,248 multimodal demonstrations collected with simulated Shadow Hands equipped with 123 taxels. Evaluating representative policies, we find that the best achieves only 38.5% mean success, and that functionally matched Core data tends to yield larger gains than mismatched data. Results show that existing methods still struggle to extract and reuse functional interaction knowledge.

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.

Evo-DexBench provides an effective evaluation of the generalization ability of general-purpose dexterous manipulation policies across related tasks.

1 INTRODUCTION

General dexterous manipulation requires policies to learn shared manipulation patterns from existing tasks and transfer functional interactions to new tasks with only a few demonstrations (Fu et al., 2026; Tsuji et al., 2025). Recent progress in Vision-Language-Action (VLA) models (Physical Intelligence et al., 2026; Luo et al., 2025; Zheng et al., 2026; Yuan et al., 2026a) and World Action Models (WAMs) (Ye et al., 2026; Li et al., 2026; Liao et al., 2026) makes it possible to learn reusable functional interactions from multi-task demonstrations. Many tasks use different objects and scenes but share the same functional interaction. For example, turning a knob, opening a bottle cap, and twisting a tool all require a stable grasp and proper contact. Success should not come from memorizing specific objects, trajectories, or instructions. Instead, a policy should learn reusable object-use behaviors.

General dexterous manipulation requires policies to abstract and reuse interaction knowledge from related tasks. Yet current evaluations struggle to measure this ability accurately. Dexterous hands have many degrees of freedom and require coordinated finger movements. Task success therefore depends not only on the end-effector trajectory, but also on finger configurations and contact relationships (Zhao et al., 2026; Wang et al., 2026). When the target object changes, the policy must preserve the functional interaction requirements while adapting its finger and arm motions. Collecting many demonstrations for every new task is costly. Transferring interaction knowledge from related tasks is therefore essential for scaling to a broader range of everyday manipulation skills (An et al., 2025; Li et al., 2022). However, current evaluations (Liu et al., 2023; Yao et al., 2026) often make this ability difficult to assess: Too many target-task demonstrations may allow a model to rely on task memorization. If the source and target tasks differ too much, it is difficult to determine whether performance differences result from functional interaction transfer or other factors.

To address these issues, we propose Evo-DexBench, a benchmark for evaluating generalization across related tasks, in addition to robustness to scene variations. Given source tasks that share functional interaction requirements with a target task, Evo-DexBench changes the object instances and scene conditions and gives the policy only a few target demonstrations. By controlling how much the target task is directly exposed, we test whether the policy learns reusable knowledge from related source tasks and whether experience with more related tasks leads to better performance.

Our contributions can be summarized as:

- We introduce Evo-DexBench, which evaluates dexterous policies generalization across related tasks. It follows the Core-Gen design principle: each evaluation unit is a core-task-set–gen-task pair with shared functional interaction. We provide only a few target demonstrations and measure how well a policy transfers interaction knowledge to new tasks.
- We build an integrated simulation environment and a VR teleoperation data collection pipeline. For each task, we create three everyday scenes. We also equip simulated Shadow Hands with 123 taxels per hand and collect 1,248 multimodal demonstrations. These resources provide the data needed for evaluation.
- We systematically evaluate representative models on Evo-DexBench. We compare few-shot adaptation strategies, Core sources of different relatedness and breadth, and the effect of tactile input. The best mean success rate is only 38.5%, and functionally matched Core data tends to yield larger gains than mismatched data. These results show that current methods still struggle to extract and reuse functional interaction knowledge.

2 RELATED WORK

Evaluation for Generalization. Existing manipulation benchmarks mainly assess generalization to unseen environments (Mees et al., 2022; Zhang et al., 2025b; Chen et al., 2025). Others evaluate policies on held-out tasks, such as the ML10/ML45 splits of Meta-World (Yu et al., 2020) and the ML4 split of Bi-DexHands (Chen et al., 2022), or on knowledge-partitioned task suites in

Table 1: Comparison of existing benchmarks.

Benchmark	Arm–hand	Bimanual	Force	Scenes/task	Core–Gen	Data Collection
LIBERO	✗	✗	✗	1	✗	SpaceMouse teleoperation
RoboTwin 2.0	✗	✓	✗	1	✗	MLLM-based generation
Bi-DexHands	✗	✓	✗	1	ML4	RL policy
DexArt	✓	✗	✗	1	✗	RL policy
DexCraft	✓	✗	✗	1	✗	Keyboard teleoperation
HANDFUL	✓	✗	✗	1	✗	RL policy
DexJoCo	✓	✓	✗	1	✗	Motion-capture teleoperation
DexVerse	✓	✓	Optional	1	✗	VR teleoperation
Evo-DexBench (Ours)	✓	✓	123 taxels	3	✓	VR teleoperation

LIBERO (Liu et al., 2023). However, these splits are defined by task identity rather than by shared functional interaction, leaving source–target relatedness uncontrolled, methods then adapt to the target either from rewards alone via meta-reinforcement learning or from abundant demonstrations (50 per task in LIBERO). Where2Act (Mo et al., 2021) and VAT-Mart (Wu et al., 2022) learn reusable interaction structures, but only for individual articulated objects.

Dexterous Manipulation Benchmarks. Table 1 shows that benchmarks for dexterous manipulation cover functional grasping, manipulation of unseen objects and bimanual coordination (Bao et al., 2023; Foong et al., 2026). DexCraft (Yuan et al., 2026b) scales grasp pretraining toward tool-oriented manipulation; DexJoCo (Wang et al., 2026) targets task-oriented functional manipulation; and DexVerse (Yao et al., 2026) emphasizes multi-task, multi-embodiment operation. These efforts extend evaluation beyond single-step grasping, but they typically ask whether a policy can complete predefined tasks, rather than whether contact-level interaction knowledge transfers across tasks.

Dexterous Manipulation Policies. Prior work has advanced vision-based dexterous manipulation and generalization to unseen objects and scenes (Rajeswaran et al., 2017; Andrychowicz et al., 2020; Qi et al., 2023; Xu et al., 2023; Wan et al., 2023; Qin et al., 2023a; Lum et al., 2024), while other studies have scaled up human demonstration data and incorporated tactile information (Qin et al., 2022; Wang et al., 2024; Zhang et al., 2025a). More recent models such as DexVLA (Wen et al., 2025), DexGraspVLA (Zhong et al., 2026), and $\pi_{0.5}$ (Intelligence et al., 2025) seek behavior priors across tasks, objects, and embodiments. Despite this progress toward reusable knowledge, most benchmarks still mainly measure task success and do not systematically control source–target task relatedness. They therefore struggle to distinguish transferable interaction knowledge from target-task memorization. Evo-DexBench addresses this gap by evaluating transfer from multiple related source tasks to a new target task with only a few demonstrations.

3 Benchmark Design

Evo-DexBench targets dexterous manipulation with functional components. It evaluates whether policies can distill transferable knowledge from multiple *Core* tasks that share the same functional interaction. It then assesses whether they can apply that knowledge to *Gen* tasks given only a few demonstrations of the target task. Prior benchmarks mainly test isolated task completion, while our goal is to measure few-shot generalization from multiple related tasks to a novel task.

Section 3.1 introduces the Core–Gen design principle and task composition, Section 3.2 presents the demonstration-collection pipeline, and Section 3.3 details the shared environment, observation, action, and tactile interfaces used for collection and evaluation.

3.1 Design Principles and Task Composition

To systematically assess whether models can learn reusable knowledge from functionally related tasks and transfer it to a target task with only a few target demonstrations, we propose the Core–Gen

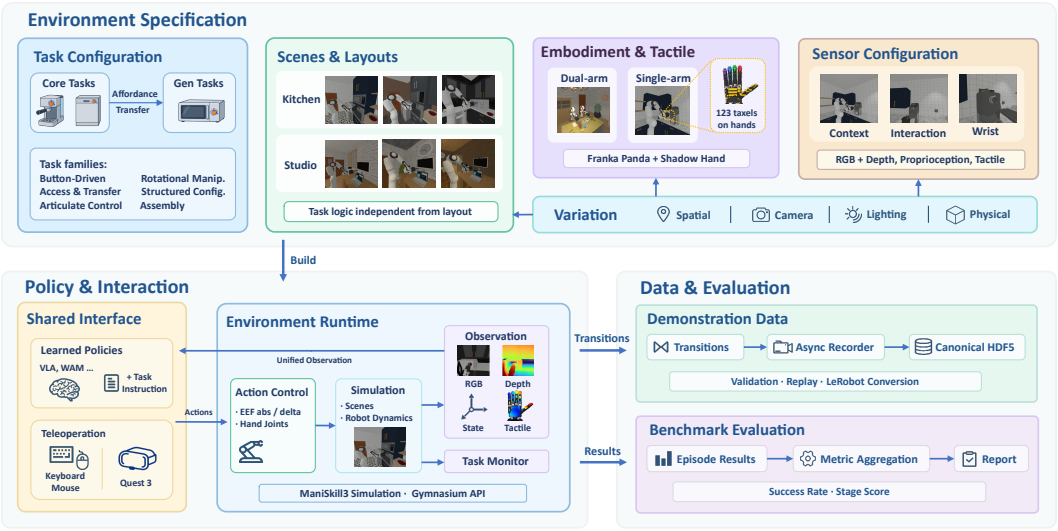


Figure 2: **Overall framework of the benchmark.** The environment covers six task categories, three Kitchen/Studio scenes, 123 hand taxels, and variations in layout, cameras, lighting, and physical properties, while decoupling task logic from scene layout. Policies and keyboard/mouse or Quest 3 teleoperation share a unified interface.

Table 2: Interaction families and shared functional interaction patterns.

Family	Shared interaction
Button-driven	Displace a localized control to change device state.
Rotational manipulation	Apply directed rotation to a functional component.
Access-and-transfer	Open an access mechanism, transfer an object, and restore its state.
Structured configuration	Construct a spatial arrangement under ordering or stability constraints.
Assembly	Align complementary objects with two hands for insertion or fitting.
Articulated configuration control	Stabilize one part while changing another's relative configuration.

design principle. As shown in Figure 3, the evaluation unit in Evo-DexBench is a Core–Gen task pair. Core consists of a set of functionally related interaction tasks. Gen is a target task that shares the functional interaction with Core. For example, Core may include pressing a desk-lamp button and a smartphone home button, while Gen requires transferring this interaction skill to pressing a monitor power button. Because Gen provides only a few demonstrations, a model cannot pass by memorizing the answer. It must extract reusable interaction capabilities from Core and transfer them to Gen.

We group tasks by their primary functional interactions, considering the manipulated components, contact modes, motion constraints, and ordering of actions. For example, a faucet handle and a dishwasher knob are different objects, but both require rotating an actuated component. Formally, each task is defined as

$$\mathcal{T} = (\Omega, \mathcal{S}_0, \mathcal{O}, \mathcal{A}, \mathcal{G}),$$

where Ω denotes the objects and actuated components involved in the task, $\mathcal{S}_0$ the initial environment state, $\mathcal{O}$ and $\mathcal{A}$ the observations available to and actions executed by the policy, $\mathcal{G}$ the task goal. Based on these principles, Evo-DexBench organizes its thirty tasks into six families, each with four Core tasks and one Gen task. Core tasks within a family instantiate the same functional relation across different objects, component locations, and motion conditions, creating diverse source tasks

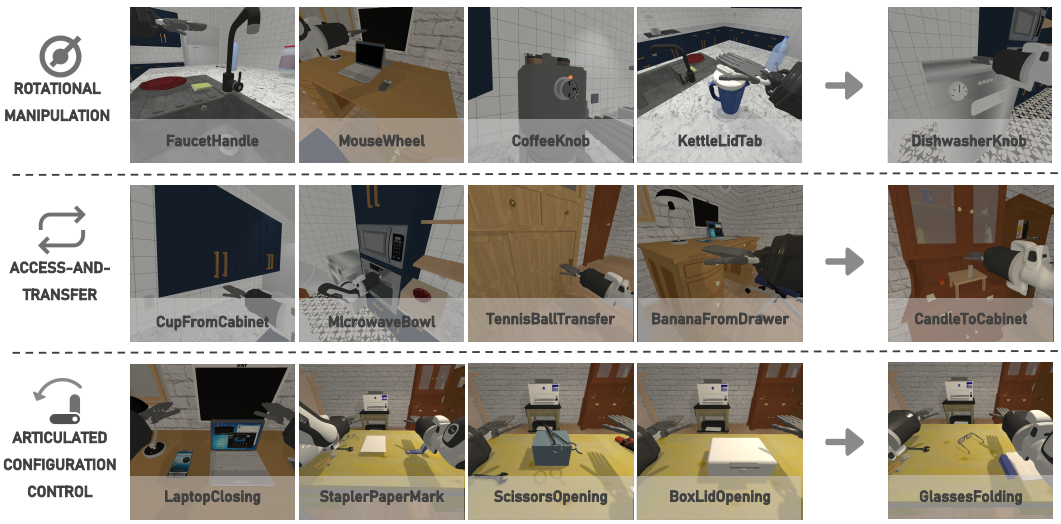


Figure 3: **Core-Gen task examples.** The first and second rows show single-arm rotational manipulation and access-and-transfer tasks, respectively, while the third row shows bimanual articulated configuration control tasks. The first four columns show Core tasks, and the last column shows Gen tasks.

from which a policy can learn stable functional relations rather than rely on a single object or fixed trajectory. Gen extends this relation to a new object or usage context, requiring the policy to transfer prior interaction knowledge to a new task from limited target-task demonstrations. As summarized in Table 2, the six affordance families cover button-driven, rotational manipulation, access-and-transfer, structured configuration, assembly, and Articulated configuration control. The first three use single-arm, single-hand setups, while the last three use bimanual setups. Access-and-transfer and structured configuration tasks are long-horizon.

3.2 Data Collection Pipeline

VR Teleoperation. Evo-DexBench uses Quest 3 WebXR hand tracking to collect dexterous manipulation demonstrations in simulation. We map the operator's wrist motion to the Panda arm end-effector and convert hand keypoints into Shadow Hand joint targets with a dexterous-hand retargeting method (Qin et al., 2023b).

Data Collection. The demonstrations cover thirty tasks and six categories of functional interaction. For each task, demonstrations are collected across three scene layouts selected at random. This preserves task consistency while reducing dependence on any single scene structure. The released dataset contains 1,248 successful demonstration trajectories; each Core task contributes fifty trajectories and each Gen task contributes eight. Figure 4 shows four individual trajectories. The dataset is provided in two formats. Canonical HDF5 is the source format for complete multimodal information. We additionally provide a LeRobot-v3 format for direct policy training.

3.3 Implementation Details

Composable Environment Design. As shown in Figure 2, Evo-DexBench uses a composable environment design, task logic is reusable across layouts; scenes are composable across tasks. Benchmark scenes are organized into two categories, Studio and Kitchen, each with three layouts that vary furniture arrangement, relative object placement, and visual background. Tasks are randomly executed across everyday environments rather than on a fixed workbench, preventing policies from relying on location or appearance cues tied to one scene. Policies must instead understand task requirements and adapt to environmental changes.

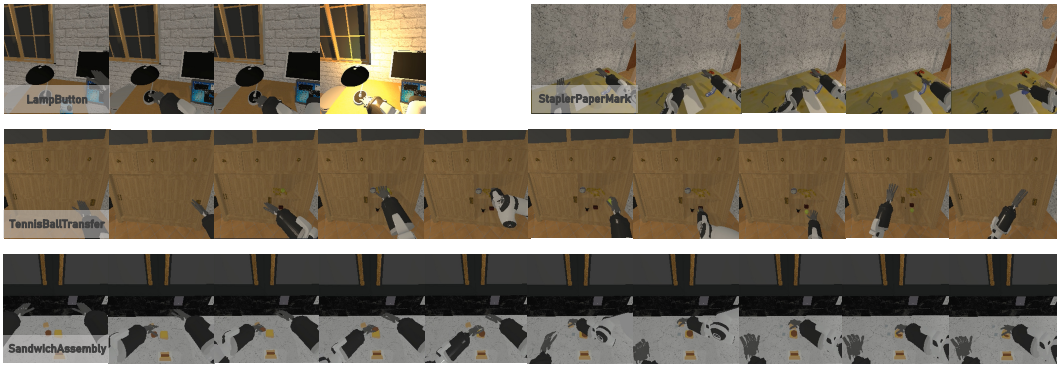


Figure 4: **Task visualizations.** The first column shows short-horizon tasks: pressing a desk lamp button and stapling paper. The second and third columns show long-horizon tasks: opening a cabinet with one hand and placing a tennis ball on the lower shelf, and assembling a sandwich with both hands.

Scene Assets and Visual Observations. Scene assets are organized by functional role. YCB (Calli et al., 2015) provides everyday rigid bodies with relatively consistent scale and appearance. PartNet-Mobility (Xiang et al., 2020) provides articulated devices with realistic joint structures. RoboCasa (Nasiriany et al., 2024) provides semantically meaningful kitchen scenes. Some tasks additionally use custom assets to meet specific appearance or functional requirements. Visual observations come from three camera types. The context camera observes the overall environment, the interaction camera focuses on the task workspace, and the wrist camera provides local detail around the hand. The context and interaction cameras are manually calibrated for each task.

Tactile Design. Tactile sensing uses a unified distributed representation. Each Shadow Hand outputs a 123-dimensional normal-force signal, different fingers receive different numbers of taxels according to contact area and manipulation role. Each taxel reports a nonnegative normal force in newtons.

4 Experiments

In this section, we structure our experiments around the following four research questions: (i) How do existing policy architectures perform on Evo-DexBench? (Sec. 4.2); (ii) How can Core data be most effectively leveraged for few-shot adaptation to Gen tasks? (Sec. 4.3); (iii) Does functionally matched Core data enable better transfer to Gen tasks? (Sec. 4.4); (iv) To what extent does tactile sensing improve manipulation performance? (Sec. 4.5).

4.1 Experimental Setup and Protocol

Scope and training. We conduct experiments on three functionally distinct families: Button-driven manipulation and Rotational manipulation with one hand, and Articulated configuration control with both hands. Each family comprises four Core tasks, with 50 successful demonstrations per task, and one Gen task with eight demonstrations. Following the officially recommended training recipes, we evaluate $\pi_{0.5}$ (Intelligence et al., 2025), convolutional and transformer-based Diffusion Policies (DP-C and DP-T) (Chi et al., 2024), the tactile-aware KineDex policy (Zhang et al., 2025a), and the compact SmolVLA policy (Shukor et al., 2025) under a unified benchmark interface. Detailed model-specific adaptations and optimization settings are given in Appendix D.

Evaluation. Each task is evaluated over 100 episodes, consisting of 20 nominal episodes and 20 episodes for each of four single-factor variation categories: camera, lighting, physical, and spatial variations. The three predefined scene layouts are evaluated with 34, 33, and 33 episodes, respectively. For each task, all compared policies share the same registered reset schedule to ensure identical evaluation conditions. Model weights remain frozen throughout evaluation.

Metrics. Let $y_e \in \{0, 1\}$ denote the final task success of episode e, and let $k_e \in \{0, \ldots, K_t\}$ denote the highest consecutively completed stage for task t, where K_t is the total number of stages. We report the success rate (SR) and normalized stage score:

$$\mathrm{SR} = \frac{1}{N}\sum_{e=1}^{N} y_e, \qquad \mathrm{StageScore} = \frac{1}{N}\sum_{e=1}^{N} \frac{k_e}{K_t}. \tag{1}$$

StageScore assigns credit only to consecutively completed stages in the prescribed order and reaches one only when all stages are completed. Unless otherwise specified, we macro-average results across tasks so that each task contributes equally to the aggregate metric.

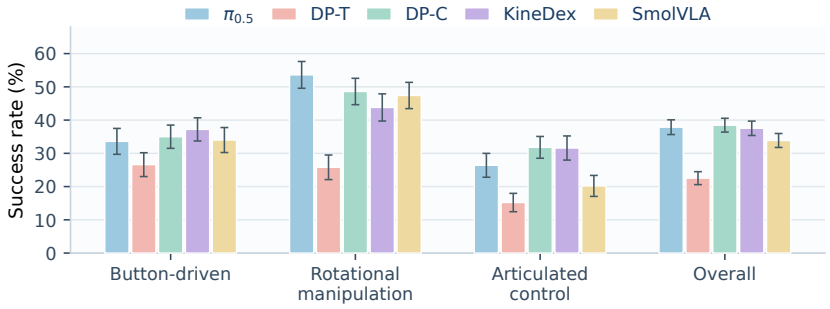


Figure 5: Family-level baseline SR for the evaluated policies, with 95% confidence intervals.

4.2 Baseline Policy Performance

Table 3 summarizes performance across all 15 tasks under the standard evaluation protocol. No single policy dominates across task families. DP-C, $\pi_{0.5}$, and KineDex form a closely matched leading group in aggregate performance, while SmolVLA achieves competitive but lower overall results and DP-T trails behind.

Table 3: Standard evaluation on 15 tasks. Each entry is **SR (%) / StageScore**.

Family	Task	$\pi_{0.5}$	DP-T	DP-C	KineDex	SmolVLA
Button-driven	LampButton	30% / 35.7	15% / 12.6	26% / 28.7	24% / 23.2	**41% / 35.2**
	WallSwitch	60% / 74.1	35% / 57.8	68% / 79.7	**76% / 85.6**	66% / 78.0
	PhoneHomeButton	**16% / 24.4**	1% / 2.2	8% / 13.7	5% / 14.8	8% / 12.8
	ToasterSlider	42% / 75.2	49% / 79.6	**64% / 84.0**	63% / 84.0	37% / 74.2
	DisplayPowerButton	20% / 39.4	**33% / 52.8**	9% / 31.5	18% / 40.8	18% / 36.0
Rotational manipulation	FaucetHandle	21% / **29.1**	**24%** / 27.2	19% / 23.0	16% / 22.0	12% / 22.2
	MouseWheel	**63% / 80.8**	37% / 65.0	47% / 69.0	53% / 72.5	62% / 77.8
	CoffeeKnob	**59% / 82.9**	11% / 49.3	33% / 71.2	33% / 68.9	45% / 72.6
	KettleLidTab	**79% / 85.2**	16% / 51.7	76% / 81.5	62% / 75.3	76% / 84.1
	DishwasherKnob	46% / 62.5	41% / 56.6	**68% / 76.7**	55% / 67.9	42% / 57.3
Articulated control	LaptopClosing	47% / 60.5	49% / 59.2	**85% / 91.3**	69% / 75.9	53% / 59.9
	StaplerPaperMark	**42% / 60.9**	2% / 26.9	24% / 50.6	33% / 54.7	12% / 37.0
	ScissorsOpening	23% / 55.9	6% / 45.4	**29% / 58.6**	26% / 56.8	15% / 49.9
	BoxLidOpening	18% / 41.1	13% / 39.1	15% / 39.6	**26% / 46.4**	21% / 40.8
	GlassesFolding	2% / 6.0	**6%** / 16.8	**6% / 20.4**	4% / 16.4	0% / 5.2
Task mean		37.9% / 54.2	22.5% / 42.8	**38.5% / 54.6**	37.5% / 53.7	33.9% / 49.5

Notably, large-scale pretrained model $\pi_{0.5}$ does not exhibit a clear aggregate advantage over other policies. While such pretrained policies have shown strong performance on conventional gripper-based manipulation, their advantage may transfer only partially to dexterous-hand control, where the action space, contact patterns, and manipulation strategies differ substantially from those encountered during gripper-centric pretraining.

The family-level breakdown in Figure 5 further reveals complementary strengths across policy architectures. KineDex performs best on Button-driven manipulation, $\pi_{0.5}$ on Rotational manipulation, and DP-C on Articulated configuration control.

Gen tasks are generally more challenging than their corresponding Core tasks. With only eight demonstrations per Gen task, most model–family combinations achieve lower performance on Gen than on the more densely demonstrated Core tasks. This gap motivates the following experiments on how related Core demonstrations can be leveraged for few-shot Gen adaptation.

4.3 Few-Shot Adaptation with Core Data

A central question in few-shot adaptation is how demonstrations from related manipulation tasks should be incorporated when only a small number of target-task examples are available. We therefore study whether Core demonstrations can serve as useful post-training data for adapting to Gen tasks with only eight target demonstrations. We compare four strategies:

- *Gen only*: trains exclusively on the eight Gen demonstrations;
- *Full Core → Gen*: first trains on Core data to convergence and then fine-tunes on Gen data;
- *Early Core → Gen*: fine-tunes on Gen data from an earlier Core checkpoint;
- *Core + Gen joint*: jointly samples Core and Gen data throughout training.

These settings isolate whether related source-task data is most useful as initialization or when retained continuously during target adaptation.

Table 4: Few-shot adaptation strategies at 50 epochs. Each entry is **SR (%) / StageScore**.

Strategy	DisplayPowerButton	DishwasherKnob
Gen only	15% / 29.7	36% / 55.1
Full Core → Gen	8% / 23.6	8% / 29.3
Early Core → Gen	9% / 24.6	22% / 41.2
Core + Gen joint	**20% / 39.4**	**46% / 62.5**

Joint Core + Gen training performs best on both targets, while sequential fine-tuning from Core checkpoints provides limited or negative transfer. These results suggest that, under limited target supervision, retaining access to related Core data during adaptation is more effective than using it only for initialization. We therefore use joint Core+Gen training in the following experiments.

4.4 Functional Transfer across Task Families

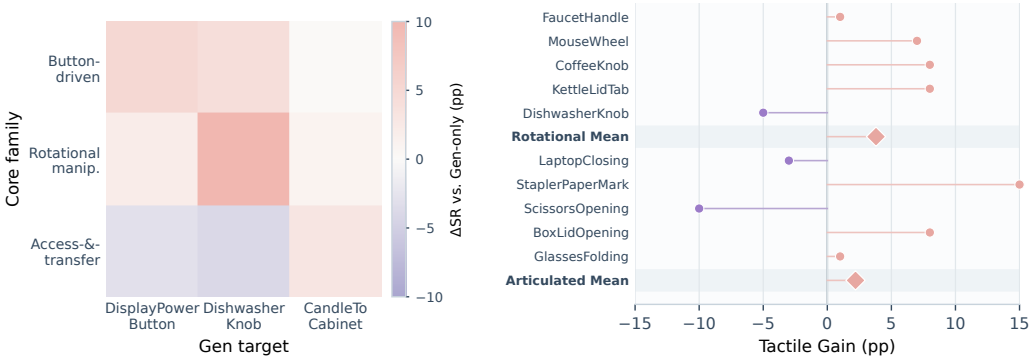


Figure 6: **Transfer heatmap and tactile analysis. Left:** Change in $\pi_{0.5}$ SR relative to Gen-only training when mixing each Core family with a Gen target. The diagonal contains the functionally matched pairs. **Right:** Task-wise SR change from adding tactile observations to KineDex.

We first examine whether functional relatedness between source and target tasks predicts transfer quality. Keeping the Gen demonstrations, Core-Gen sampling, and training protocol fixed, we pair

each Gen target with Core data from different task families. Figure 6 (left) reports the change in $\pi_{0.5}$ success rate relative to Gen-only training. The largest gains consistently occur along the diagonal, where the Core and Gen tasks belong to the same functional family, indicating that functionally matched Core data provides more effective support for few-shot Gen adaptation than unrelated source data.

We further test whether this effect persists across policy architectures. On the DishwasherKnob task, we repeat the source-family comparison with $\pi_{0.5}$, DP-T, and KineDex (Table 5). Across all three models, the matched Rotational source provides the strongest overall transfer, whereas mismatched Core families yield smaller or less consistent gains. This cross-model agreement suggests that the benefit of functional matching is not specific to a particular policy architecture and supports the use of functional task families as a meaningful structure for transfer in Evo-DexBench.

Table 5: Core-source relatedness on DishwasherKnob across three models. Each entry is **SR (%) / StageScore**; the functionally matched Rotational source is bold.

Method	Gen only	Button-driven	Rotational manip.	Access-and-transfer
$\pi_{0.5}$	36% / 55.1	40% / 61.0	**46% / 62.5**	32% / 52.4
DP-T	28% / 42.1	27% / 51.0	**41% / 56.6**	**41%** / 56.0
KineDex	42% / 57.9	39% / 48.9	**55% / 67.9**	51% / 63.9

Finally, we vary the breadth and composition of the matched Core data. Using all four Core tasks generally provides the strongest transfer, outperforming narrower subsets on Gen tasks. Overall, these results suggest that broader coverage within a matched functional family is beneficial, while the specific choice of source tasks can still influence transfer quality. Complete results are provided in Appendix Table 17.

4.5 Tactile Sensing Ablation

Following the official KineDex tactile-ablation protocol, we evaluate the contribution of the complete tactile-conditioned control pathway. As in KineDex, the *No-Touch* variant removes the tactile observations and corresponding model branch, while keeping the demonstrations, visual and proprioceptive inputs, backbone, optimization budget, and evaluation protocol unchanged. We evaluate both variants on the ten tasks from the Rotational and Articulated families, where physical contact plays an important role in task execution.

As shown in Figure 6 (right), under this protocol, adding tactile sensing improves the success rate on seven of the ten tasks, yielding positive mean gains for both task families. The benefit is particularly pronounced on contact-rich tasks, with the largest improvement observed on StaplerPaperMark. These results indicate that the tactile signals provided by Evo-DexBench can be effectively exploited by tactile-aware policies, especially when successful manipulation depends on sustained or precise physical interaction. The variation across tasks also suggests that the utility of touch is task-dependent rather than uniform.

5 Discussion and Conclusion

We introduce Evo-DexBench, which uses Core-Gen task pairs sharing functional interaction requirements as evaluation units. It systematically assesses how well dexterous manipulation policies transfer to new objects and tasks. We evaluate representative models and find that current methods remain limited in their ability to extract and reuse functional interaction knowledge. Evo-DexBench provides a new benchmark for the design and evaluation of general-purpose dexterous manipulation policies.

Limitations and Future Work. Current evaluations are primarily based on a specific robot embodiment and do not systematically examine cross-embodiment transfer. Future work could evaluate policies across multiple robot embodiments to further assess their generalization capability.

AI USE STATEMENT

AI tools were used to assist with correcting grammar and syntax and polishing the language of the manuscript. All AI-assisted content was verified by the authors. The authors take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, Appendix A describes the complete benchmark specifications and task definitions; Appendix B presents the scene layouts and generalization protocol; Appendix C details the 123-taxel tactile system; Appendix D provides the data, training, and evaluation settings; and Appendix E reports the complete experimental results and supplementary analyses. In addition, we will release the code, datasets and model weights to support reproduction and future research.

REFERENCES

Shan An, Ziyu Meng, Chao Tang, Yuning Zhou, Tengyu Liu, Fangqiang Ding, Shufang Zhang, Yao Mu, Ran Song, Wei Zhang, et al. Dexterous manipulation through imitation learning: A survey. *IEEE Transactions on Automation Science and Engineering*, 23:1760–1792, 2025.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Chen Bao, Helin Xu, Yuzhe Qin, and Xiaolong Wang. Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21190–21200, 2023.

Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pp. 510–517. IEEE, 2015.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

Ethan Foong, Yunshuang Li, Hao Jiang, Gaurav S Sukhatme, and Daniel Seita. Handful: Sequential grasp-conditioned dexterous manipulation with resource awareness. *arXiv preprint arXiv:2604.25126*, 2026.

Jiayun Fu, Haotian Huang, Zhehao Jin, Andong Liu, Wen-An Zhang, Li Yu, Weiyong Si, and Chenguang Yang. A survey on learning an autonomous dynamic system for human–robot skills transfer from demonstration. *Robotics and Computer-Integrated Manufacturing*, 97:103092, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a

vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Yinlin Li, Peng Wang, Rui Li, Mo Tao, Zhiyong Liu, and Hong Qiao. A survey of multifingered robotic manipulation: Biological results, structural evolvements, and learning methods. *Frontiers in neurorobotics*, 16:843267, 2022.

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Si Liu, Jianlan Luo, Liliang Chen, et al. Genie envisioner: A unified world foundation platform for robotic manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 88446–88463, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Tyler Ga Wei Lum, Martin Matak, Viktor Makoviychuk, Ankur Handa, Arthur Allshire, Tucker Hermans, Nathan D Ratliff, and Karl Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. *arXiv preprint arXiv:2407.02274*, 2024.

Hao Luo, Yicheng Feng, Wanpeng Zhang, Sipeng Zheng, Ye Wang, Haoqi Yuan, Jiazheng Liu, Chaoyi Xu, Qin Jin, and Zongqing Lu. Being-h0: vision-language-action pretraining from large-scale human videos. *arXiv preprint arXiv:2507.15597*, 2025.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

Kaichun Mo, Leonidas Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *International Conference on Computer Vision (ICCV)*, 2021.

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. *arXiv preprint arXiv:2406.02523*, 2024.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: A steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026. URL https://arxiv.org/abs/2604.15483.

Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning*, pp. 2549–2564. PMLR, 2023.

Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, pp. 570–587. Springer, 2022.

Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *Conference on Robot Learning*, pp. 594–605. PMLR, 2023a.

Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023b.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

Toshiaki Tsuji, Yasuhiro Kato, Gokhan Solak, Heng Zhang, Tadej Petrič, Francesco Nori, and Arash Ajoudani. A survey on imitation learning for contact-rich tasks in robotics. *The International Journal of Robotics Research*, pp. 02783649261417694, 2025.

Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3868–3879. IEEE, 2023.

Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

Hanwen Wang, Weizhi Zhao, Xiangyu Wang, Siyuan Huang, He Lin, Boyuan Zheng, Rongtao Xu, Gang Wang, Yao Mu, He Wang, et al. Dexjoco: A benchmark and toolkit for task-oriented dexterous manipulation on mujoco. *arXiv preprint arXiv:2605.16257*, 2026.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

Ruihai Wu, Yan Zhao, Kaichun Mo, Zizheng Guo, Yian Wang, Tianhao Wu, Qingnan Fan, Xuelin Chen, Leonidas Guibas, and Hao Dong. VAT-mart: Learning visual action trajectory proposals for manipulating 3d ARTiculated objects. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=iEx3PiooLy.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11094–11104. IEEE, 2020.

Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4737–4746. IEEE, 2023.

Yunchao Yao, Zhuxiu Xu, Tianqi Zhang, Zixian Liu, Sikai Li, Zhenyu Wei, Feng Chen, Dihong Huang, Kechang Wan, Chenyang Ma, et al. Dexverse: A modular benchmark for multi-task, multi-embodiment dexterous manipulation. *arXiv preprint arXiv:2607.08751*, 2026.

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

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, et al. Qwen-robotmanip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint arXiv:2606.17846*, 2026a.

Ying Yuan, Xinyu Liu, Sriram Krishna, and David Held. From grasps to dexterity: Large-scale grasp pretraining for dexterous manipulation. *arXiv preprint arXiv:2606.30749*, 2026b.

Di Zhang, Chengbo Yuan, Chuan Wen, Hai Zhang, Junqiao Zhao, and Yang Gao. Kinedex: Learning tactile-informed visuomotor policies via kinesthetic teaching for dexterous manipulation, 2025a. URL https://arxiv.org/abs/2505.01974.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11142–11152. IEEE, 2025b.

Weiguang Zhao, Tian Liang, Xihao Guo, Rui Zhang, Irwin King, and Kaizhu Huang. Towards robotic dexterous hand intelligence: A survey. *arXiv preprint arXiv:2605.13925*, 2026.

Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, et al. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

Yifan Zhong, Xuchuan Huang, Ruochong Li, Ceyao Zhang, Zhang Chen, Tianrui Guan, Fanlian Zeng, Ka Nam Lui, Yuyao Ye, Yitao Liang, et al. Dexgraspvla: A vision-language-action framework towards general dexterous grasping. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18836–18844, 2026.

A Complete Benchmark Specification

Evo-DexBench contains 30 tasks arranged into six functional families. Each family contains four *Core* tasks with 50 successful demonstrations per task and one *Gen* task with eight demonstrations. The Core tasks expose repeated instances of a shared functional interaction, whereas the Gen task requires the same interaction in a new task context. The first three families use single Franka Panda–Shadow Hand system and the last three use dual-arm systems. Tables 6–11 specify the natural-language task description and ordered partial-credit stages for every task; each 100% entry gives the complete terminal success criterion. Names in the tables are the canonical task names used throughout the paper; development identifiers are omitted because they are not part of the public task semantics.

Stage thresholds are defined from simulator state and contact measurements, not from policy observations. A stage is credited only after every preceding stage has been reached in the prescribed order. Thus, the final stage is equivalent to task success, while earlier stages distinguish meaningful progress from arbitrary motion, StageScore is not necessarily greater than or equal to SR. Criteria that require a condition to be held for multiple control steps suppress transient contacts and one-frame threshold crossings.

Table 6: Task definitions for the Button-driven family.

Task	Description	Stage success conditions
Lamp Button	Press the lamp button and keep the lamp on.	30%: index-finger force $\geq$0.15 N; 60%: button travel $\geq$70%; 100%: lamp remains lit for $\geq$1 s after triggering.
WallSwitch	Flip the wall switch on.	30%: hand tip within 0.20 m; 60%: angle $\geq$0.061 rad; 100%: switch angle $\geq$0.122 rad for 1 s.
Phone Home Button	Press and release the phone Home button.	30%: correct contact force $\geq$0.12 N; 70%: contact held while travel $\geq$70%; 100%: button released, return travel $\leq$20%, and phone remains safely on the table.
Toaster Slider	Press the toaster slider.	30%: hand tip within 0.40 m of toaster; 60%: hand tip within 0.35 m of slider point; 100%: toaster slider displacement $\geq$0.062 m.
Display Power Button	Press the display power button.	30%: fingertip within 0.05 m; 60%: contact force $\geq$0.15 N; 100%: after contact is recorded, press travel $\geq$70% for 3 control steps.

Table 7: Task definitions for the Rotational manipulation family.

Task	Description	Stage success conditions
Faucet Handle	Turn the faucet handle open.	30%: handle contact force $\geq$0.40 N; 60%: opening travel $\geq$50%; 100%: opening travel $\geq$80%.
Mouse Wheel	Hold the mouse and scroll its wheel.	25%: mouse contact force $\geq$0.15 N or body displacement > 0.03 m; 50%: wheel contact force $\geq$0.15 N or rotation > 0.03 rad; 100%: wheel rotation $\geq$1.5 rad, mouse linear speed $\leq$0.04 m/s, and angular speed $\leq$0.60 rad/s.

Continued on the next page

Table 7 continued from the previous page

Task	Description	Stage success conditions
Coffee Knob	Turn the coffee-machine knob to setting 2.	30%: knob contact force $\geq$0.40 N; 50%: contact held while turning toward target at $\geq$0.03 rad/s; 75%: angle error $\leq$20°; 100%: angle error $\leq$6°, knob angular speed $\leq$0.20 rad/s, machine displacement $\leq$0.025 m, tilt $\leq$12°, linear speed $\leq$0.05 m/s, and angular speed $\leq$0.60 rad/s.
KettleLid Tab	Rotate the kettle-lid tab to the notch.	30%: thumb and index finger contact, each $\geq$0.35 N; 55%: positive rotation $\geq$5° at $\geq$0.03 rad/s; 80%: angle in 60°–90°; 100%: angle error $\leq$9°, tab angular speed $\leq$0.50 rad/s, kettle displacement $\leq$0.01 m, and tilt $\leq$3°.
Dishwasher Knob	Turn the dishwasher knob to its target setting.	30%: knob contact force $\geq$0.40 N; 60%: angle error $\leq$45°; 100%: effective turn recorded, angle error $\leq$9°, and angular speed $\leq$1.00 rad/s.

Table 8: Task definitions for the Access-and-transfer family.

Task	Description	Stage success conditions
CupFrom Cabinet	Open the cabinet, remove the cup, and place it upright on the counter.	25%: door angle $>$ 45°; 50%: valid cup-body/handle grasp (force $\geq$0.40 N); 75%: cup clears the cabinet opening; 100%: cup is on the counter, upright, released, with linear speed $\leq$0.05 m/s and angular speed $\leq$0.60 rad/s for 0.40 s.
Microwave Bowl	Open the microwave, place the bowl inside, and close the door.	25%: door opening $\geq$45%; 50%: two-finger bowl grasp held 0.12 s; 75%: bowl fully inside, supported, tilt $\leq$35°, released and still for 0.35 s; 100%: bowl remains fully inside, supported, released, and stable while door opening $\leq$5% and door speed $\leq$0.08 rad/s for 0.30 s.
TennisBall Transfer	Move the tennis ball from shelf 1 to shelf 2 and close the cabinet.	20%: door opening $\geq$55%; 45%: two-finger grasp with force $\geq$0.40 N per finger; 75%: ball in shelf-2 target, supported, released, linear speed $\leq$0.03 m/s, angular speed $\leq$0.60 rad/s for 5 steps; 100%: ball remains stable on shelf 2 while door opening $\leq$5% and door speed $\leq$0.01 m/s for 5 steps.
Banana From Drawer	Open the drawer, place the banana on the desk, and close the drawer.	25%: drawer opening $\geq$45% (exits below 40%); 40%: valid grasp held 0.15 s; 60%: banana fully clears drawer; 85%: desk support $\geq$0.20 N, released and still for 0.50 s; 100%: banana remains released and stable on the desk while drawer opening $\leq$5% and drawer speed $\leq$0.01 m/s for 0.15 s.

Continued on the next page

Table 8 continued from the previous page

Task	Description	Stage success conditions
CandleTo Cabinet	Open the upper glass cabinet, place the candle inside, and release it.	25%: an upper door angle $> 60°$; 50%: at least two fingers, each $\geq$0.30 N; 75%: candle bounding box inside cabinet; 100%: candle remains inside and upright with base-height error $\leq$0.025 m, released, tilt $\leq 20°$, linear speed $\leq$0.03 m/s, and angular speed $\leq$0.50 rad/s for 1 s.

Table 9: Task definitions for the Structured configuration family.

Task	Description	Stage success conditions
Tableware Placement	Place the plate in the center, fork left, and knife right.	30%: plate within 0.12 m, tilt $\leq 35°$, released and still; 60%: fork meets the same limits, yaw error $\leq 35°$, left of plate; 90%: knife meets the same limits and is right of plate; 100%: all three pieces have valid poses, left/right relations, clearance, release, and stability for 1 s.
Hammer Block Assembly	Arrange three blocks into the reference hammer shape.	25%: one block position/height error $\leq$0.015 m, yaw/tilt $\leq 8°$, released; 50%: two blocks also adjacent (center error $\leq$0.020 m, edge angle $\leq 12°$); 75%: all three target poses and releases valid; 100%: footprint IoU $\geq$0.90, adjacency valid, all blocks released, linear speed $\leq$0.03 m/s, and angular speed $\leq$0.50 rad/s for 0.50 s.
Sandwich Assembly	Stack the bread, cheese, patty, and top bread.	20%: bottom bread position error $\leq$0.050 m, height error $\leq$0.018 m, tilt $\leq 25°$, stable; 40%: cheese supported by bottom bread; 60%: patty supported in order; 100%: all four layers are supported in order, released, aligned with valid overlaps, and within the speed limits for 0.25 s.
RingTower Assembly	Stack the red, blue, and yellow rings from largest to smallest.	25%: red ring on bottom; 50%: blue ring above red; 75%: yellow ring on top; each ring center within 0.028 m of post axis, height error $\leq$0.018 m, tilt $\leq 22°$; 100%: correct three-ring order and poses, with linear speed $\leq$0.06 m/s and angular speed $\leq$0.80 rad/s for 20 steps.
Box Packing	Pack three shaped objects into the box and close the lid.	20%: at least one object fully inside; 40%: at least two; 75%: all three; 100%: all three objects fully inside and lid opening $\leq 4\%$.

Table 10: Task definitions for the Assembly family.

Task	Description	Stage success conditions
Pen Capping	Remove the pen, grasp its cap, and cap the pen.	20%: pen contact force $\geq$0.15 N; 40%: cap contact force $\geq$0.15 N; 70%: axis error $\leq$18°, lateral error $\leq$0.015 m, tip inside cap range; 100%: cap aligned with the pen axis, lateral error $\leq$0.010 m, and insertion depth $\geq$0.015 m.
CupSleeve Fit	Slide a sleeve from the cup bottom to its middle.	20%: cup contact force $\geq$0.18 N; 40%: sleeve contact force $\geq$0.18 N; 70%: axis error $\leq$15°, lateral error $\leq$0.018$\times$scale, center below 0.040$\times$scale, opposite-hand roles; 100%: sleeve entered from below, center height 0.062–0.100$\times$scale, lateral error $\leq$0.010$\times$scale, and axis error $\leq$15°.
Bottle Funnel Insertion	Align a funnel with a bottle and insert it.	20%: bottle contact force $\geq$0.18 N; 40%: funnel contact force $\geq$0.18 N; 70%: axis error $\leq$15°, lateral error $\leq$0.018 m, tip near mouth; 100%: insertion depth $\geq$0.010 m, lateral error $\leq$0.010 m, relative linear speed $\leq$0.06 m/s, and angular speed $\leq$1.00 rad/s for 3 steps.
Plug Insertion	Hold a plug and socket and fully insert the plug.	25%: opposite hands hold both objects, at least two parent contacts per object at $\geq$0.20 N; 50%: pin-tip lateral error $\leq$0.012375 m; 75%: aligned pins enter socket; 100%: pins remain aligned, insertion depth $\geq$0.01575 m, and plug linear speed $\leq$0.15 m/s.
Scissors Sheathing	Insert closed scissors fully into the sheath.	75%: tip aligned in sheath (lateral error $\leq$0.075 m); 100%: tip lateral error $\leq$0.075 m and insertion depth $\geq$0.035 m.

Table 11: Task definitions for the Articulated configuration control family.

Task	Description	Stage success conditions
Laptop Closing	Stabilize the laptop base and close the screen to about 25°.	30%: opposite hands contact base/screen, base displacement $\leq$0.040 m and tilt $\leq$5°; 60%: screen reaches 60°$\pm$8°; 100%: screen reaches 25°$\pm$5°, base remains stable, and screen closing speed $\leq$1.00 rad/s for 0.25 s.
Stapler PaperMark	Insert paper, press the stapler, then move it away.	20%: paper force $\geq$0.50 N or displacement $\geq$0.03 m; 50%: paper enters staple area; 70%: paper remains in area, stapler press arm at 0$\pm$0.10 rad, mark created; 100%: after a staple mark is created, the stapler remains $\geq$0.06 m clear of the staple area for 5 steps.
Scissors Opening	Stabilize scissors and open the moving handle.	40%: support-handle contact $\geq$0.15 N or stable low scissors-base speed; 70%: moving-handle contact $\geq$0.15 N or angle error $\leq$0.50 rad; 100%: opening angle 1.263$\pm$0.20 rad, scissors-base linear speed $\leq$0.30 m/s, and angular speed $\leq$6.00 rad/s for 10 steps.

Continued on the next page

Table 11 continued from the previous page

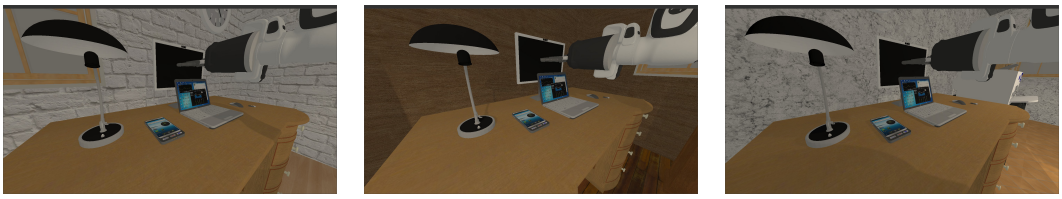
Task	Description	Stage success conditions
BoxLid Opening	Stabilize a box and open its hinged lid.	30%: task contact and box displacement $\leq$0.10 m, tilt $\leq$7°, linear speed $\leq$0.05 m/s, angular speed $\leq$0.50 rad/s; 60%: stable box, opening motion, lid angle $\geq$10°; 100%: lid angle 90°–105°, box stable, and lid angular speed $\leq$0.25 rad/s for 1 s.
Glasses Folding	Fold both eyeglass temples into the storage position.	40%: left temple reaches about 60% travel; 80%: right temple reaches about 60% travel; 100%: both temples reach one valid folding-order target pair (about 89.4% on one side and 45.1% on the other, with 0.05 rad tolerance).

B Scene Layouts and Generalization Protocol

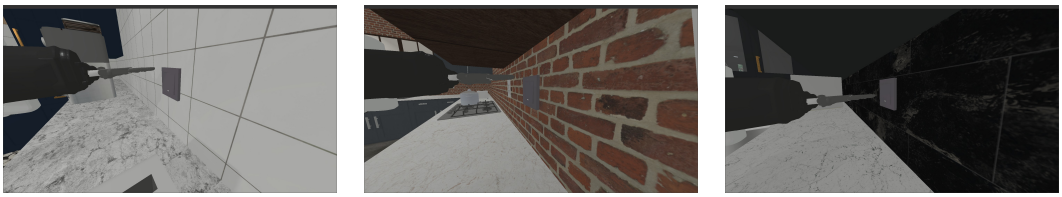
B.1 Scene-layout variation

Each task is instantiated in three independently constructed scene layouts. These layouts preserve the same task semantics and success criteria while varying the visual context, object arrangement, support geometry, and overall scene composition to reduce dependence on a single canonical scene; they are not intended to form a held-out scene-generalization split. This design separates repeatable scene-level diversity from the continuous single-axis perturbations described below. The panels below show all 90 task–layout combinations. Each task occupies one row, with its three layouts ordered from left to right and its canonical name centered below.

Button-driven



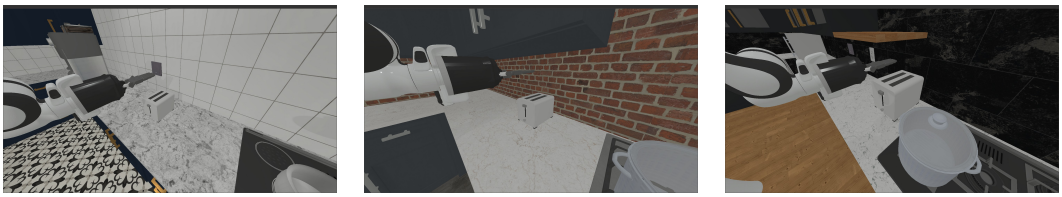
`LampButton` *(Core)*



`WallSwitch` *(Core)*



`PhoneHomeButton` *(Core)*

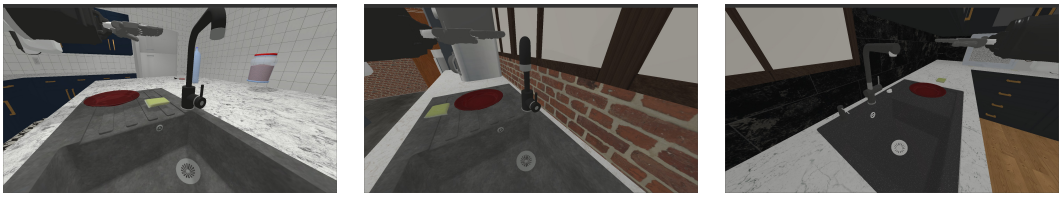


ToasterSlider *(Core)*



DisplayPowerButton *(Gen)*

Rotational manipulation



FaucetHandle *(Core)*



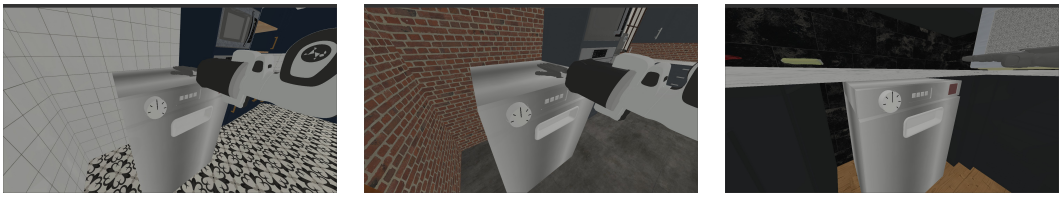
MouseWheel *(Core)*



CoffeeKnob *(Core)*

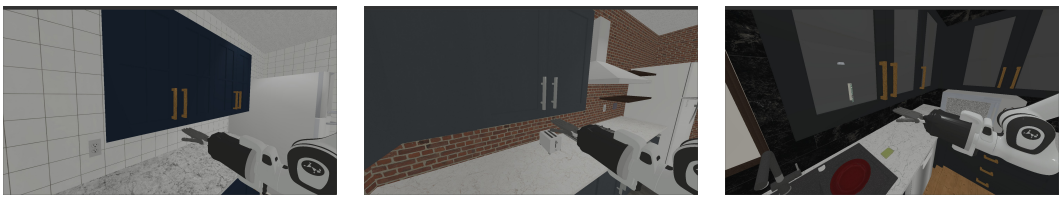


KettleLidTab *(Core)*

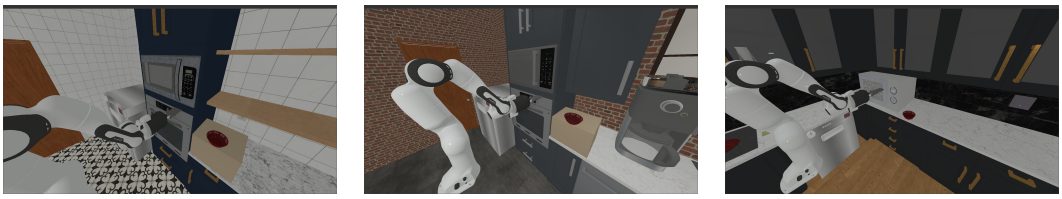


DishwasherKnob *(Gen)*

Access-and-transfer



`CupFromCabinet` *(Core)*



`MicrowaveBowl` *(Core)*



`TennisBallTransfer` *(Core)*

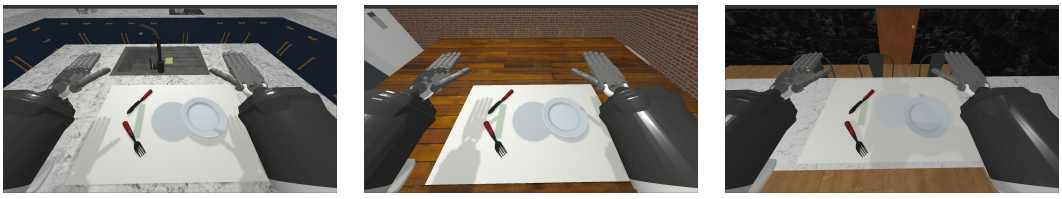


`BananaFromDrawer` *(Core)*



`CandleToCabinet` *(Gen)*

Structured configuration



`TablewarePlacement` *(Core)*



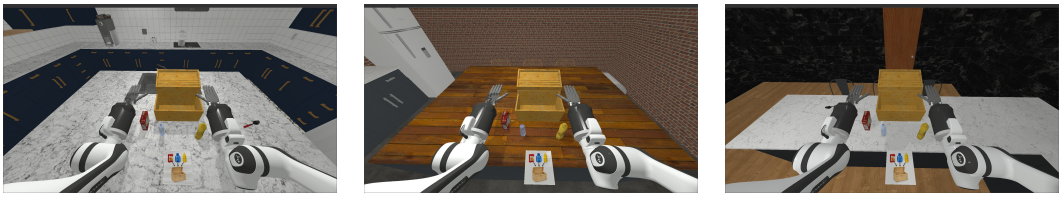
HammerBlockAssembly *(Core)*



SandwichAssembly *(Core)*

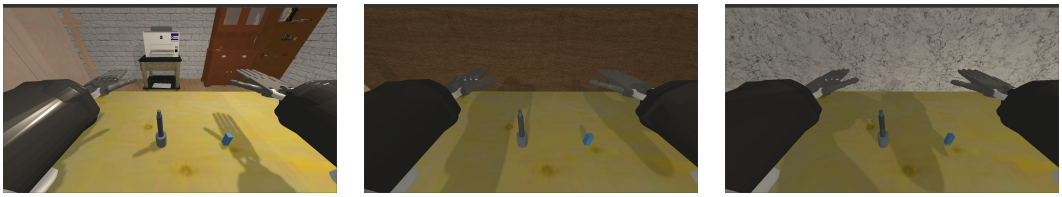


RingTowerAssembly *(Core)*



BoxPacking *(Gen)*

Assembly



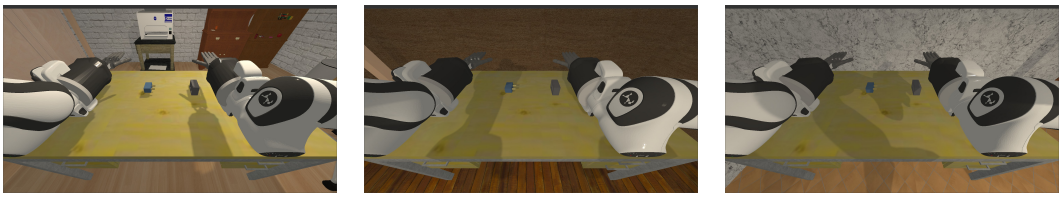
PenCapping *(Core)*



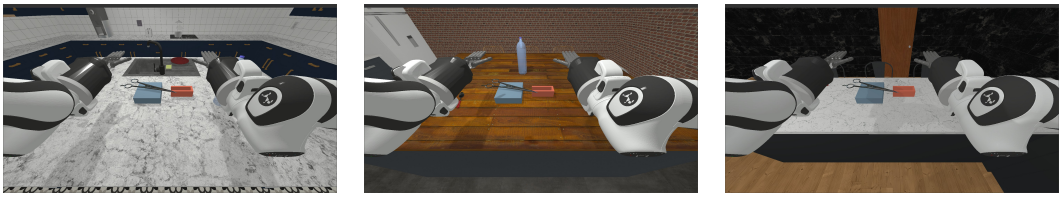
CupSleeveFit *(Core)*



BottleFunnelInsertion *(Core)*



PlugInsertion *(Core)*



ScissorsSheathing *(Gen)*

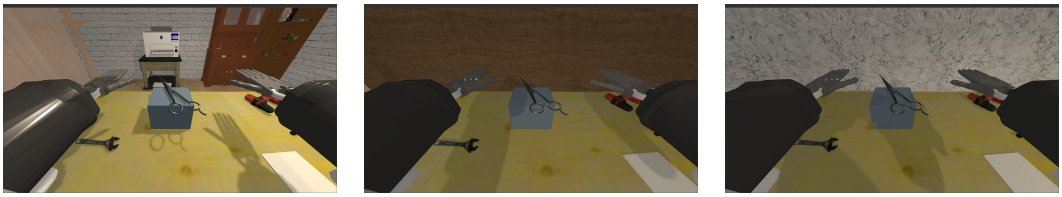
Articulated configuration control



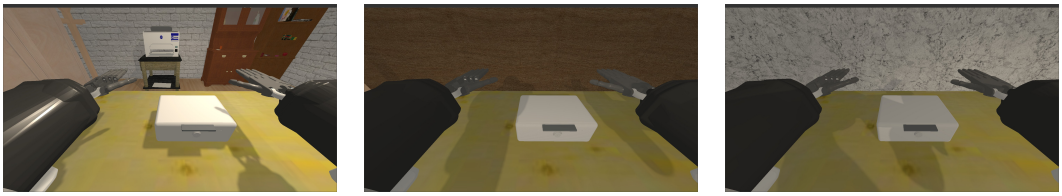
LaptopClosing *(Core)*



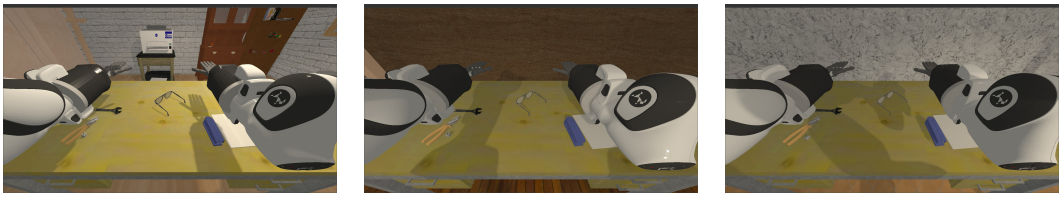
StaplerPaperMark *(Core)*



ScissorsOpening *(Core)*



BoxLidOpening *(Core)*



GlassesFolding *(Gen)*

B.2 GENERALIZATION AXES

The evaluation protocol varies one factor at a time. A nominal episode uses the registered layout without continuous perturbation. Each generalized episode activates exactly one of the following axes while holding the others nominal:

Camera. The context camera is perturbed by up to $\pm 7.5^\circ$ in azimuth, $\pm 5^\circ$ in elevation, $\pm 10\%$ in viewing distance, ± 0.05 m independently in each target coordinate, and $\pm 5^\circ$ in field of view. The interaction camera uses the corresponding limits $\pm 5^\circ$, $\pm 4^\circ$, $\pm 8\%$, ± 0.03 m, and $\pm 5^\circ$. Wrist cameras remain rigidly attached to the wrists and are not independently randomized.

Lighting. Ambient intensity is scaled in $[0.70, 1.30]$ and direct-light intensity in $[0.75, 1.25]$. A color parameter in $[-0.08, 0.08]$ applies coupled exponential red/blue gains while preserving non-negative RGB values and the nominal light geometry.

Physical. Task-specific parameters alter quantities that affect contact dynamics, such as mass, friction, damping, or joint properties. Each task declares valid bounds for the relevant objects; the sampled scene is rejected if it violates task feasibility.

Spatial. Task-specific translational and rotational offsets alter object poses and, where appropriate, robot placement or articulated initial states. For evaluation, parameters whose names encode metric or angular offsets are expanded to $1.5\times$ their collection range.

Continuous parameters are sampled deterministically from seeds $\{1001, 1002, 1003, 1004, 1005\}$. Invalid configurations are resampled up to 16 times under the same registered protocol. Each task is evaluated in 100 episodes: 20 nominal episodes and 20 for each generalization axis. Within a generalized condition, every seed is used four times. Layout assignments are rotated across conditions, yielding 34, 33, and 33 episodes for the three layouts over the full evaluation. A compared set of policies receives the same condition, seed, layout, and reset schedule, making differences in score attributable to the policy rather than to evaluation sampling.

C 123-TAXEL TACTILE SYSTEM

C.1 TAXEL LAYOUT AND REPRESENTATION

Evo-DexBench equips each Shadow Hand with 123 virtual taxels. Every taxel is defined by a parent link, a surface center, an outward normal, an in-plane tangent, a finite rectangular patch, its area, and anatomical finger and region labels. The layout is fitted to the visible hand surface; the released fit keeps each taxel within approximately 1 mm of the surface. The left-hand layout is obtained by mirroring the calibrated right-hand layout, so a bimanual system exposes 246 measurements with consistent semantics.

Table 12 summarizes the layout in two complementary ways. The five fingers contain 100 taxels and the palm/sub-palm contains 23. The denser distal coverage captures fingertip and lateral contacts that are important for buttons, knobs, and precision grasps, while the palm coverage captures enveloping and stabilizing contact.

Table 12: Distribution of the 123 taxels on one Shadow Hand.

Grouping	Thumb	Index	Middle	Ring	Little	Palm	Total
By finger/area	29	21	18	18	14	23	123
Region	Distal	Middle	Proximal	Palm	Sub-palm		Total
By anatomy	62	24	14	21	2		123

The policy-facing observation is a non-negative scalar normal force in Newtons for each taxel. Geometry and taxel ordering are versioned as part of the benchmark metadata. This prevents a checkpoint trained with one layout from being silently evaluated with a different ordering or sensor geometry.

C.2 From contacts to tactile force

PhysX reports contact impulses at 120 Hz, whereas policies act and observations are recorded at 30 Hz. At every physics step, a contact is eligible for a taxel only when it acts on the same parent link, lies inside the finite taxel patch with a 1 mm assignment margin, is within 10 mm of the patch plane, and produces compression along the outward taxel normal. When several patches cover one contact, compact spatial weights favor the closest patches and are normalized across the eligible set. Consequently, a covered contact's vector impulse is distributed rather than duplicated, preserving its total impulse.

Impulses are accumulated over all physics steps in a control period and divided by the complete control-period duration. The resulting vector force is projected onto each taxel normal and clamped below at zero, yielding compressive normal force in Newtons. The observation therefore corresponds to an average force over the same 30 Hz interval used by the policy, rather than a potentially noisy instantaneous solver value.

C.3 KineDex interface and No-Touch control

KineDex consumes the 100 finger taxels rather than all 123 measurements. The thumb, index, middle, ring, and little finger contain 29, 21, 18, 18, and 14 taxels, respectively; shorter sequences are zero-padded to form a $5 \times 29 \times 1$ tensor per hand for the shared per-finger encoder. The 23 palm and sub-palm taxels remain available through the benchmark interface but are not passed to this model. The five scalar force-supervision targets are the sums of the same-timestamp taxels for the corresponding fingers.

For each hand, Full KineDex predicts the 30 benchmark motion values followed by five finger-force targets. At deployment, predicted forces are clipped to the range observed during training. A bounded feedback controller compares each target with the current finger force and adds a flexion residual to the corresponding motion command. We use a gain of 0.005 rad/N and cap the total residual at 0.05 rad per finger and control step; joint commands are then clipped to the environment limits.

Following KineDex's practice of retraining a without-tactile-input variant rather than masking touch only at evaluation time, we train No-Touch as a separate, backbone-matched comparison. It retains the tactile encoder but feeds an identity-normalized zero placeholder at both training and inference. It predicts only the 30 motion values per hand, does not construct force targets, and bypasses the deployment-time force controller. Visual inputs, proprioception, demonstrations, backbone, optimizer, and evaluation protocol are otherwise unchanged. This isolates the complete tactile sensing and force-feedback pathway without conflating its effect with a different visual architecture or training budget.

D Detailed Data, Training, and Evaluation Settings

D.1 Dataset and common policy contract

Demonstrations are collected through the benchmark teleoperation interface and stored in a policy-independent LeRobot v3 representation. Each Core task contains 50 successful trajectories and each Gen task contains eight. At 30 Hz, a sample stores synchronized RGB observations, robot state, the command applied by the demonstrator, the natural-language instruction, task-stage state, and tactile measurements when enabled. The simulator advances at 120 Hz, so four physics steps contribute to one policy observation and action. Only successful trajectories are included in the training sets used in this paper.

All policies use a common semantic action interface. For one arm–hand system, the state contains a 7-D end-effector pose and 24 Shadow Hand joint positions (31 dimensions), and the action contains a 6-D end-effector pose increment in the robot-root frame followed by 24 absolute hand-joint targets (30 dimensions). The bimanual interface concatenates the left and right vectors, giving 62-D state and 60-D action. Single-arm and bimanual data are trained as separate checkpoints. The single-arm visual input comprises context, interaction, and right-wrist RGB views. The bimanual input

comprises context, left-wrist, and right-wrist views. State and action normalization statistics are computed from the exact training mixture for each run and are not reused after the mixture changes.

D.2 Policy implementations and adaptations

We retain each method's released architecture, learning objective, and native training workspace. All policies share the benchmark's LeRobot-v3 data interface, compact state/action convention, source-balanced sampler, and network inference boundary. Table 13 therefore lists only model-specific interfaces and adaptations.

Table 13: Policy implementations and Evo-DexBench compatibility adaptations.

Policy	Native model	Policy interface	Model-specific adaptation
$\pi_{0.5}$	OpenPI JAX; pretrained flow-matching VLA	Three RGB views, language, 31-D/62-D state; 50-step chunks	Pad state/action interfaces to 32-D/64-D and discard padded action dimensions
SmolVLA	LeRobot 0.4.0; SmolVLM2-500M with flow-matching expert	Three/four RGB views, language, 31-D/62-D state; 50-step chunks	Pad to 32-D/64-D and expand four projection tensors for bimanual control
DP-C	Convolutional Diffusion Policy with temporal U-Net	Two observation steps; 30-step external action horizon	Use a 32-step internal target for U-Net divisibility and discard the final two steps
DP-T	Transformer Diffusion Policy	Two observation steps; 30-step action horizon	Map benchmark compact state and action channels to the released workspace
KineDex	Multi-view ResNet-18, per-finger tactile CNN, conditional U-Net	Two observation steps; 16-step horizon; motion plus five force targets per hand	Map 123 taxels to 100 finger taxels, expand the action head, and add bounded force feedback
KineDex No-Touch	Backbone-matched KineDex variant	Visual/proprioceptive input; 30 motion values per hand	Feed zero tactile input and remove force supervision and force feedback

$\pi_{0.5}$ Policy. All $\pi_{0.5}$ results in this paper use the official JAX implementation. The adapter extracts the compact benchmark state from the policy-independent record, maps the three camera roles to OpenPI image keys, and prefixes the task instruction with the benchmark control convention. The native OpenPI padding interface presents 32 dimensions for single-arm runs and 64 for bimanual runs; only the semantic 30 or 60 action dimensions are executed. Full model parameters are optimized, and checkpoints are served directly from their JAX Orbax representation.

SmolVLA. We use the official `lerobot/smolvla_base` implementation and retain its SmolVLM2-500M-Video-Instruct backbone and flow-matching action expert. The vision encoder is frozen; the action expert and state projection are trained. Single-arm runs consume the context, interaction, and right-wrist views with 31 state and 30 action dimensions. Bimanual runs add the left-wrist view and use 62 state and 60 action dimensions. The model-facing state/action widths are padded to 32 for single-arm and 64 for bimanual operation; only physical dimensions are normalized, supervised, and executed. To construct the bimanual initialization, we expand the state projection, action-input projection, and action-output weight and bias tensors while loading all remaining released weights unchanged.

DP-C and DP-T. Both variants use the official Stanford Diffusion Policy code path. They share the same observation and external action contract; the only horizon difference is DP-C's 32-step internal target required by its temporal U-Net. At inference, the models replan in receding-horizon fashion after executing eight predicted steps. Changing a training mixture imports model weights when appropriate but reinitializes normalization, optimizer, and schedule state.

KineDex. The released KineDex architecture was originally parameterized for a different tactile array. We parameterize its shared per-finger encoder by input-channel and finger count while preserving the visual encoder, diffusion backbone, and training objective. The per-hand action head is extended from the benchmark's 30 motion dimensions to 35 dimensions by appending five force targets. Full and No-Touch checkpoints are trained separately because their output heads and losses differ.

D.3 Optimization and transfer protocols

Table 14 reports the settings used for the paper results. All schedules use cosine decay from the stated peak to minimum learning rate. Batch size is global per optimization step. Unless listed as an exception, $\pi_{0.5}$ is trained for 25 epochs. DisplayPowerButton, DishwasherKnob, and GlassesFolding use 50 epochs, and CandleToCabinet uses 100 epochs. The diffusion baselines and both KineDex variants use 600 epochs. SmolVLA Core checkpoints use 25 epochs, while DisplayPowerButton, DishwasherKnob, and GlassesFolding use 100 epochs. All training runs use seed 42.

Table 14: Training hyperparameters. KineDex No-Touch uses the same settings as Full KineDex.

Policy	Epochs	Peak LR	Minimum LR	Batchsize
$\pi_{0.5}$	25[†]	5×10^{-6}	5×10^{-7}	32
SmolVLA	25[‡]	1×10^{-4}	2.5×10^{-6}	32
DP-T	600	1×10^{-4}	1×10^{-6}	32
DP-C	600	1×10^{-4}	1×10^{-6}	32
KineDex	600	1×10^{-4}	1×10^{-5}	32
KineDex No-Touch	600	1×10^{-4}	1×10^{-5}	32

[†]DisplayPowerButton, DishwasherKnob, and GlassesFolding use 50 epochs; CandleToCabinet uses 100. [‡]SmolVLA DisplayPowerButton, DishwasherKnob, and GlassesFolding use 100 epochs.

For the standard $\pi_{0.5}$, SmolVLA, DP-C, DP-T, and KineDex baselines, the Gen task contributes one half of each minibatch and the four Core tasks in its family uniformly share the other half (12.5% each). This prevents the 50-trajectory Core sets from overwhelming the eight-shot target.

The adaptation-strategy study fixes $\pi_{0.5}$, the target demonstrations, and a 50-epoch target-stage budget. *Gen only* trains from the released base parameters using only the eight target demonstrations. *Full Core* $\rightarrow$ *Gen* initializes target training from the terminal Core checkpoint; *Early Core* $\rightarrow$ *Gen* instead uses the epoch-5 Core checkpoint. In both sequential conditions, target normalization and optimizer state are reinitialized. *Core + Gen joint* trains on the 1:1 mixture for the full schedule. Runs identified as holdout in the development logs use a separate seed and are not included in the reported strategy table.

The Core-source relatedness study changes only the Core source paired with a fixed Gen target. Each mixed run preserves the 1:1 Core–Gen ratio and balances the selected Core tasks within their half of the minibatch. DisplayPowerButton and DishwasherKnob use 50 epochs; CandleToCabinet uses 100. The DishwasherKnob replications with DP-T and KineDex follow the same source ratios and use the matched checkpoints reported in the small-model result summary.

D.4 Evaluation, metrics, and uncertainty

Every task score aggregates exactly 100 frozen-policy episodes: 20 nominal and 20 for each of camera, lighting, physical, and spatial variation. Compared models receive the same registered episode list. An episode terminates on success, a task-specific irreversible failure, or the registered horizon. No model update, online normalization update, or test-time adaptation occurs.

Success rate is the fraction of episodes satisfying the terminal task criterion. StageScore is the mean normalized index of the highest consecutively completed task stage, as defined in Eq. 1. Aggregate scores are macro-averaged over tasks, so a task contributes equally regardless of trajectory length or number of stages.

For a single task, uncertainty in SR can be computed directly from the 100 binary outcomes. We use the 95% Wilson interval with $z = 1.96$,

$$\frac{\hat{p} + z^2/(2n)}{1 + z^2/n} \pm \frac{z}{1 + z^2/n}\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}, \qquad n = 100. \tag{2}$$

For a macro-average over T fixed tasks, we use

$$\bar{p} = \frac{1}{T}\sum_{t=1}^{T}\hat{p}_t, \qquad \mathrm{SE}(\bar{p}) = \frac{1}{T}\sqrt{\sum_{t=1}^{T}\frac{\hat{p}_t(1-\hat{p}_t)}{100}}, \tag{3}$$

and report $\bar{p} \pm 1.96\,\mathrm{SE}(\bar{p})$ when an interval is required. This interval quantifies rollout uncertainty for a fixed set of trained policies and tasks; it is not an estimate of training-seed variability. Condition-specific SR has a resolution of five percentage points because each condition contains 20 episodes. We report StageScore point estimates without an unsupported confidence interval because the archived summaries do not retain condition-wise episode-level StageScores.

E Detailed Experimental Results

E.1 Complete baseline results

Table 15 reports the complete standard evaluation underlying Table 3. Overall entries are **SR (%) / StageScore**; the five condition columns report SR because the archived summaries do not retain condition-wise StageScore. The table contains the 15 tasks evaluated by all five policies.

Table 15: Complete baseline results. Overall is SR (%) / StageScore; Nom., Cam., Light., Phys., and Spat. are condition-specific SR (%).

Policy	Task	Overall	Nom.	Cam.	Light.	Phys.	Spat.
$\pi_{0.5}$	LampButton	30% / 35.7	40	15	35	40	20
	WallSwitch	60% / 74.1	60	50	75	55	60
	PhoneHomeButton	16% / 24.4	30	25	10	0	15
	ToasterSlider	42% / 75.2	40	55	50	20	45
	DisplayPowerButton	20% / 39.4	25	15	15	20	25
	FaucetHandle	21% / 29.1	15	15	5	40	30
	MouseWheel	63% / 80.8	70	80	55	40	70
	CoffeeKnob	59% / 82.9	65	50	45	75	60
	KettleLidTab	79% / 85.2	95	100	85	95	20
	DishwasherKnob	46% / 62.5	50	70	45	60	5
	LaptopClosing	47% / 60.5	50	55	30	65	35
	StaplerPaperMark	42% / 60.9	25	45	65	35	40
	ScissorsOpening	23% / 55.9	40	40	20	15	0
	BoxLidOpening	18% / 41.1	20	30	20	15	5
	GlassesFolding	2% / 6.0	0	5	0	5	0
DP-T	LampButton	15% / 12.6	15	20	5	20	15
	WallSwitch	35% / 57.8	55	15	25	55	25
	PhoneHomeButton	1% / 2.2	5	0	0	0	0
	ToasterSlider	49% / 79.6	70	45	65	20	45
	DisplayPowerButton	33% / 52.8	30	55	35	30	15
	FaucetHandle	24% / 27.2	20	30	15	25	30
	MouseWheel	37% / 65.0	50	50	35	15	35
	CoffeeKnob	11% / 49.3	10	5	15	0	25
	KettleLidTab	16% / 51.7	10	25	20	20	5
	DishwasherKnob	41% / 56.6	50	35	60	55	5

Continued on next page

Table 15 continued

Policy	Task	Overall	Nom.	Cam.	Light.	Phys.	Spat.
	LaptopClosing	49% / 59.2	35	60	60	75	15
	StaplerPaperMark	2% / 26.9	5	0	0	5	0
	ScissorsOpening	6% / 45.4	0	0	10	20	0
	BoxLidOpening	13% / 39.1	10	15	10	10	20
	GlassesFolding	6% / 16.8	5	5	10	0	10
	LampButton	26% / 28.7	20	25	15	45	25
	WallSwitch	68% / 79.7	75	65	65	80	55
	PhoneHomeButton	8% / 13.7	10	10	15	0	5
	ToasterSlider	64% / 84.0	90	70	80	25	55
	DisplayPowerButton	9% / 31.5	10	10	10	5	10
	FaucetHandle	19% / 23.0	20	15	30	20	10
	MouseWheel	47% / 69.0	75	65	40	10	45
DP-C	CoffeeKnob	33% / 71.2	35	40	40	15	35
	KettleLidTab	76% / 81.5	85	80	95	95	25
	DishwasherKnob	68% / 76.7	70	90	90	70	20
	LaptopClosing	85% / 91.3	70	90	90	85	90
	StaplerPaperMark	24% / 50.6	20	30	20	15	35
	ScissorsOpening	29% / 58.6	60	25	45	10	5
	BoxLidOpening	15% / 39.6	15	15	10	15	20
	GlassesFolding	6% / 20.4	0	0	20	5	5
	LampButton	24% / 23.2	30	20	15	35	20
	WallSwitch	76% / 85.6	85	75	75	65	80
	PhoneHomeButton	5% / 14.8	5	5	15	0	0
	ToasterSlider	63% / 84.0	70	70	75	45	55
	DisplayPowerButton	18% / 40.8	20	15	20	30	5
	FaucetHandle	16% / 22.0	20	15	15	20	10
	MouseWheel	53% / 72.5	75	40	70	20	60
KineDex	CoffeeKnob	33% / 68.9	25	20	45	20	55
	KettleLidTab	62% / 75.3	65	65	75	80	25
	DishwasherKnob	55% / 67.9	65	60	75	55	20
	LaptopClosing	69% / 75.9	60	70	80	80	55
	StaplerPaperMark	33% / 54.7	45	15	25	40	40
	ScissorsOpening	26% / 56.8	45	45	25	15	0
	BoxLidOpening	26% / 46.4	25	15	20	55	15
	GlassesFolding	4% / 16.4	0	10	0	0	10
	LampButton	41% / 35.2	45	30	45	50	35
	WallSwitch	66% / 78.0	70	70	65	55	70
	PhoneHomeButton	8% / 12.8	15	5	20	0	0
	ToasterSlider	37% / 74.2	35	45	55	15	35
	DisplayPowerButton	18% / 36.0	25	25	5	20	15
	FaucetHandle	12% / 22.2	5	15	20	15	5
	MouseWheel	62% / 77.8	65	50	85	40	70
SmolVLA	CoffeeKnob	45% / 72.6	40	45	50	35	55
	KettleLidTab	76% / 84.1	95	80	90	100	15
	DishwasherKnob	42% / 57.3	55	40	50	50	15
	LaptopClosing	53% / 59.9	50	50	65	40	60
	StaplerPaperMark	12% / 37.0	25	0	15	5	15
	ScissorsOpening	15% / 49.9	25	15	15	15	5
	BoxLidOpening	21% / 40.8	30	10	10	30	25
	GlassesFolding	0% / 5.2	0	0	0	0	0

E.2 Complete Core-source relatedness results

Table 16 reports all retained $\pi_{0.5}$ source conditions. In a mixed condition, the target receives 50% of samples and the listed Core source receives the remaining 50%. Gen-only values use the same retested reference runs as Table 4 and the transfer heatmap.

Table 16: Complete $\pi_{0.5}$ Core-source relatedness results. Entries are SR (%) / StageScore; bold marks the strongest retained result for each target.

Gen target	Epochs	Training data	SR / StageScore
DisplayPowerButton	50	Gen only	15% / 29.7
	50	Button-driven Core + Gen	**20% / 39.4**
	50	Rotational manip. Core + Gen	17% / 34.7
	50	Access-and-transfer Core + Gen	12% / 33.0
DishwasherKnob	50	Gen only	36% / 55.1
	50	Button-driven Core + Gen	40% / 61.0
	50	Rotational manip. Core + Gen	**46% / 62.5**
	50	Access-and-transfer Core + Gen	32% / 52.4
CandleToCabinet	100	Gen only	0% / 4.5
	100	Button-driven Core + Gen	0% / 7.8
	100	Rotational manip. Core + Gen	1% / 8.0
	100	Access-and-transfer Core + Gen	**3% / 19.3**

E.3 Core-set breadth and composition

Table 17 compares the complete matched four-task Core family with two complementary two-task subsets. The comparison holds the Gen data and epoch budget fixed but changes both source-set breadth and composition.

Table 17: Core-set breadth/composition ablation for $\pi_{0.5}$. Entries are SR (%) / StageScore; bold marks the best result for each target.

Gen target	Training data	SR / StageScore
DisplayPowerButton	Gen only	15% / 29.7
	LampButton + ToasterSlider + Gen	17% / 32.9
	WallSwitch + PhoneHomeButton + Gen	**20%** / 39.3
	Four matched Core + Gen	**20% / 39.4**
DishwasherKnob	Gen only	36% / 55.1
	MouseWheel + CoffeeKnob + Gen	35% / 53.6
	FaucetHandle + KettleLidTab + Gen	37% / 60.7
	Four matched Core + Gen	**46% / 62.5**
CandleToCabinet	Gen only	0% / 4.5
	CupFromCabinet + BananaFromDrawer + Gen	2% / 8.5
	MicrowaveBowl + TennisBallTransfer + Gen	1% / 9.3
	Four matched Core + Gen	**3% / 19.3**

E.4 Complete tactile ablation results

Table 18 compares KineDex with and without the complete tactile pathway. Full KineDex improves mean SR by 3.8 points on Rotational manipulation and 2.2 points on Articulated configuration control, with StageScore gains of 3.0 and 3.5 points. The effect is heterogeneous: touch helps strongly on StaplerPaperMark, CoffeeKnob, KettleLidTab, and BoxLidOpening, but does not improve every task. This supports a task-dependent tactile benefit rather than a claim of uniform dominance.

Table 18: Task-wise KineDex tactile ablation. Entries are SR (%) / StageScore.

Task	No-Touch	Full
FaucetHandle	15% / 16.3	16% / 22.0
MouseWheel	46% / 67.0	53% / 72.5
CoffeeKnob	25% / 63.9	33% / 68.9
KettleLidTab	54% / 70.7	62% / 75.3
DishwasherKnob	60% / 73.5	55% / 67.9
LaptopClosing	72% / 77.7	69% / 75.9
StaplerPaperMark	18% / 47.1	33% / 54.7
ScissorsOpening	36% / 62.8	26% / 56.8
BoxLidOpening	18% / 39.9	26% / 46.4
GlassesFolding	3% / 5.0	4% / 16.4
Rotational manip. mean	40.0% / 58.3	43.8% / 61.3
Articulated control mean	29.4% / 46.5	31.6% / 50.0